UM0701-02

PN532 User Manual

Rev. 02 **User Manual**

Document information

Info	Content
Keywords	NFC, PN532, V1.6
Abstract	This document describes the firmware V1.6 embedded in the PN532.

Revision history

01	2007-04-27	Initial version for firmware version **V1.5 (PN532/C105)**
02	2007-11-05	Version for firmware version **V1.6 (PN532/C106)**

Contact information

For additional information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **sales.addresses@www.nxp.com**

1. Introduction

1.1 Purpose and Scope

The PN532 is a highly integrated transmission module for contactless communication at 13.56 MHz including microcontroller functionality based on an 80C51 core with 40 Kbytes of ROM and 1 Kbytes of RAM.

The PN532 combines a modulation and demodulation concept completely integrated for different kinds of contactless communication methods and protocols at 13.56 MHz with an easy-to-use firmware for the different supported modes and the required host controller interfaces.

This document describes the firmware embedded in the PN532 chip, in particular the global behavior in the system depending if the PN532 device is used as initiator or target.

1.2 Intended audience

This document has been written to allow the use of the PN532 from the host controller point of view.

All the RF protocols used by the PN532 are not described in this document. The reader is supposed to have knowledge on NFCIP-1 (Reference **Error! Reference source not found.**) and ISO/IEC14443 (Reference **Error! Reference source not found.**).

1.3 Glossary

APDU	Application Protocol Data Unit
ATQA	Answer To Request, type A
ATQB	Answer To Request, type B
C-APDU	Command APDU
CIU	Contactless Interface Unit
CL	ContactLess
CLAD	ContactLess Active Detection
CPU	Central Processing Unit
CT	Cascade Tag
DEP	ISO/IEC18092 Data Exchange Protocol
DRI	Bit duration of Target to Initiator
DSI	Bit duration of Initiator to Target
FSL	Maximum value for the Frame Length
HSU	High Speed UART
I2C	Inter Integrated Circuit
IC	Integrated Circuit
ID	Card Identifier
N/A	Not Applicable
NAD	Node ADdress
N/I	Not Implemented
NU	Not Used
PCB	Protocol Control Byte (ISO/IEC14443-4)

PCD	Proximity Coupling Device (Contactless PCD)
PFB	Control Information for Transaction (NFCIP-1)
PICC	Proximity IC Card
PPS	Protocol and Parameter Selection
R-APDU	Response APDU
RATS	Request for Answer To Select
RFU	Reserved for Future Use
SAM	Security Access Module
SDD	Single Device Detection
SPI	Serial Peripheral Interface
SRS	Software Requirements Specification
TSN	Time Slot Number
TBD	To Be Defined
TPE	NFC Transport Protocol Equipped (DEP: Data Exchange Protocol)
T=CL	ISO/IEC14443-4 protocol
UID	Unique Identifier, Type A

1.4 References

[1]	ISO/IEC 14443-3	Identification cards – Contactless integrated circuit(s) cards - Proximity card(s) Part 3: Initialization and anti-collision
[2]	ISO/IEC 14443-4	Identification cards – Contactless integrated circuit(s) cards - Proximity card(s) Part 4: Transmission protocol
[3]	ISO/IEC 18092[1]	Near Field Communication - Interface and Protocol (NFCIP-1)
[4]	PN532/C1 Product Datasheet	PN532 NFC Controller Product data sheet
[5]	AN10449	PN532 Application Note
[6]	AN10609-2	PN532 Application Note, C106 appendix

[1] Purchase of an NXP Semiconductors IC that complies with one of the NFC Standards (ISO/IEC18.092;ISO/IEC21.481) does not convey an implied license under any patent right on that standards.
A license for the portfolio of the NFC Standards patents of NXP B.V. needs to be obtained at Via Licensing, the pool agent of the NFC Patent Pool, e-mail: info@vialicensing.com.

1.5 General presentation of the PN532

The embedded firmware and the internal hardware support the handling of the host controller protocol for the different interfaces (PC, mobile base-band CPU, PDA CPU …) as

- I2C,
- SPI, specific hardware implementation is needed to use the PN532 in LowVbat mode when PVDD is absent. See the application note [6].
- Serial High Speed UART (HSU).

The host controller protocol is defined in chapter 0.

The firmware of the PN532 supports the following operating modes:

- LowVbat feature: see § 6.3.4.1
- PCD mode for FeliCa, ISO/IEC14443-3 Type A, Mifare, ISO/IEC14443-4 Type A and Innovision Jewel cards
- Card interface mode for FeliCa, ISO/IEC14443-3 Type A and Mifare in combination with secure microcontroller companion chip,
- NFC IP-1 mode.
  The NFC IP-1 mode offers different baud rates up to 424 kbps. The PN532 handles the complete NFC framing and error detection.
- PCD mode for ISO/IEC14443-3 Type B and ISO/IEC14443-4 Type B cards[2].

In this document:

- PN532 refers to PN532/C106.

Using a set of high-level commands described in chapter 4 configures all the different operating modes of the PN532.

[2] This NXP IC is licensed under Innovatron's ISO/IEC 14443 Type B patent license.

2. Configuration Modes

The PN532 has 3 possible modes that can be chosen by using two GPIOs during the reset phase of the IC:

Table 1. Configuration modes

Mode	Selection Pins	
	P70_IRQ (pin #25)	**P35** (pin #19)
Standard	1	1
	0	1
PN512 emulation	1	0
RF field ON	0	0

2.1 Standard Mode

This is the default mode of the PN532.
The description of this mode is detailed in this document starting from chapter 0.

2.2 PN512 emulation mode

In this test mode, the PN532 is configured to act as real PN512 IC using serial interface.

The PN512 is a transmission module for contactless communication at 13.56 MHz. It integrates a modulation and demodulation concept for different kind of contactless communication methods and protocols.
Then, the PN532 can be easily interfaced with the PN512 dedicated host controller software, as e.g. **Joiner PC Serial**.

The link used is RS232 at 9600 bauds[3]. It is not possible to change the value of the baud rate; the *SerialSpeedReg* register is not emulated.

The emulation of the PN512 IRQ pin is supported as well; the pin used is P70_IRQ. The level of the P70_IRQ pin is low when an interrupt occurs. The bit *IRQInv* in the register *CIU_CommIEnReg* has no effect (see **Error! Reference source not found.**).

[3] The RS232 link used here is the standard UART, not the High Speed UART. Consequently, in this mode the PN532 must be interconnected with P30 (pin#24) for the HSU_RX line and P31 (pin#31) for the HSU_TX line.

2.3 RFfieldON Mode

In this mode, the PN532 is configured to switch on its RF field immediately after the reset.

The modulation and the baud rate used depend on the selection GPIOs P33_INT1 and P34/SIC_CLK and random data bytes are continuously sent.

In this mode, the temperature sensor is not activated, so that tests can be done at temperature higher than 125°C.

Table 2. TX framing and TX speed in RFfieldON configuration

	Selection Pins	
TX framing – TX speed	**P33_INT1** (pin #33)	**P34/SIC_CLK** (pin #34)
Mifare - 106 kbps	1	1
	0	0
FeliCa - 212 kbps	0	1
FeliCa - 424 kbps	1	0

3. Power management

The PN532 is dedicated for mobile equipments, where the power consumption is a very important parameter.

The design of the firmware embedded in the PN532 takes care of that, in a sense that it minimizes the overall power consumption.

This chapter defines the strategy used to save current consumption. The firmware can play with different parameters described hereafter:

- CPU frequency,
- Power modes of the CPU,
- Power modes of the CL front-end,
- Management of pin configuration.

3.1.1 CPU frequency

Three different states can be considered for the PN532:

- **WAIT**
  This is the state in which the PN532 is when it does not manage any communication either with its host controller or with an external device (initiator, target, card or PCD),
- **HOST**
  In this state, the PN532 is exchanging data with its host controller (reception or transmission),
- **RFCOM**
  In this state, the PN532 is exchanging data with the RF channel (reception or transmission).

The CPU frequency is automatically changed during a transition from one state to another one. The frequencies used in each state are stored in three variables (bFreqHost, bFreqWait and bFreqRFCom); possible values are 0x00 for 27.12 MHz, 0x01 for 13.56 MHz and 0x02 for 6.78 MHz.

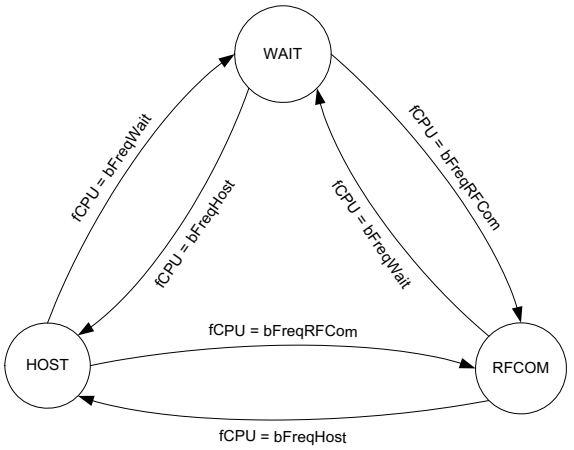


Fig 1. States of the PN532 regarding CPU frequency

Table 3. CPU frequency used

The host controller can modify the default power mode by a **`WriteRegister`** command (§7.2.5, p: 78).

State	Variable Name	Address	Default value
HOST	bFreqHost	0x02FD	0x00
RFCOM	bFreqRFCom	0x02FE	0x00
WAIT	bFreqWait	0x02FF	0x02

3.1.2 Power modes of the PN532

The PN532 has different power modes that are listed in the following table (refer to Error! Reference source not found. to have a complete description).

3.1.2.1 Power modes for CPU

Table 4. Power modes for CPU

Power Mode	Description
Hard Power Down	The CPU is in reset state. This mode can be reached only by an external action on RSTPDN pin, not by a firmware action.
Normal	The CPU is running.
Power Down	Oscillator is stopped, needs delay to be waken up (typ. 500µs).

3.1.2.2 Power modes for Contact Less interface

Table 5. Power modes for CL interface

Power Mode	Description
Hard Power Down	The contactless UART and the analog front end are in reset state. This mode can be reached only by external action on RSTPDN pin, not by a firmware action.
CL_A	The contactless UART is running. The analog front end is operational. RF field is **not** generated.
CL_B	The contactless UART is running. The analog front end is operational. **RF field** is generated.
CL_C	The contactless UART is in Power Down mode. The analog front end is partially operational (only the RF level detector is active). RF field is **not** generated.
CL_D	The contactless UART and the analog front end are set in the mode in which the power consumption is the minimum, i.e. power down with RF level detector not activated.

3.1.3 Operating modes of the PN532

3.1.3.1 Mode dispatcher

The firmware adapts the overall power consumption to the real needs depending on the state where it is.

Several cases are to be considered; the power modes involved being different:

- Standby mode,
- LowVbat mode,
- Virtual Card mode,
- Wired Card mode,
- Initiator / PCD mode,
- Target / PICC mode.

The transition between the five defined modes is automatically done by the firmware, either due to a change in the internal state machine (PN532 acting as target has been released for example) or by the reception of a command from the host controller.

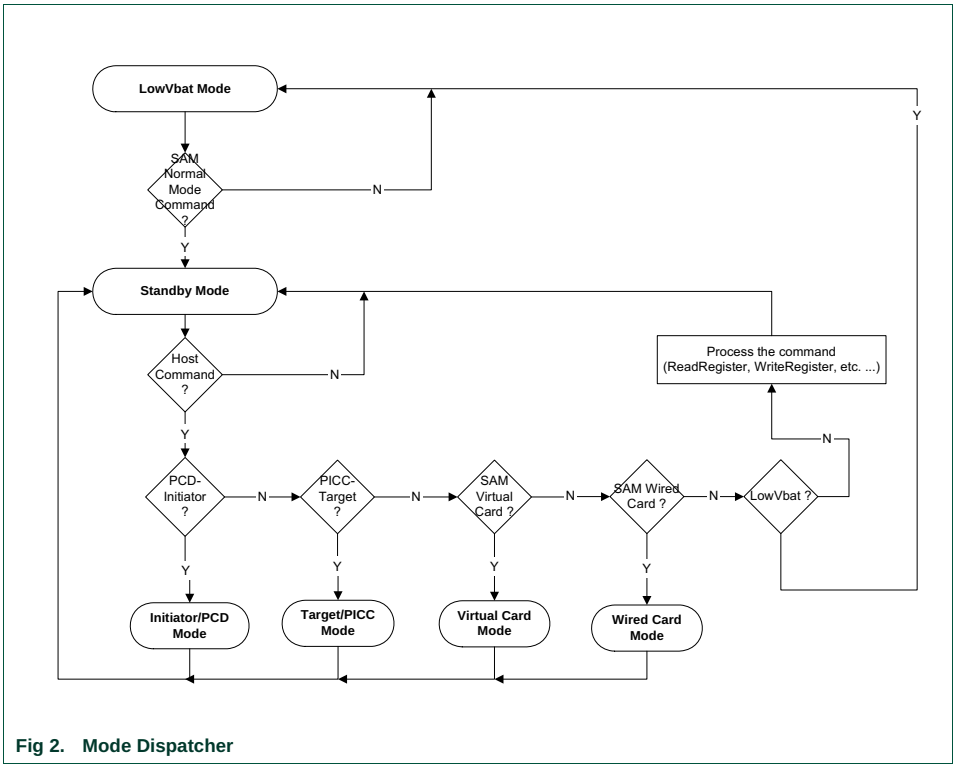


Fig 2. Mode Dispatcher

3.1.3.2 Standby mode

The Standby mode is the mode used when no session of LowVbat, initiator, target or virtual card is running.

The PN532 goes into the mode chosen by the host controller (see Table 3, p: 9).

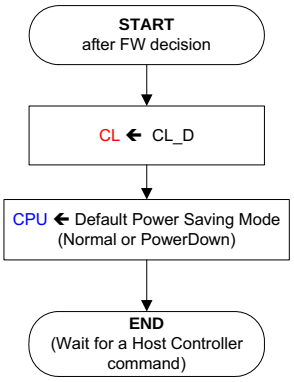


Fig 3. Standby mode after FW decision

The default power saving mode for the CPU defined in Fig 3 can be either:

- **Power Down mode.**
  This is the preferred mode to save power consumption.
  Before sending a new command, the host controller must wake the PN532 up.
  This is the drawback of this mode; for each command sent, the host controller must take care of a wakeup mechanism that is slightly different depending on the link used.
- **Normal mode.**
  In that case, there is no power saved but the PN532 reacts as fast as possible on the host controller interface.

The host controller can modify the default power mode by a **`WriteRegister`** command (§7.2.5, p: 78).

Table 6. CPU PowerMode used

Power Mode	Variable Name	Default Value	Address	Value
Normal	bCPUPowerMode	0x00	0x02FC	0x00
Power Down				0x02

3.1.3.3 LowVbat mode

The LowVbat mode is the starting mode after reset (and power-up).

When within that mode, the host must send a **SAMConfiguration** command with Normal mode parameter (see **SAMConfiguration** command (§7.2.10, p: 89)) in order to access other modes (see **Initialization Sequence**, §3.1.3.8, p: 19).In LowVbat mode, the PN532 enables a transaction with the SAM without informing the host.

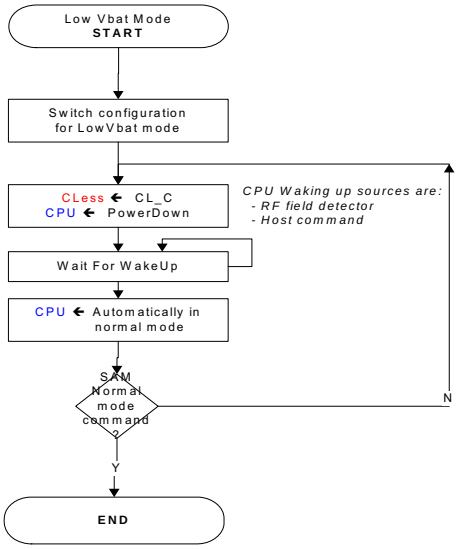


Fig 4. LowVbat mode

Be careful, with SPI as a host interface, a specific hardware implementation is needed to use the LowVbat mode when PVDD is absent. See the application note [6].

3.1.3.4 Initiator / PCD mode

This mode is the one used when the PN532 has at least one target / card activated. All the initiator commands listed in Table 12, p: 64 are using this mode.

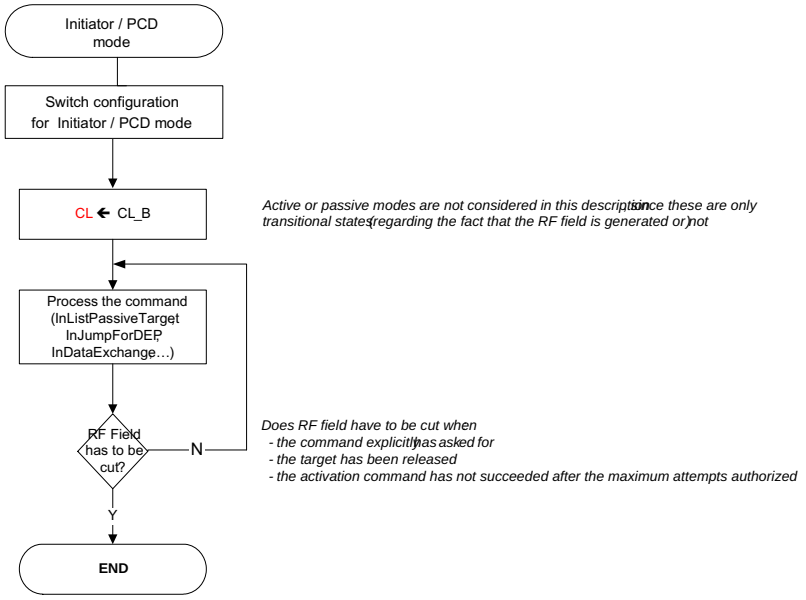


Fig 5. Initiator / PCD mode

3.1.3.5 Target / PICC Mode

This mode is the one used when the PN532 is configured as target or ISO/IEC14443-4 card.
All the target commands listed in Table 12, p: 65 are using this mode.

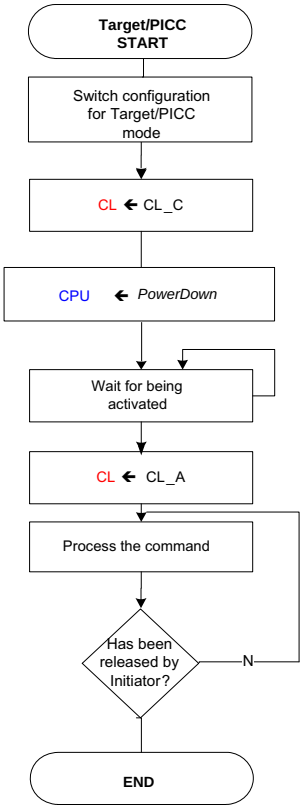


Fig 6. Target / PICC mode

3.1.3.6 Virtual Card mode

When within that mode, the host must send a **SAMConfiguration** command with Normal mode parameter in order to access other modes (see **SAMConfiguration** command (§7.2.10, p: 89)).

The PN532 is in charge to watch over the NFC-WI/S2C link established between an external reader (PCD) and the SAM companion chip.
As long as there is no RF field detected, the PN532 in Power Down mode. When a potential transaction has been detected (by the means of the CLAD line or SigIn activities) or potential timeout occurred, the PN532 informs the host controller (see **SAMConfiguration** command (§7.2.10, p: 89)).

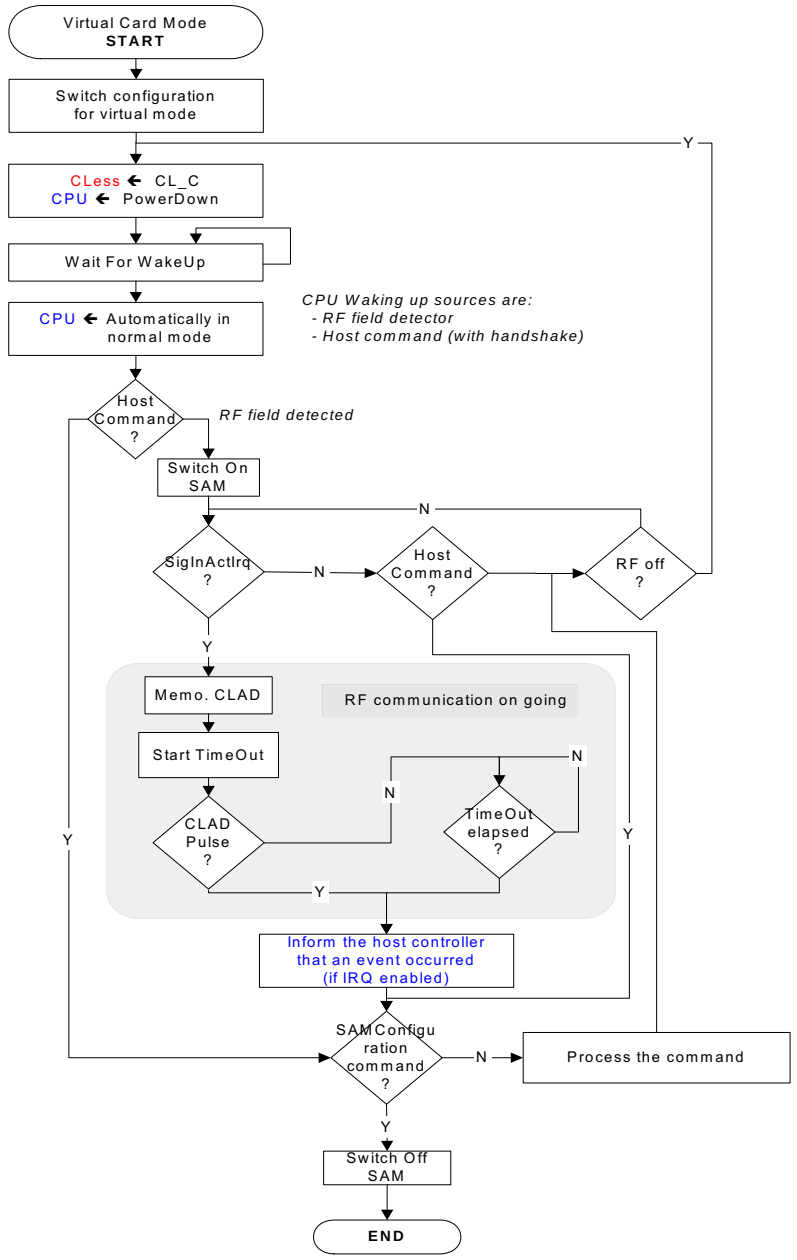


Fig 7. Virtual card mode

3.1.3.7 Wired Card mode

When within that mode, the host must send a **`SAMConfiguration`** command with Normal mode parameter (see **`SAMConfiguration`** command (§7.2.10, p: 89)) in order to access other modes.

The host controller can access to the SAM with standard PCD commands (**`InListPassiveTarget`**, **`InDataExchange`** …),

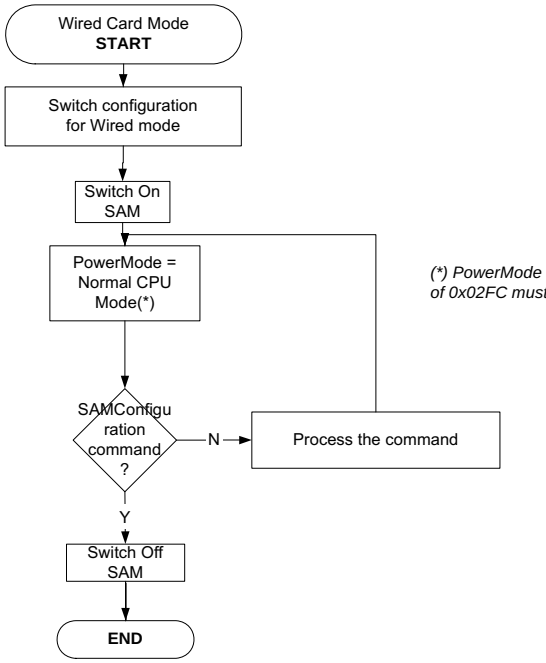
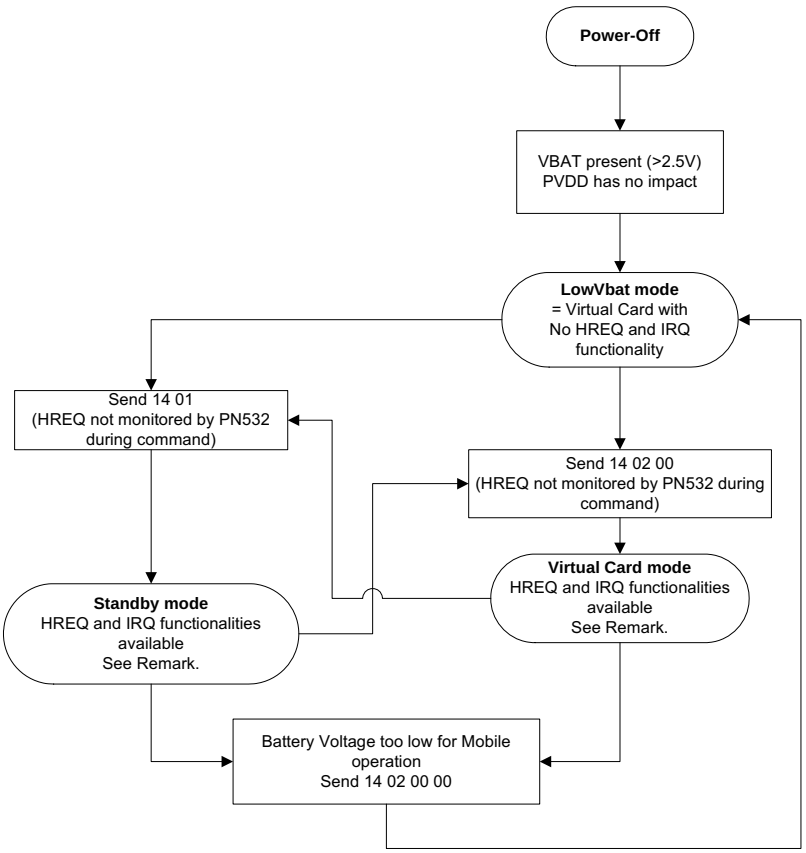


Fig 8. Wired Card mode

3.1.3.8 Initialization sequence

The diagram only describes the cases when after power-up the user wants to program the PN532 in Standby mode (using 14 01 command) or Virtual Card Mode (14 02 00 command).



Remark: In that modes, in order to fullfill the application requirements, any commands of the User Manual can be sent using HREQ and IRQ informations. These scenarios are not described in the diagram.

Fig 9. Initialization sequence

3.1.3.9 Management of GPIO configuration

A management of GPIO configuration of the ports P3 and P7 of the PN532 is implemented in order to reduce the power consumption in power-down mode.

If a GPIO is forced in low state by external conditions, the following actions are taken before going into power-down mode:

- The initial configuration of the GPIO is saved (input, quasi-bidirectional or output mode),
- The GPIO is then configured in input mode.

When the PN532 exits the power-down mode, the initial configuration of the GPIO is restored. Therefore, every GPIO of the ports P3 and P7 recovers its initial configuration.

3.1.3.10 Management of RF field in the activation commands

The activation commands are the first RF communication commands used to initialize a communication session. The usual activation commands are **`InListPassiveTarget`**, **`InJumpForDEP`**, **`and InJumpForPSL`** (see chapter List of commands Table 12, p: 64).

When these commands are launched, they send activation requests until a target is found.

The abortion of an on-going activation command will automatically switch off the RF field in order to reduce the power consumption.

The RF field is also switch off if no target has been found before the number of retries (see §7.3.1, p: 101) is over.

4. ISO/IEC14443-4 PICC emulation concept

The PN532 device can behave like an ISO/IEC14443-4 Type A PICC.

In this mode, all the commands (C-APDU) coming from the ISO/IEC14443-4 PCD are transmitted through the PN532 to the host controller.

The host controller is responsible for elaborating the R-APDU responses that the PN532 will send to the ISO/IEC14443-4 PCD.

A mechanism of automatic waiting time extension (S(WTX) request) is used by the PN532 in order to bypass the potential problem of a time out that could happen if the host controller takes too much time to send back the R-APDU.

The host controller has the possibility to disable the ISO/IEC14443-4 PICC emulation (see **`SetParameters`** command §7.2.9, p: 85).

Features supported:

- ISO/IEC14443-4 Type A PICC,
- Automatic predefined ATS response (only the historical bytes can be personalized),
- PPS handling (106, 212 and 424 kbps) with automatic data rate switching,
- ISO/IEC14443-4 protocol management (S blocks, R blocks, I blocks, chaining, errors handling),
- NAD,
- CID,
- Short APDU
    - Up to 261 bytes in the way from the external PCD to the PN532 emulating the PICC:
      CLA/INS/P1/P2/P3 + 255 data bytes + Le = **261** bytes
    - Up to 258 bytes in the way from PICC to PCD:
      256 data bytes + SW1/SW2 = **258** bytes

Features not supported:

- Type B PICC,
- Extended APDU.

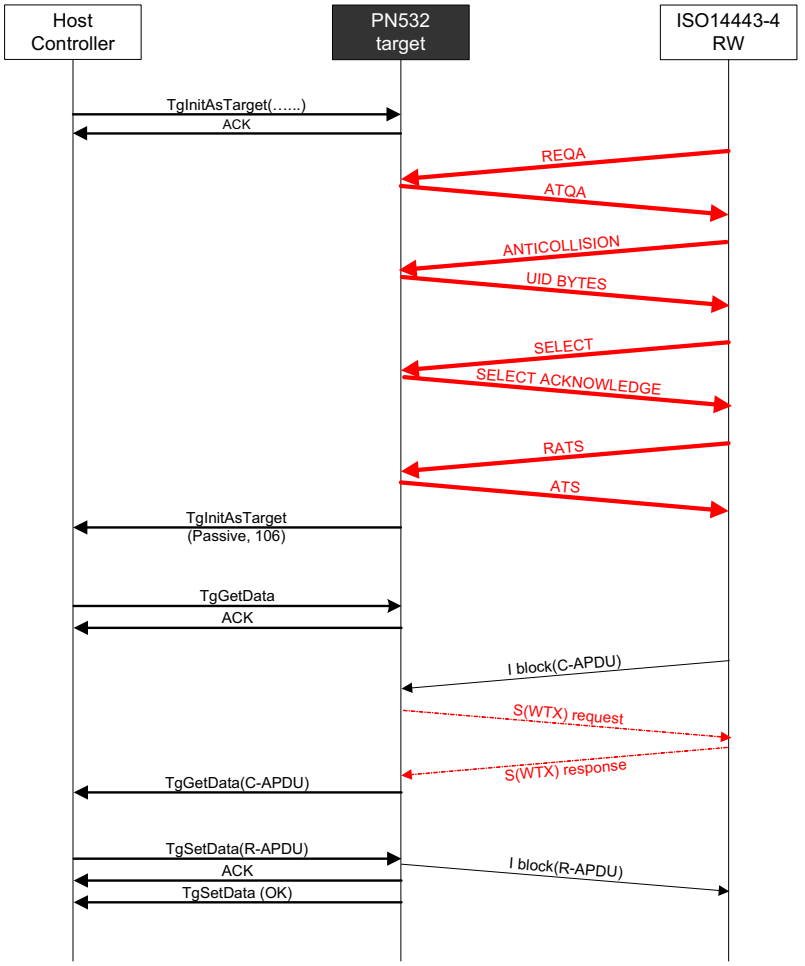


Fig 10. ISO/IEC14443-4 PICC emulation

5. Over-current detection

The PN532 integrates a mechanism of over-current detection. This functionality prevents the PN532 from accidental over-currents, which can cause serious damages to the power supply circuitries and the PN532 itself. Moreover, the user is informed of the occurrence of any over-current events.

This functionality is normally deactivated after the power-on-reset. To use this functionality, the user can activate it by enabling the interrupt number #14 (IE1_0). It is done by setting the bit 0 of the SFR interrupt enable register IE1 (@0xE8). It can be performed using the **`WriteRegister`** command (§7.2.5, p: 78).

The occurrence of over-current event is checked before executing the following RF communication commands:

Initiator / PCD: all Initiator commands listed Table 12, p: 64

Active mode target: **`TgGetData, TgSetData, TgSetMetaData, TgResponseToInitiator, TgInitAsTarget`**

If it has happened, the RF command is not executed and a specific error code is returned (**Status** = 0x2D). After that, any new RF communication command can be submitted and executed as usual (if the root cause has disappeared).

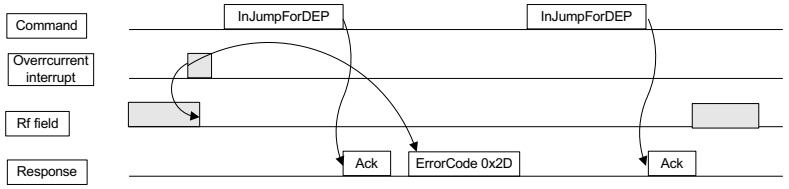


Fig 11. Over-current detection before a RF communication command

If an over-current event occurs during the execution of previously given RF communication commands, the RF field is switched off, the command is aborted and a specific error code is returned (**Status** = 0x2D). In the same way as described before, new RF communication command can be executed after the error code signaling the over-current issue has been returned.

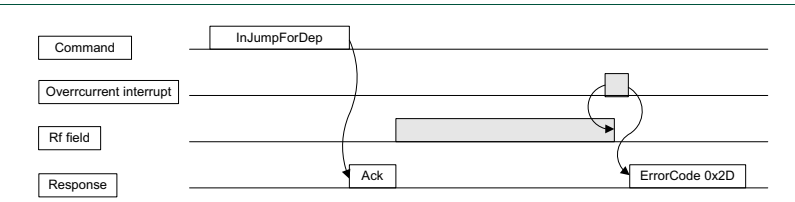


Fig 12. Over-current detection during a RF communication command

The over-current detection has no impact on the miscellaneous commands (**`GetFirmwareVersion`**, **`GetGeneralStatus`**, etc…; see chapter List of commands Table 12, p:64). No error code is returned and commands are executed normally.

In case of over-current during the execution of an **`InListPassiveTarget`** or **`TgInitAsTarget`** commands, the RF field is switched off and the command is not answered. The host shall stop the command and if the command is resent, the answer received is 0x2D.

6. Host controller Interfaces

6.1 General points

6.1.1 Possible links

The system host controller can communicate with the PN532 by using the SPI, I2C or HSU (High Speed UART) serial links.

The protocol between the host controller and the PN532, on top of these physical links is described in §6.2, p: 28.

Only one link can be used at once, and the choice is done by a hardware configuration (Interface mode lines **I0-I1**) during the power up sequence of the chip.

Table 7. Host controller interface selection

	Interface Selection Pin	
	I0 (pin #16)	**I1** (pin #17)
HSU	0	0
I2C	1	0
SPI	0	1
RFU	*1*	*1*

Be careful, with SPI as a host interface, a specific hardware implementation is needed to use the LowVbat mode when PVDD is absent. See the application note [6].

6.1.1.1 SPI interface

Be careful, with SPI as a host interface, a specific hardware implementation is needed to use the LowVbat mode when PVDD is absent. See the application note [6].

Refer to the PN532 data sheet (see **Error! Reference source not found.**).

The PN532 is a SPI slave with the following pins used:

Table 8. Pin used for SPI interface

	PN532 Pin number
NSS	27
MOSI	28
MISO	29
SCK	30

The mode used for the clock is **Mode 0:**

Data is always sampled on the **first** clock edge of SCK

SCK is active **high.**

The data order used is LSB first.

Remark: The PN532 is waked up as long as NSS is low whatever the mode (Virtual Card or LowVbat). This feature can only be disabled by the PowerDown command. This pin should be tied to high when not used.

6.1.1.2 HSU interface

Refer to the PN532 data sheet (see **Error! Reference source not found.**).

HSU interface default configuration is:

Data bit	: 8 bits,
Parity bit	: none,
Stop bit	: 1 bit,
Baud rate	: 115 200 bauds,
Data order	: LSB first.

Table 9. Pin used for HSU interface

	PN532 Pin number
HSU_RX (NSS)	27
HSU_TX (MOSI)	28

6.1.1.3 I2C interface

Refer to the PN532 data sheet (see **Error! Reference source not found.**).

The PN532 is an I2C slave.

The PN532 is configured with I2C address 0x48 and is able to support a clock frequency up to 400 kHz.

The data order used is MSB first.

Table 10. Pin used for I2C interface

	PN532 Pin number
SCL (NSS)	27
SDA (MOSI)	28

6.1.2 P70_IRQ pin

In addition to the physical link used to communicate with the host controller, another dedicated IRQ line is used (see reference [**3**]) to inform the host controller when a response to a command is available.

This IRQ pin is driven automatically by the firmware. It is used by the handshake mechanism.

The behavior of this IRQ line is described in the *Handshake mechanism* chapter (§6.3, p: 48) and in the specific communications details for each link:

- HSU communication details §6.3.2, p:49,
- I2C communication details §6.3.3, p:54
- SPI communication details §6.3.4, p: 60.

6.2 Host controller communication protocol

6.2.1 Frames structure

Communication between the host controller and the PN532 is performed through frames, in a half-duplex mode.

Four different types of frames are used in one or both directions (host controller to the PN532 and PN532 to the host controller).

6.2.1.1 Normal information frame

Information frames are used to convey:

- Commands from the host controller to the PN532,
- And responses to these commands from the PN532 to the host controller.

The structure of this frame is the following:

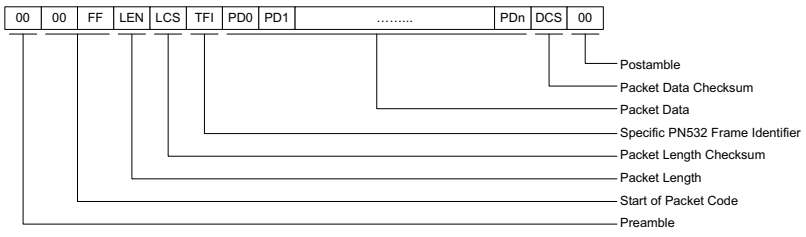


Fig 13. Normal information frame

- **PREAMBLE** 1 byte[4],
- **START CODE** 2 bytes (0x00 and 0xFF),
- **LEN** 1 byte indicating the number of bytes in the data field (TFI and PD0 to PDn),
- **LCS** 1 Packet Length Checksum LCS byte that satisfies the relation:
  Lower byte of [LEN + LCS] = 0x00,
- **TFI** 1 byte frame identifier, the value of this byte depends on the way of the message
  - **D4h** in case of a frame from the host controller to the PN532,
  - **D5h** in case of a frame from the PN532 to the host controller.
- **DATA** LEN-1 bytes of Packet Data Information
  The first byte PD0 is the Command Code,
- **DCS** 1 Data Checksum DCS byte that satisfies the relation:
  Lower byte of [TFI + PD0 + PD1 + … + PDn + DCS] = 0x00,
- **POSTAMBLE** 1 byte[2].

The amount of data that can be exchanged using this frame structure is limited to 255 bytes (including TFI).

[4] The preamble and postamble fields are represented here as byte whose value is 0x00. In the way from the host controller to the PN532, refer to each host link communication detailed paragraphs.

6.2.1.2 Extended information frame

The information frame has an extended definition allowing exchanging more data between the host controller and the PN532.
In the firmware implementation of the PN532, the maximum length of the packet data is limited to 264 bytes (265 bytes with TFI included).

The structure of this frame is the following:

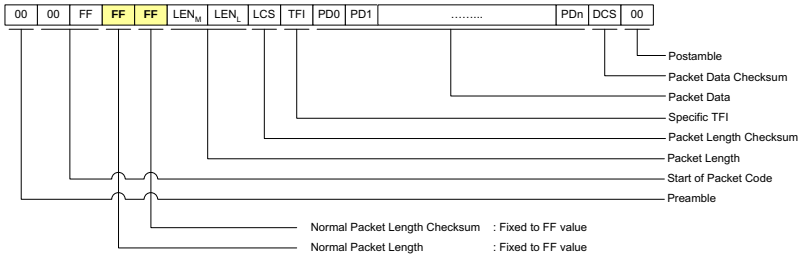


Fig 14. Extended Information frame

The normal **LEN** and **LCS** fields are fixed to the **0xFF** value, which is normally considered as erroneous frame, due to the fact that the checksum does not fit.

The real length is then coded in the two following bytes $\mathbf{LEN_M}$ (MSByte) and $\mathbf{LEN_L}$ (LSByte) with:

LENGTH = $\mathbf{LEN_M}$ x 256 + $\mathbf{LEN_L}$ coding the number of bytes in the data field (TFI and PD0 to PDn)

- **LCS** 1 Packet Length Checksum LCS byte that satisfies the relation: Lower byte of [$\mathbf{LEN_M}$ + $\mathbf{LEN_L}$ + **LCS**] = 0x00,
- **DATA** **LENGTH**-1 bytes of Packet Data Information
  The first byte PD0 is the Command Code.

The host controller, for sending frame whose length is less than 255 bytes, can also use this type of frame.
But, the PN532 always uses the suitable type of frame, depending on the length (Normal Information Frame for frame <= 255 bytes and Extended Information Frame for frame > 255 bytes).

6.2.1.3 ACK frame

The specific ACK frame is used for the synchronization of the packets and also for the abort mechanism.

This frame may be used either from the host controller to the PN532 or from the PN532 to the host controller to indicate that the previous frame has been successfully received.

ACK frame:

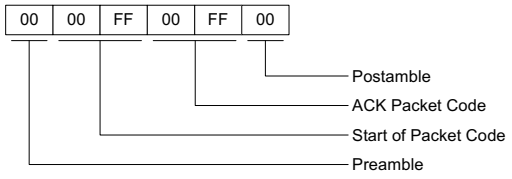


Fig 15. ACK frame

6.2.1.4 NACK frame

The specific NACK frame is used for the synchronization of the packets.

This frame is used only from the host controller to the PN532 to indicate that the previous response frame has not been successfully received, then asking for the retransmission of the last response frame from the PN532 to the host controller.

NACK frame:

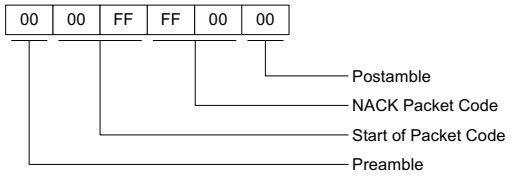


Fig 16. NACK frame

6.2.1.5 Error frame

The syntax error frame is used to inform the host controller that the PN532 has detected an error at the application level.

Error frame:

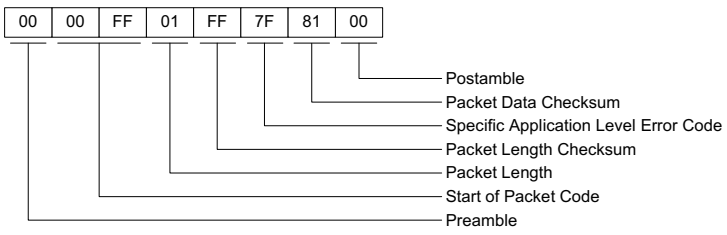


Fig 17. Error frame

6.2.1.6 Preamble and Postamble

These two specific fields of the frames are described in the previous paragraphs as single byte, which the value is 0x00.

In fact, these fields can be composed with an undetermined number of bytes:

o Preamble

The preamble field is composed of an undetermined number of bytes in which two consecutives bytes are not equal to 0x00 0xFF (otherwise specified in host controller link communication details, see §6.2.3, p:40, §6.2.4, p:42, §6.2.5, p:45).
The PN532 uses this synchronization pattern (0x00 0xFF) to detect the beginning of a frame; all the previous data are ignored.

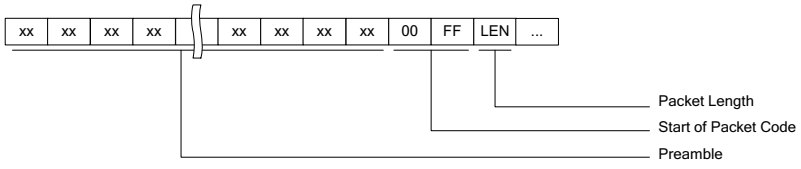


Fig 18. Preamble

o Postamble

The postamble field is composed of an undetermined number of bytes in which two consecutives bytes are not equal to 0x00 0xFF (otherwise specified in host controller link communication details, see §6.2.3, p:40, §6.2.4, p:42, §6.2.5, p:45).
The PN532 receives and analyses the frame until the DCS byte.
After this checksum byte, the common synchronization pattern detection starts again.

Thus, all the data comprised between the DCS byte and the next synchronization pattern (0x00 0xFF) is ignored.

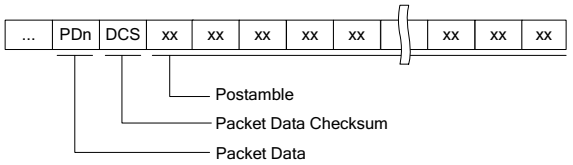


Fig 19. Postamble

o Frames sent by the PN532

Concerning the frames sent by the PN532 to the host controller, both the preamble and the postamble are constituted of only one 0x00 byte. However, it is possible for the PN532 not to send these two fields (preamble and postamble) to increase the overall data throughput (see §7.2.9, p:85)

o Examples

All the following frames are the same for the PN532's point of view (**`GetFirmwareVersion`**).

xx xx xx xx xx **00 FF 02 FE D4 02 2A** xx xx xx xx
00 FF 02 FE D4 02 2A xx xx xx xx
xx xx xx xx xx **00 FF 02 FE D4 02 2A**
00 FF 02 FE D4 02 2A

6.2.2 Dialog structure

The following chapters explain the dialog structure, whatever the physical link used.

The host controller is always the master of the complete exchange:

It sends a command to the PN532,

The PN532 sends back an acknowledge to inform the host controller that the command has been successfully received,

The PN532 executes the command,

The PN532 sends back the corresponding answer to the host controller,

Optionally, the host controller may send an ACK frame to indicate to the PN532 that the answer has been successfully received.

6.2.2.1 Data link level

a) Successful exchange at data link level

The figure below describes a normal exchange:

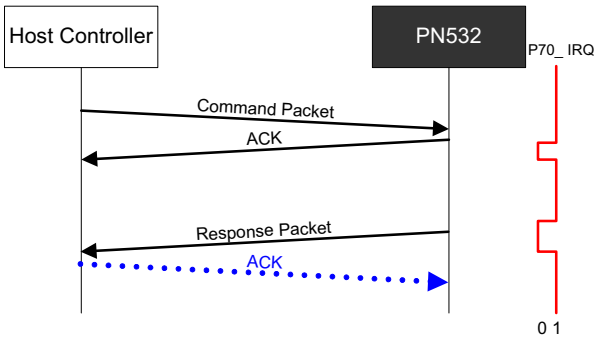


Fig 20. Data link level: normal exchange

b) Error at data link level, from host controller to PN532

When an error is detected by the PN532 at the data link level, it does not send back an ACK frame to the host controller.

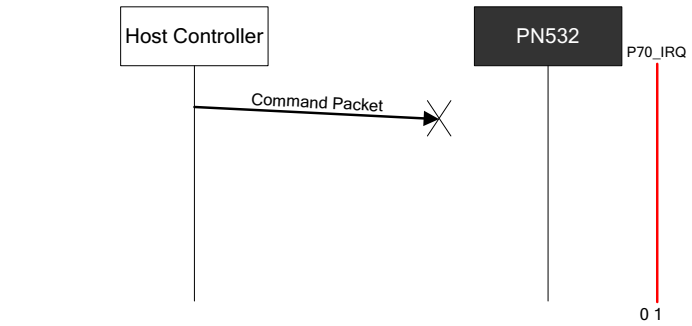


Fig 21. Data link level: error from the host controller to the PN532

The following errors are considered by the PN532 as data link level errors:

- LCS error,
- Framing error in case of HSU (stop bit is at logic level 0),
- DCS error,
- Timeout error in case of HSU
  The PN532 detects a timeout error if the complete frame is not received within a time interval corresponding to four times the duration of a 256-bytes length frame with the current baud rate used. The timeout detection starts after the reception of the LCS byte.

  Thus the timeout values for all the possible baud rates are:

Table 11. HSU timeout values

Baud Rate	1-byte duration (µs)	256-bytes duration (ms)	Timeout value (ms)
9 600	1 041,7	266,7	1 067
19 200	520,8	133,3	533
38 400	260,4	66,7	267
57 600	173,6	44,4	178
115 200	86,8	22,2	89
230 400	43,4	11,1	44
460 800	21,7	5,6	22
921 600	10,9	2,8	11
1 288 000	7,8	2,0	8

c) Error at data link level, from the PN532 to the host controller

When the host controller detects an error in the response packet (erroneous frame or no response), it uses a NACK frame to ask for the PN532 to send again the last response frame.

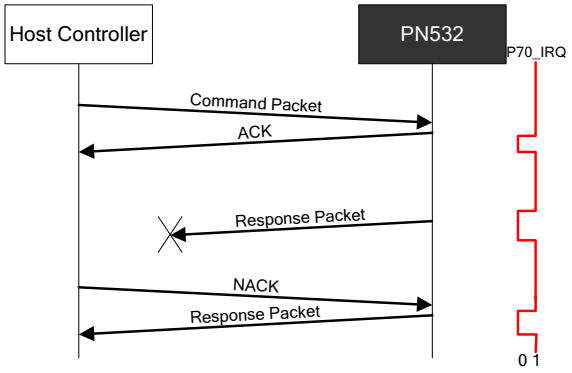


Fig 22. Data link level: error from the PN532 to the host controller

d) Abort

The host controller may send an ACK frame to force the PN532 to abort the current process.
In that case, the PN532 discontinues the last processing and does not answer anything to the host controller.

Then, the PN532 starts again waiting for a new command.

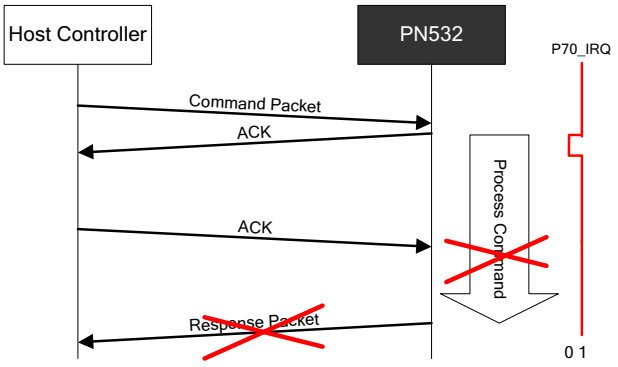


Fig 23. Data link level: error from the PN532 to the host controller

6.2.2.2 Application level

a) Successive exchanges

The host controller sends a new command after having received the answer of the previous one.

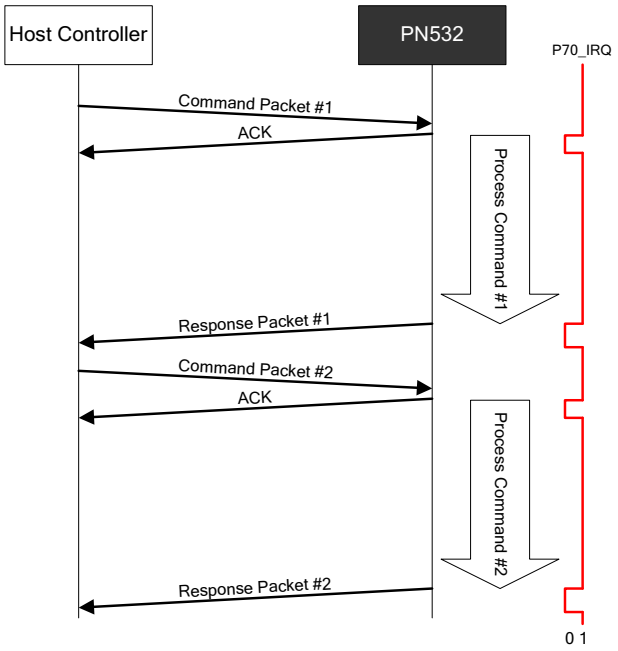


Fig 24. Application level: Successive exchanges

b) Abort

The host controller can force the PN532 to abort an ingoing process thanks to two different methods.

1. *Abort previous command with a ACK frame*

The host controller may send an ACK frame to force the PN532 to abort the current process.

In that case, the PN532 discontinues the last processing and does not answer anything to the host controller.

Then, the PN532 starts again waiting for a new command.

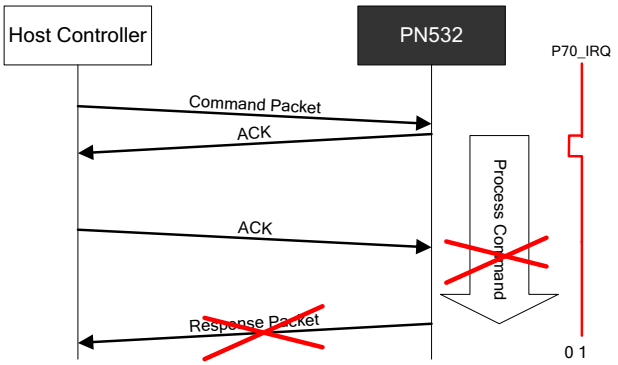


Fig 25. Data link level: Abort

2. *Abort previous command with a new command*

If the PN532 receives a command before having answered to the previous one, it stops the current process and start processing the new command received. It will send only the response to the last command.

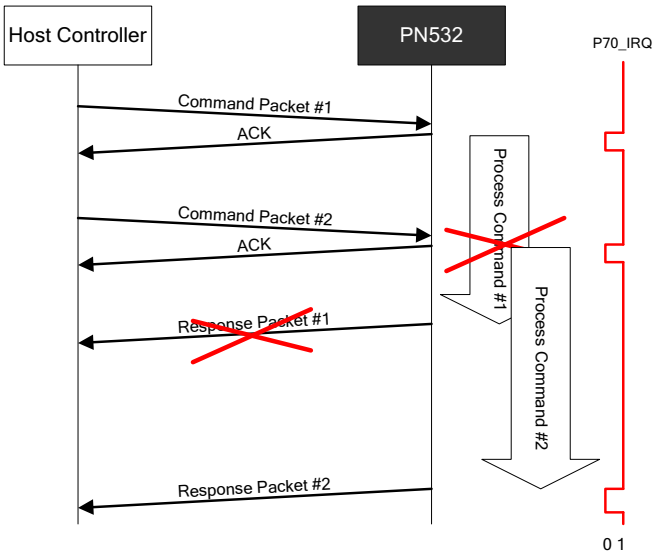


Fig 26. Application level: Abort a command and process a new one

c) Error at application level

When the PN532 detects an error at the application level, it sends back the specific "Syntax Error frame" to the host controller (see §6.2.1.5, p: 31).

An application level error may be due to one of the following reasons:

- Unknown **Command Code** sent by the host controller in the command frame,
- Unexpected frame length,
- Incorrect parameters in the command frame.

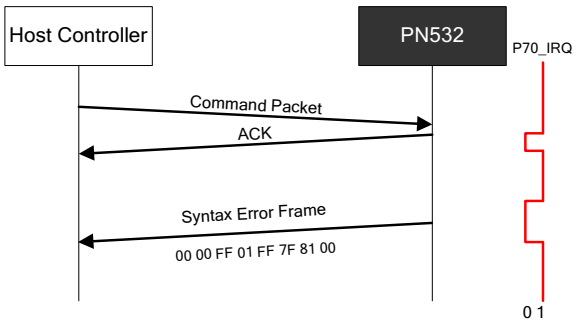


Fig 27. Application level: Error detected

6.2.3 HSU communication details

The HSU interface of the PN532 is a full duplex serial port capable of communicating with a host controller with a baud rate up to 1.288 Mbaud.

The PN532 receives the host controller command on its HSU_RX pin and transmits the response to the host controller on its HSU_TX pin.

The frames used when communicating with the HSU are exactly the same as defined in the previous paragraphs §6.2.1:

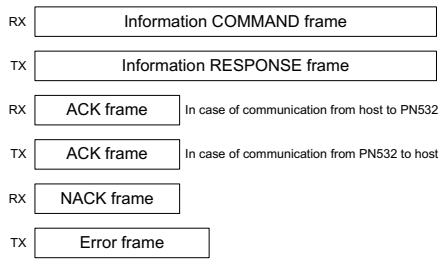


Fig 28. HSU link: frames

The figure below depicts the normal scheme of communication with the HSU:

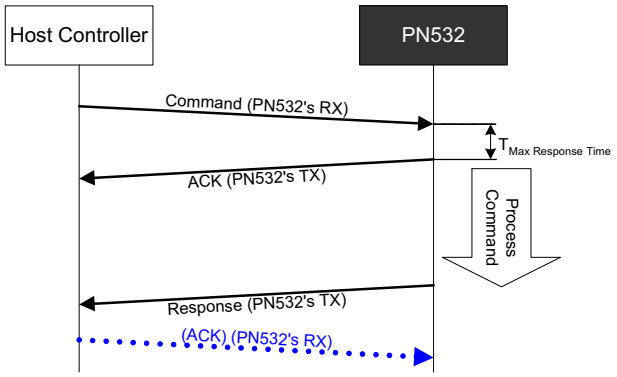


Fig 29. HSU link: general principle of communication

The PN532 has to respond to the incoming command frame within 15 ms ($T_{Max\ Response\ Time:}$ delay between the command frame and the ACK frame).

In the case the host controller does not detect an ACK frame within these 15 ms, the host controller should resend the same command frame.

Preamble and Postamble:

In the way from the host controller to the PN532, both the preamble and postamble fields may have a length different from one byte (0 to n) and the value has no impact on the frame processing.

6.2.4 I2C communication details

The I2C interface of the PN532 is compliant with the I2C bus specification.
The PN532 is configured as slave (address 0x48) and is able to communicate with a host controller in fast mode (up to 400 KHz CLK).

The frames used when communicating with the I2C are slightly modified compared to those defined in the previous paragraphs §6.2.1.

Some modifications are added due to the necessity of synchronization.

Indeed, as the PN532 is configured as pure I2C slave, and as the host controller has the possibility to discontinue the current process of the PN532 (as described in §6.2.2.1-d and 6.2.2.2-b), the PN532 has to be configured in a mode such as it must acknowledge the entire frame coming from the host controller (here acknowledge must be understood as the I2C meaning).
However, it may happen that the PN532 does not acknowledge its own address immediately after having finished a previous exchange.

When the PN532 has acknowledged its own address on a read command, a status byte is added at the beginning of the frame so that the host controller can know if the PN532 is ready to give an answer frame to the host controller.

7	6	5	4	3	2	1	0
rfu	rfu	rfu	rfu	rfu	rfu	rfu	RDY

- When bit **RDY = 0**, the PN532 has no frame available to be transferred to the host controller,
- When bit **RDY = 1**, the PN532 has a frame available to be transferred to the host controller.

The different frames are modified as follows:

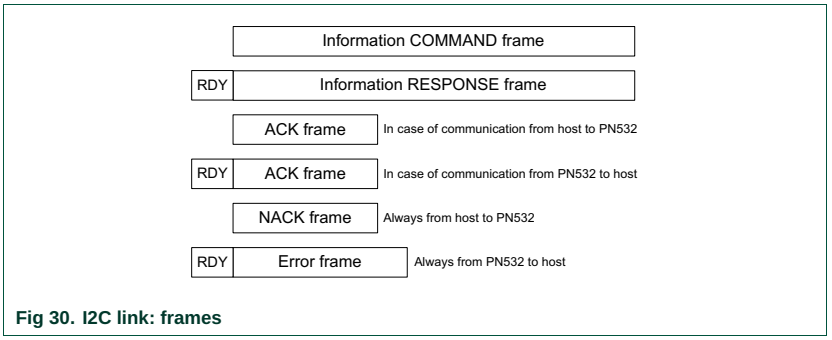


Fig 30. I2C link: frames

When the host controller wants to read data from the PN532, it has to read first the status byte and as long as the RDY bit is not equal to 1, it has to retry:

Each time a status byte is read with NOT READY information, before retrying the host controller must close the communication by sending an **I2C STOP** condition.

So, each frame should start with an **I2C START** condition to read the **status byte**.

If this byte indicates that the PN532 is not available, an **I2C STOP** condition should be generated by the bus controller. In fact, all the bytes read following a NOT READY status byte are not relevant.

If the PN532 indicates that it is ready, the rest of the frame shall be read before sending an **I2C STOP** condition. If an **I2C STOP** condition is sent before a complete frame is read, the remaining bytes are lost.

Remark: The PN532 I2C communication is combined with Handshake mechanism (see §6.2.4.2 p:44). Although Handshake mechanism is always used, the PN532 supports also the classic I2C communication (see §6.2.4.1 p: 43)

6.2.4.1 Classic I2C communication (without Handshake mechanism combination)

The following figure depicts this mechanism:

- COMMAND sent by the host controller,
- The host controller polls the Status byte (using its own frequency),
- ACK frame “sent” by the PN532,
- Polling of the Status byte by the host controller ,
- RESPONSE frame “sent” by the PN532,
- Optional ACK sent by the host controller.

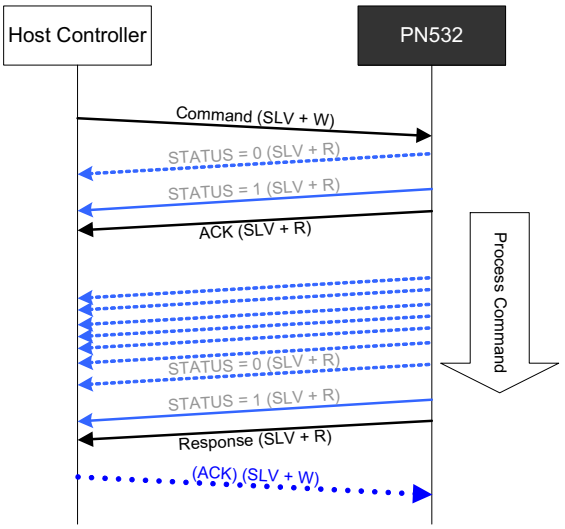


Fig 31. I2C link: general principle of communication

Legend used:

SLV + W: write operation at PN532 address (0x48),

SLV + R: read operation at PN532 address (0x49).

6.2.4.2 Advanced I2C communication (with Handshake mechanism combination)

A better way for the host controller is to use the P70_IRQ pin that indicates when the PN532 is ready to send its frame .

In that case, the host controller can wait for this line to be asserted by the PN532 before to read the status byte. As a consequence, the overall traffic on the I2C bus is reduced.

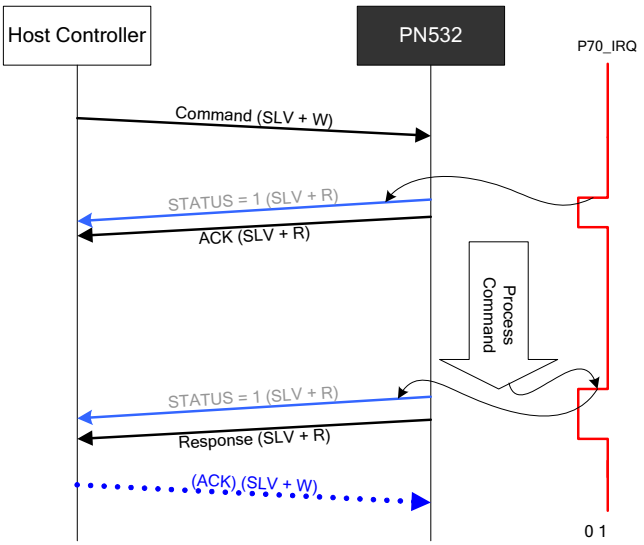


Fig 32. I2C link: using P70_IRQ pin

Preamble and Postamble:

In the way from the host controller to the PN532, both the preamble and postamble fields may have a length different from one byte (0 to n) and the value has no impact on the frame processing.

6.2.5 SPI communication details

The SPI interface of the PN532 is compliant with the SPI bus specification.

The PN532 is configured as slave and is able to communicate with a host controller with a clock (SCK) up to 5MHz.

The SPI interface includes a specific register allowing the host controller to know if the PN532 is ready to receive or to send data back.

7	6	5	4	3	2	1	0
rfu	rfu	rfu	rfu	nu	nu	nu	RDY

- When bit **RDY = 0**, the PN532 has no frame available to be transferred to the host controller,
- When bit **RDY = 1**, the PN532 has a frame available to be transferred to the host controller.

As a consequence, the frames used when communicating with the SPI are slightly modified compared to those defined in the paragraph §6.2.1.

Before initiating an exchange (either from the host controller to the PN532 or from the PN532 to the host controller), the host controller must write a byte indicating to the PN532 what the following operation is:

- **First byte = xxxx xx10b**, status reading (PN532 to host controller),
- **First byte = xxxx xx01b**, data writing (host controller to PN532),
- **First byte = xxxx xx11b**, data reading (PN532 to host controller).

The different frames are modified as follows:

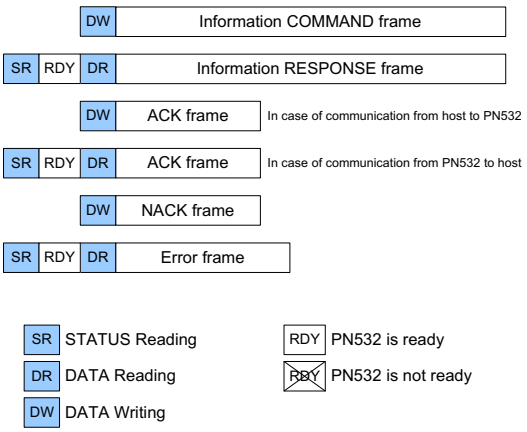


Fig 33. SPI: frames

When the host controller wants to read data from the PN532, it has to read first the status byte and as long as the RDY bit is not equal to 1, it has to retry:

Remark: The PN532 SPI communication is combined with Handshake mechanism (see §6.3.4 p: 60). Although Handshake mechanism is always used, the PN532 supports also the classic SPI communication (see §6.2.5.1 p: 46)

6.2.5.1 Classic SPI communication (without Handshake mechanism combination)

The following figure depicts this mechanism:

- COMMAND sent by the host controller,
- The host controller polls the Status byte using its own frequency,
- ACK frame “sent” by the PN532,
- Polling of the Status byte by the host controller,
- RESPONSE frame “sent” by the PN532,
- Optional ACK sent by the host controller.

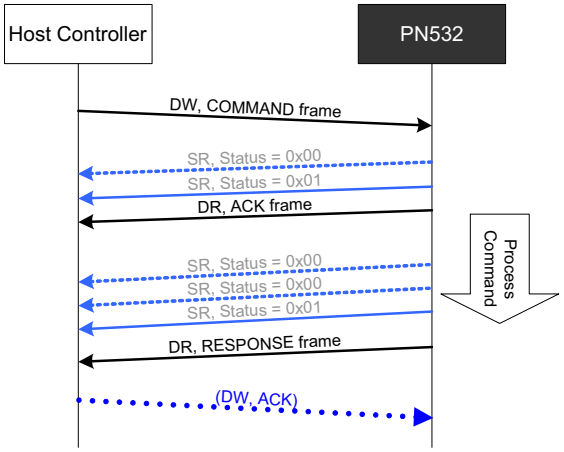


Fig 34. SPI: general principle of communication

6.2.5.2 Advanced SPI communication (with Handshake mechanism combination)

A better way for the host controller is to use the P70_IRQ pin that indicates when the PN532 is ready to send its frame.

In that case, the host controller can wait for this line to be asserted by the PN532 and has no more need to read the status byte. As a consequence, the overall traffic on the SPI bus is reduced.

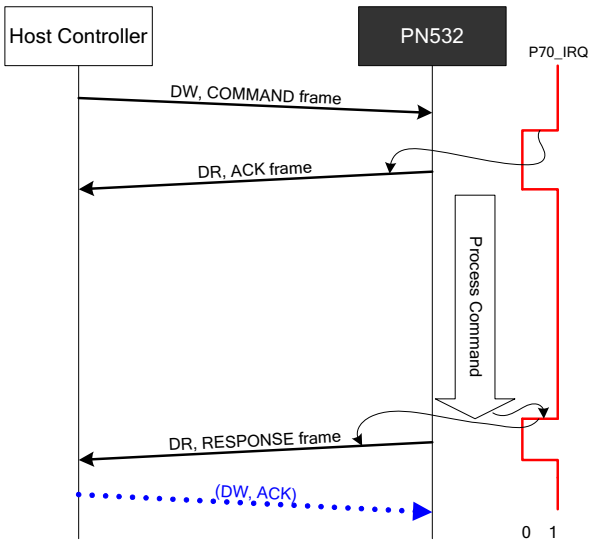


Fig 35. SPI link: using P70_IRQ pin

Preamble and Postamble:

In the way from the host controller to the PN532, both the preamble and postamble fields have to be composed of a single byte with value of 0x00.

6.3 Handshake mechanism

6.3.1 General presentation

The goals of the handshake mechanism are:

- For the host controller, to wake the PN532 up (if it was asleep),
- For the PN532, to wake the host controller up (if it was asleep),
- For the host controller, to reduce the traffic on I2C bus or SPI bus when waiting for the response frame,
- To warn the host controller when an event occurred at SAM side (in Virtual Card mode only).

This mechanism is particularly interesting when used in a system where both the PN532 and the host controller are frequently in Power Down mode.

The two handshake signals are:

- In the way from the PN532 to the host controller ➔ **P70_IRQ**,
- In the way from the host controller to the PN532 ➔ **H_REQ**.
  This one is optional, since the PN532 offers the possibility to be waked up directly by receiving data on the operational link (HSU, I2C or SPI).
  When it is used, this line helps the host controller not to send data that could be lost. Indeed, the PN532 needs some time to be able to receive properly a data frame when leaving Power Down mode, and the H_REQ line forces the PN532 to wake up.

Remark: Except in LowVbat mode, the IRQ functionality is always present within the PN532, but as described in the chapters §6.2.3 p:40, §6.2.4.1 p:43 and §6.2.5.2 p:47, the communication can be managed by the host without looking at P70_IRQ signal.

The handshake mechanism is highly coupled with Power Down mode. There are different possibilities for the PN532 to be in Power Down mode:

- Use the **`PowerDown`** command (§7.2.11, p:98),
- Use the **`TgInitAsTarget`** command (§7.3.14, p: 151). After having received this command and sent the ACK frame, the PN532 goes automatically into Power Down mode if no external RF field is detected,
- When in LowVbat or Card Emulation modes after having received the corresponding **`SAMConfiguration`** command (§7.2.10, p: 89). After having sent the ACK frame, the PN532 goes automatically into Power Down mode if no external RF field is detected,
- When the bCPUPowerMode variable is set to Power Down mode by host controller (see §3.1.3.2, p:12).

There are some differences depending on the link used, HSU, SPI or I2C; the following paragraphs detail the way of using the handshake for each link.

6.3.2 Handshake mechanism in case of HSU link

The handshake/wake-up mechanism is based on a four lines interface between the host controller processor and the PN532:

- **T_R_X** : Serial reception line of the PN532,
- **T_T_X** : Serial transmission line of the PN532,
- **H_REQ** : Request or acknowledge line from the host controller (connected to the P32_INT0 line of the PN532). This line is optional,
- **P70_IRQ** : PN532 request line (IRQ for the host controller).

6.3.2.1 Case of LowVbat

In LowVbat mode (configured with the **`SAMConfiguration`** command (§7.2.10 p: 89)), the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

In this mode, in comparison with Virtual card mode, the host is not informed about SAM activities, the handshake mechanism cannot be used.

The PN532 needs a particular sequence to get out of this mode (See Initialization Sequence §3.1.3.8 p: 19). All commands sent during this mode need a Long preamble (see Fig 65 p: 99)

6.3.2.2 Normal case

When neither the PN532 nor the host controller is in Power Down mode, the host controller does not need to assert the H_REQ line. However, the host controller can start an exchange with asserting the H_REQ signal as mentioned in the Fig 36.

In both cases (H_REQ asserted or not), the PN532 reacts in the same way, as described below.

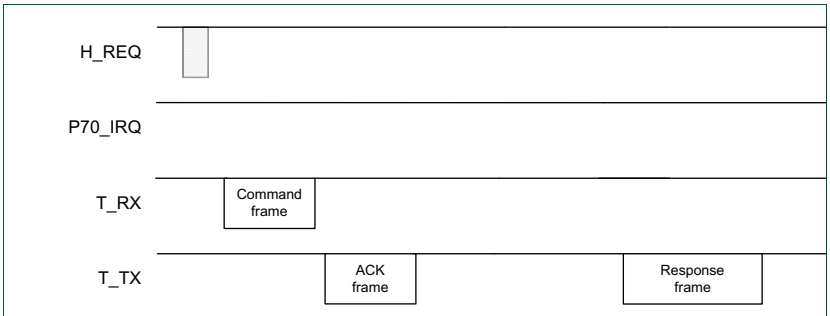


Fig 36. Handshake in case of HSU link – case 1

6.3.2.3 Case of PN532 in Power Down mode

The PN532 is in Power Down mode and the host controller wants to send a new command.

Depending on the enabled wake up sources, the host controller can wake it up either (see **`PowerDown`** command, §7.2.11, p: 98):

- By generating a pulse on the H_REQ line (minimum duration: T_osc_start) or
- By sending a command on T_Rx line with Long preamble (see Fig 65 p: 99) (in order that this command is not lost from the PN532 point of view, the host controller must respect some rules that are detailed in the §7.2.11, p: 98).

In both cases, the PN532 does not inform that it is awaken, and then the host controller has to wait for at least T_osc_start before sending a new command that will be properly understood.

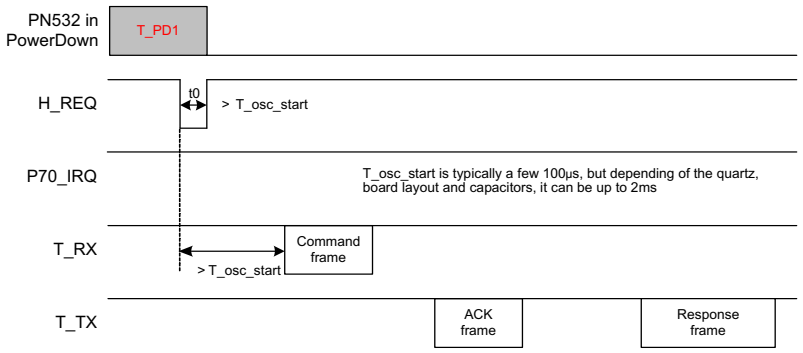


Fig 37. Handshake in case of HSU link – case 2

6.3.2.4 Case of the TgInitAsTarget command

After having received the **`TgInitAsTarget`** (§7.3.14, p: 151) command from the host controller and if no external RF field is detected, the PN532 goes into Power Down mode. It then will be waken up by an external RF field, or by a new command from the host controller (see §6.3.2.3).

Once the PN532 has been activated as target, it informs the host controller that it has been activated by an external initiator with the P70_IRQ pin.

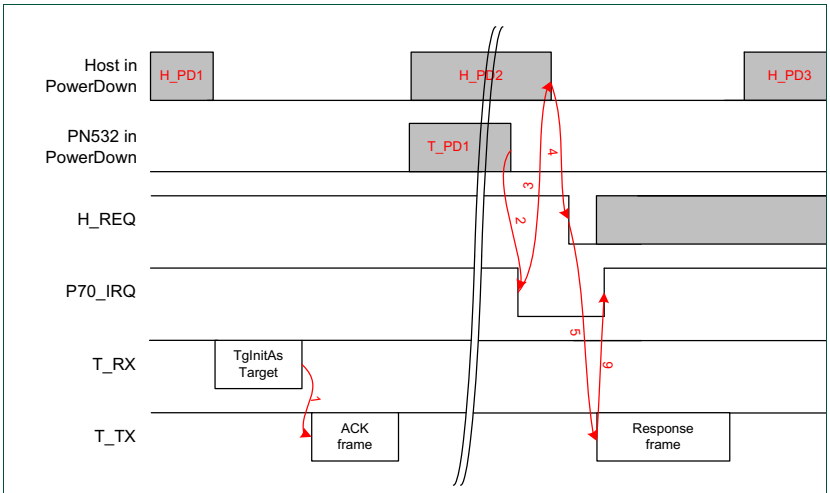


Fig 38. Handshake in case of HSU link – case 3

- The host controller sends the command and the PN532 acknowledges the command frame (**1**) and goes in power down (**T_PD1**),
- Then the host controller may go to power down (**H_PD2**),
  The PN532 is woken up by the detection of an external RF field,
- As soon as the PN532 has been activated, it is ready to send back the response to the host controller and it pulls P70_IRQ pin to low (**2**). If the host controller was asleep, it will wake it up (**3**).
  Then the host controller asserts its H_REQ line to acknowledge (**4**),
- After having detected a falling edge on H_REQ, the PN532 sends the response frame to the host controller (**5**) and releases the P70_IRQ pin to high (**6**),
- At the end of the exchange, the host controller may return into Power Down mode (**H_PD3**).

If the H_REQ line is not available (handshake with P70_IRQ pin only), the sending of the response frame by the PN532 is a bit changed.

The P70_IRQ pulse has a maximum duration of 12 ms (MaxWakupHost); if no H_REQ falling edge is detected during this 12 ms, the PN532 sends its response frame, supposing that the host controller has been awaken.

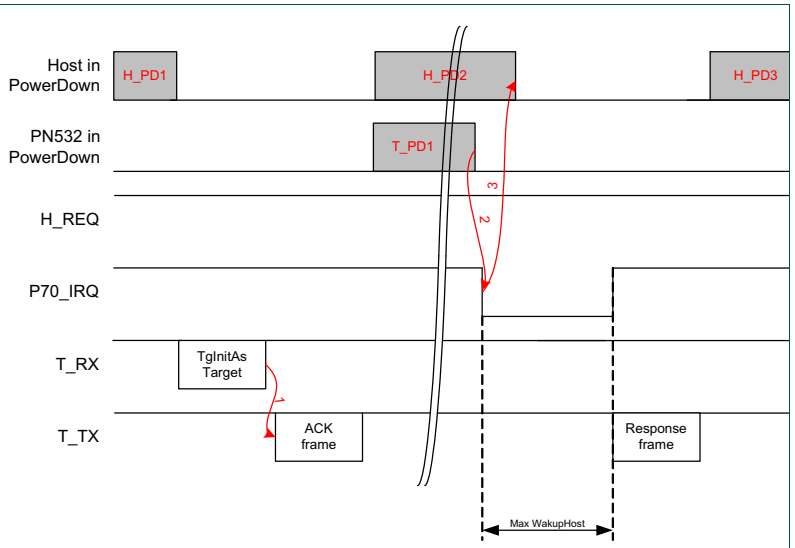


Fig 39. Handshake in case of HSU link – case 3 without H_REQ

6.3.2.5 Case of SAMConfiguration – Virtual Card

In Virtual Card mode (configured with the **`SAMConfiguration`** command (§7.2.10 p: 89)) or after Reset, the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

The handshake mechanism is then used to warn the host controller that something happened at SAM side.

The following description applies:

- As soon as the PN532 asserts the P70_IRQ pin (**1**), the host controller is waken up (**2**) (if it was asleep **H_PD1**),
- Due to the fact that this P70_IRQ negative pulse occurs outside a standard exchange initiated by the host controller, the host controller shall send a **`GetGeneralStatus`** command which releases the P70_IRQ pin to high (**4**). The H_REQ is used or not (**2** and **3**).

Example:

SAMConfiguration (Virtual Card, no timeout) ➔ D4 14 02 00

Then, the host controller waits for P70_IRQ being asserted by the PN532 indicating that something happened with the SAM.

GetGeneralStatus () ➔ D4 04

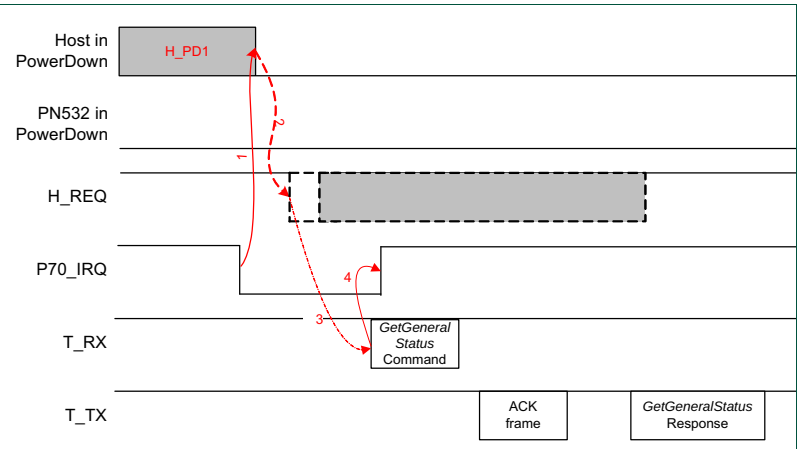


Fig 40. Handshake in case of HSU link – case 4

Remarks:

- **The combination of the SAMConfiguration Virtual mode command with the PowerDown command is not possible.**
- **In Virtual Card mode (14 02 00 command), H_REQ line can be used.**

The PN532 needs approximately 1 ms to get into Power Down mode, after the command response. Sending host commands during this time is not recommended

6.3.3 Handshake mechanism in case of I2C link

The handshake/wake-up mechanism is based on a four lines interface between the host controller processor and the PN532:

- **SDA** : I2C data line,
- **SCL** : I2C clock line,
- **H_REQ** : Request or acknowledge line from the host controller (connected to the PN532 P32_INT0 line). This line is optional,
- **P70_IRQ** : PN532 request line (IRQ for the host controller).

6.3.3.1 Case of LowVbat

In LowVbat mode (configured with the **`SAMConfiguration`** command (§7.2.10 p:89)) or after Reset (or power-up), the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

Although handshake mechanism is always used, H_REQ has no influence on PN532 and P70_IRQ is never managed except with host communications.

In this mode, in comparison with Virtual card mode, the host is not informed about SAM activities.

The PN532 needs a particular sequence to get out of this mode (See §3.1.3.8 p:19). All commands sent during this mode need a clock (SCL) stretch (see Fig 44 p:57)

6.3.3.2 Normal case

When neither the PN532 nor the host controller is in Power Down mode, the host controller does not need to assert the H_REQ line.

The PN532 generates a P70_IRQ signal to inform that the answer is ready (both ACK frame and answer response frame).

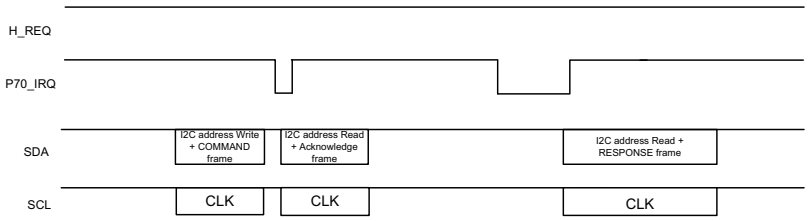


Fig 41. Handshake in case of I2C link – case 1

Remark:

However, if the host controller starts an exchange asserting the H_REQ signal, the PN532 will react as described below.

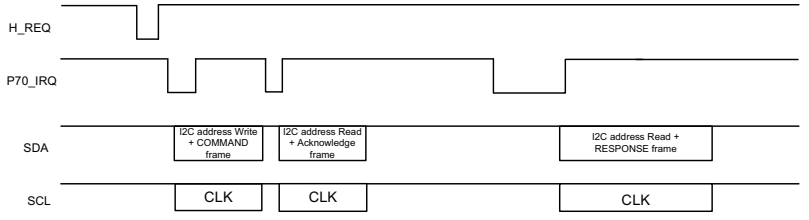


Fig 42. Handshake in case of I2C link – case 1bis

6.3.3.3 Case of PN532 in Power Down mode

The PN532 is in Power Down mode and the host controller wants to send a new command.

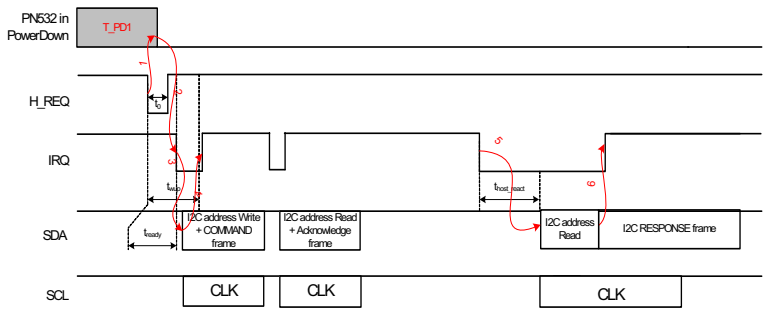


Fig 43. Handshake in case of I2C link – case 2

Remark: t_{host_react} is not managed by the PN532.

- The PN532 is in Power Down mode (**T_PD1**),
  The host controller wants to send a frame to the PN532, and so it asserts the H_REQ line. That makes the PN532 waking up (**1**) and it acknowledges with P70_IRQ pin (**2**),
- Then before sending the command frame (**3**), the host controller has 2 possibilities:
  - either the host controller waits for a defined delay (t_{wup}) which guarantees in that case that the PN532 will be waken up when the I2C frame will be sent,
  - or the host controller monitors the P70_IRQ pin to check when the PN532 is woken up (with a maximum timeout of t_{wup}).
    This option can optimize the overall waiting time if the PN532 was not asleep; there is no need to wait the maximum wakeup time ($t_{ready} < t_{wup}$).
- The PN532 sets back P70_IRQ pin after having recognized its I2C slave address (**4**),
- The PN532 acknowledges the command frame (ACK frame),
- As soon as the PN532 is ready to send back the response to the host, it asserts its IRQ line. Once ready (t_{host_react}), the host controller gets back the answer frame from the PN532 (**5**).
- The detection of its own I2C address makes the PN532 releasing the IRQ line (**6**).

Timings:

- t_0: duration of the H_REQ low pulse. The minimum value depends on the internal state of the PN532 (awake or asleep):
  - PN532 awake: $t_0 \geq 1\mu s$
  - PN532 asleep: $t_0 \geq$ **T_osc_start** : typically a few 100µs, but depending of the quartz, board layout and capacitors it can be up to 2ms.
- t_{ready}: delay between the falling edge of the H_REQ pulse and the falling edge of the P70_IRQ pin represents the minimum delay the PN532 needs to be ready. This delay depends on the internal state of the PN532, the CPU frequency and the Quartz. Typical values are:
  - PN532 awake: $t_{ready} \leq 10\mu s$
  - PN532 asleep: t_{ready} = **T_osc_start + 10µs**

Variant: In the case where the H_REQ signal is not used, the PN532 will stretch the SCL line after having recognized its own slave address.

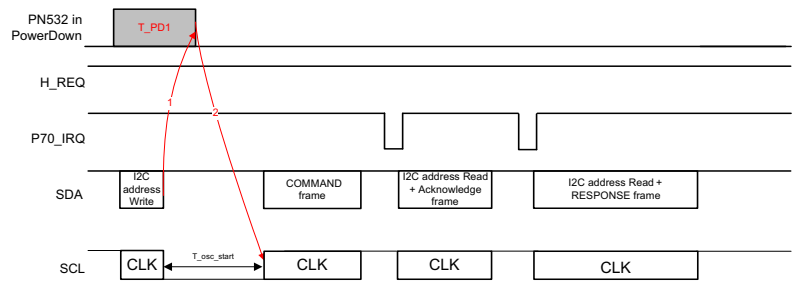


Fig 44. Handshake in case of I2C link – case 2 without H_REQ

Remark: The PN532 needs approximately **1 ms** to get into Power Down mode, after the command response. Sending host commands during this time is not recommended

6.3.3.4 Case of the TgInitAsTarget command

After having received the **`TgInitAsTarget`** command (see §7.3.14, p:151) from the host controller and if no external RF field is detected, the PN532 goes into Power Down mode. It then will be waken up by an external RF field, or by a new command from the host controller (see §6.3.3.3).

Once the PN532 has been activated as target, it informs the host controller with the P70_IRQ pin.

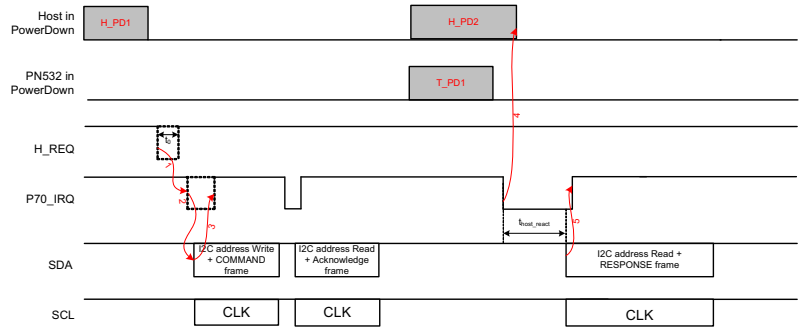


Fig 45. Handshake in case of I2C link – case 3

Remark: t_{host_react} is not managed by the PN532.

- The host controller is possibly in Power Down mode (**H_PD1**),
  The host controller wants to send a frame to the PN532, so it asserts the H_REQ line (**1**, optional),
- The host controller monitors the P70_IRQ pin to check when the PN532 is ready (**2**, optional, depending if **1** has been done or not),
- The PN532 sets back P70_IRQ pin after having recognized its I2C slave address (**3**, optional, depending if **1** has been done or not),
- The PN532 acknowledges the command frame (ACK frame) and goes in Power Down mode (**T_PD1**),
- Then the host controller may also go to power down (**H_PD2**),
- As soon as the PN532 has been activated, it is ready to send back the response to the host controller and it asserts its P70_IRQ pin. If the host controller was asleep, it will wake it up (**4**),
- Once ready (t_{host_react}), the host controller gets back the answer frame from the PN532. The detection of its own I2C address makes the PN532 releasing the P70_IRQ pin (**5**).

Timings:

t_0: duration of the H_REQ low pulse. The minimum value depends on the internal state of the PN532 (awake or asleep):

- PN532 awake: **$t_0 \geq 1\mu s$**
- PN532 asleep: **$t_0 \geq$ T_osc_start:** typically a few 100µs, but depending of the quartz, board layout and capacitors it can be up to 2ms.

6.3.3.5 Case of SAMConfiguration – Virtual Card

In Virtual Card mode (configured with the **`SAMConfiguration`** command (§7.2.10 p: 89)) or after Reset (or power up), the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

The handshake mechanism is then used to warn the host controller that something happened at SAM side.

The following description applies:

- As soon as the PN532 asserts the P70_IRQ pin to inform the host controller that something occurred at the SAM side (**1**), the host controller is waken up (if it was asleep **H_PD1**),
- Due to the fact that this P70_IRQ negative pulse occurs outside a standard exchange initiated by the host controller, the host controller shall send a **`GetGeneralStatus`** command which releases the P70_IRQ pin to high.

Then the rest of the sequence is the same as described in the Fig 41.

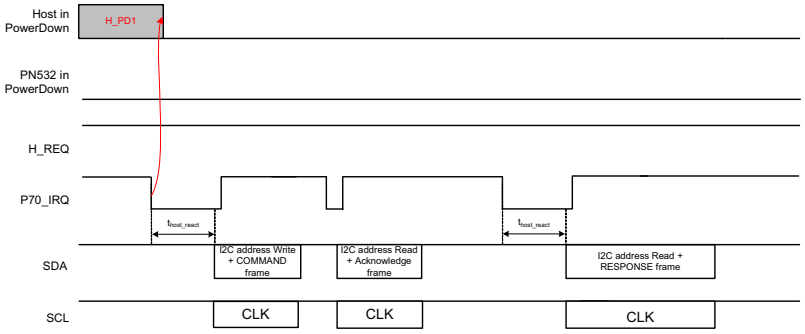


Fig 46. Handshake in case of I2C link – case 4

Remark: t_{host_react} is not managed by the PN532.

Example:

SAMConfiguration (Virtual Card, no timeout) ➔ D4 14 02 00

Then, the host controller waits for P70_IRQ being asserted by the PN532 indicating that something happened with the SAM.

GetGeneralStatus () ➔ D4 04

Remarks :

- **The combination of the SAMConfiguration Virtual mode command with the PowerDown command is not possible.**

In Virtual Card mode (14 02 00 command), H_REQ line can be used.

The PN532 needs approximately 1 ms to get into Power Down mode, after the command response. Sending host commands during this time is not recommended.

6.3.4 Handshake mechanism in case of SPI link

The handshake/wake-up mechanism is based on a six lines interface between the host controller processor and the PN532:

- **NSS** : Slave Selection,
- **SCK** : Clock line,
- **MISO** : Master Input Slave Output,
- **MOSI** : Master Output Slave Input,
- **P70_IRQ**: PN532 request line (IRQ for the host controller),
- **H_REQ** : Request or acknowledge line from the host controller

  (This is connected to the PN532 P32_INT0 line). This line is optional.

6.3.4.1 Case of LowVbat

Be careful, with SPI as a host interface, a specific hardware implementation is needed to use the LowVbat mode when PVDD is absent. See the application note [6].

In LowVbat mode (configured with the **`SAMConfiguration`** command (§7.2.10 p: 89)) or after Reset (or power-up), the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

Although handshake mechanism is always used, H_REQ has no influence on PN532 and P70_IRQ is never managed except with host communications.

In this mode, in comparison with Virtual card mode, the host is not informed about SAM activities.

The PN532 needs a particular sequence to get out of this mode (See §3.1.3.8 p: 19). Before sending any command in this mode, NSS should be maintained to low during at least T_osc_start (see Fig 49 p: 62)

6.3.4.2 Normal case

When neither the PN532 nor the host controller is in power-down mode, the host controller does not to assert the H_REQ line.

The PN532 generates a P70_IRQ signal to inform that the answer is ready.

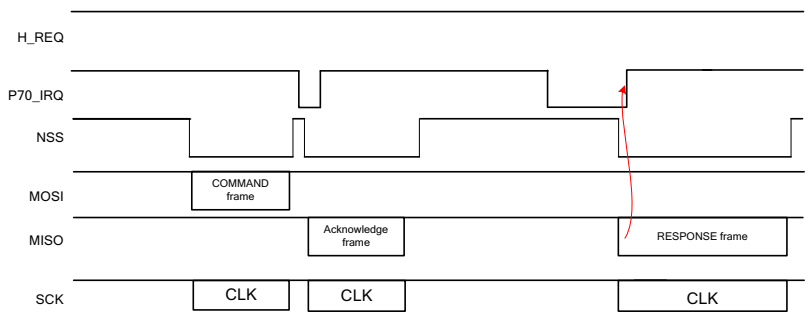


Fig 47. Handshake in case of SPI link – case 1

6.3.4.3 Case of PN532 in Power Down mode

The PN532 is in Power Down mode and the host controller wants to send a new command

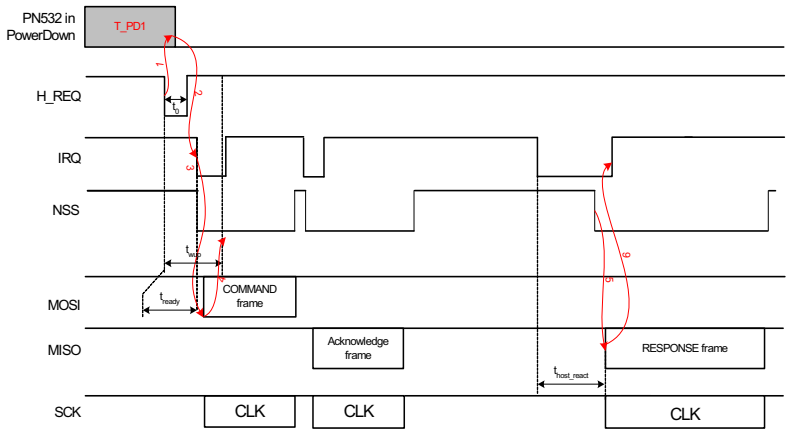


Fig 48. Handshake in case of SPI link – case 2 with H_REQ

Remark: t_{host_react} is not managed by the PN532.

- The PN532 is in Power Down mode (**T_PD1**).
  The host controller wants to send a frame to the PN532, so it asserts the H_REQ line. That makes the PN532 waking up (**1**) and immediately it acknowledges with P70_IRQ pin (**2**),
- Then before sending the command frame (**3**), the host controller has 2 possibilities:
  - o either the host controller waits for a defined delay (t_{wup}) which guarantees in that case that the PN532 will be waken up when the SPI frame will be sent,
  - o or the host controller monitors the P70_IRQ pin to check when the PN532 is waked up (with a maximum timeout of t_{wup}).
    This option can optimize the overall waiting time if the PN532 was not asleep; there is no need to wait the maximum wakeup time ($t_{ready} < t_{wup}$).
- The PN532 sets back P70_IRQ pin after having received a SPI DW (**4**) (see §6.2.5, p:45),
- The PN532 acknowledges the command frame (ACK frame),
- As soon as the PN532 is ready to send back the response to the host controller, it asserts its P70_IRQ pin,
- Once ready (t_{host_react}), the host controller gets back the answer frame from the PN532 after triggering NSS line to low level (**5**).

The detection of a SPI DW makes the PN532 releasing the IRQ line (**6**).

Variant: In the case in where the H_REQ signal is not used, the host controller will have to set NSS to low during a certain amount of time before sending the command frame.

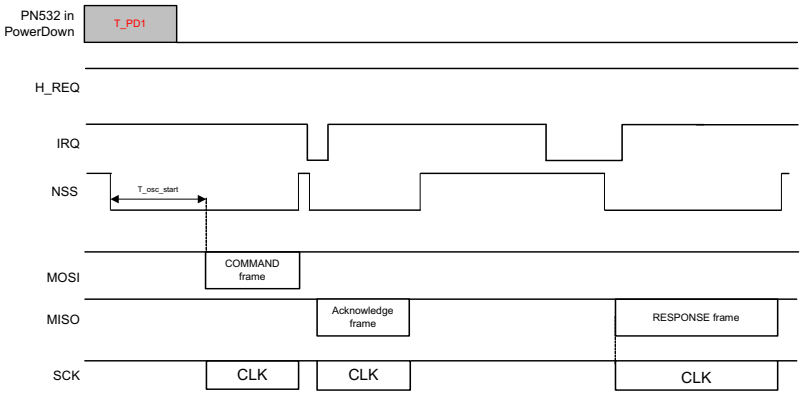


Fig 49. Handshake in case of SPI link – case 2 without H_REQ

Remark: The PN532 needs approximately **1 ms** to get into Power Down mode, after the command response. Sending host commands during this time is not recommended

6.3.4.4 Case of the TgInitAsTarget Command

The PN532 goes eventually into Power Down mode (if no external RF field is present) after having received the **`TgInitAsTarget`** command (§7.3.14, p:151) from the host controller, and will be waken up by an external RF field, or by a new command from the host controller (see §6.3.4.3). Once the PN532 has been activated as target, it informs the host controller with the P70_IRQ pin.

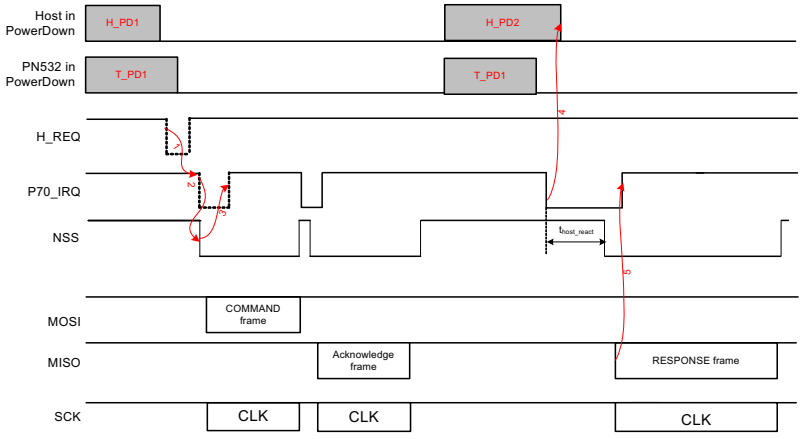


Fig 50. Handshake in case of SPI link – case 3

Remark: t_{host_react} is not managed by the PN532.

- The host controller is possibly in Power Down mode (**H_PD1**).
  The host controller wants to send a frame to the PN532, so it asserts the H_REQ line (**1**, optional),
- The host controller monitors the P70_IRQ pin to check when the PN532 is ready (**2**, optional, depending if **1** has been done or not),
- The PN532 sets back P70_IRQ pin after having received a SPI DW (see §6.2.5, p:45) (**3**, optional, depending if **1** has been done or not),
- The PN532 acknowledges the command frame (ACK frame) and goes in Power Down mode (**T_PD1**),
- Then the host controller may also go to power down (**H_PD2**),
- As soon as the PN532 has been activated, it is ready to send back the response to the host controller and it asserts its P70_IRQ pin. If the host controller was asleep, it will wake it up (**4**),
- Once ready (t_{host_react}), the host controller gets back the answer frame from the PN532. The detection of a SPI DR makes the PN532 releasing the P70_IRQ pin (**5**).

6.3.4.5 Case of SAMConfiguration – Virtual Card

In Virtual Card mode (configured with the **`SAMConfiguration`** command (§7.2.10 p:89)) or after Reset, the PN532 is in Power Down mode when waiting for the SAM to be activated by an external R/W.

The handshake mechanism is then used to warn the host controller that something happened at SAM side.

The following description applies:

- As soon as the PN532 asserts the P70_IRQ pin to inform the host controller that something occurred at the SAM side (**1**), the host controller is waken up if it was asleep (**H_PD1**),
- Due to the fact that this P70_IRQ negative pulse occurs outside a standard exchange initiated by the host controller, the host controller shall send a **`GetGeneralStatus`** command which releases the P70_IRQ pin to high.

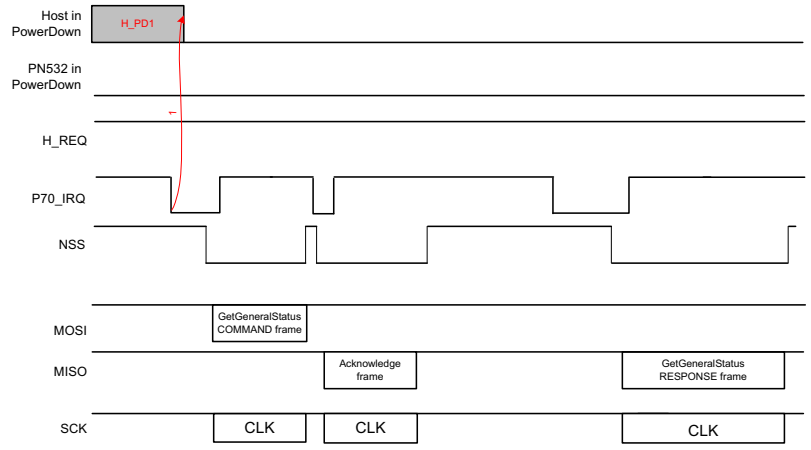


Fig 51. Handshake in case of SPI link – case 4

Remarks:

- **The combination of the SAMConfiguration Virtual mode command with the PowerDown command is not possible.**
- **In Virtual Card mode (14 02 00 command), H_REQ line can be used.**

The PN532 needs approximately 1 ms to get into Power Down mode, after the command response. Sending host commands during this time is not recommended

7. Commands supported

The following description of the commands details:

- The frame structure[5] including the type and amount of data:
  - That the host controller has to deliver to the PN532 (***Input***),
  - That the PN532 returns to the host controller (***Output***).
- When existing, the possible causes of syntax error (***Syntax Error Conditions***),
- A description of the process attached to the command (***Description***).

For Input and Output data, optional bytes are written into square brackets (**[ ]**).

List of commands:

A cross (X) in the PN532 column indicates if the command may be used with the PN532 configured as initiator or/and with the PN532 configured as target.

The “Command Code” column gives the value of the command code (CC in the two represented frames below) that is used in the frame from the host controller to the PN532.

00	00	FF	LEN	LCS	D4	CC	Optional Input Data	DCS	00

00	00	FF	LEN	LCS	D5	CC+1	Optional Output Data	DCS	00

For the “RF Communication” commands, when commands are dedicated to the PN532 as initiator (respectively Target) a **In** prefix has been added (respectively **Tg** prefix)

Table 12. Command set

Command	PN532 as Initiator	PN532 as Target	Command Code	Page
Miscellaneous				
Diagnose	X	X	0x00	69
GetFirmwareVersion	X	X	0x02	73
GetGeneralStatus	X	X	0x04	74
ReadRegister	X	X	0x06	76
WriteRegister	X	X	0x08	78
ReadGPIO	X	X	0x0C	79
WriteGPIO	X	X	0x0E	81
SetSerialBaudRate	X	X	0x10	83
SetParameters	X	X	0x12	85
SAMConfiguration	X	X	0x14	89
PowerDown	X	X	0x16	98

[5] The frame representation does not include the complete frame, but only the following field:
- TFI,
- Command Code,
- When needed the input or output data.

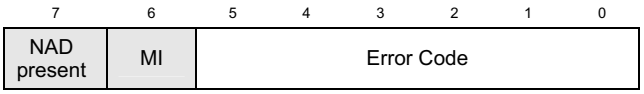
Thus, the Preamble, Start of Packet, LEN, LCS, DCS and Postamble are omitted in this description.

Command	PN532 as Initiator	PN532 as Target	Command Code	Page
RF communication				
RFConfiguration	X	X	0x32	101
RFRegulationTest	X	X	0x58	107
Initiator				
InJumpForDEP	X		0x56	108
InJumpForPSL	X		0x46	113
InListPassiveTarget	X		0x4A	115
InATR	X		0x50	122
InPSL	X		0x4E	125
InDataExchange	X		0x40	127
InCommunicateThru	X		0x42	136
InDeselect	X		0x44	139
InRelease	X		0x52	140
InSelect	X		0x54	141
InAutoPoll	X		0x60	144
Target				
TgInitAsTarget		X	0x8C	151
TgSetGeneralBytes		X	0x92	158
TgGetData		X	0x86	160
TgSetData		X	0x8E	164
TgSetMetaData		X	0x94	166
TgGetInitiatorCommand		X	0x88	168
TgResponseToInitiator		X	0x90	170
TgGetTargetStatus		X	0x8A	172

7.1 Error handling

In some of the commands detailed hereafter, there is a status byte returned by the PN532 reflecting if the RF communication has been successful or not. In case of unsuccessful command, only the status byte is sent back to the host controller.

Moreover, this byte contains two separated bits (bits 7 and 6) used for specific purposes.



- The **NADPresent** bit informs the host controller that the payload data contained in the PN532 answer frame for the **`InDataExchange`** or **`TgGetData`** contain a NAD byte. See **`SetParameters`** command §7.2.9, p: 85;
- The **MI** bit informs the host controller that the PN532 configured as target has received data from the initiator with MI bit set. Thus, chaining in reception is on going. See how the chaining is handled either by the initiator or by the target in examples given in §**7.4.5**: Chaining mechanism, p: 178;
- The Error Code (bits 0 to 5) informs the host controller on the result of the command. When null, the operation has gone properly.

Otherwise, the possible error code values are the following:

Table 13. Error code list

Error cause	Error code
Time Out, the target has not answered	0x01
A CRC error has been detected by the CIU	0x02
A Parity error has been detected by the CIU	0x03
During an anti-collision/select operation (ISO/IEC14443-3 Type A and ISO/IEC18092 106 kbps passive mode), an erroneous Bit Count has been detected	0x04
Framing error during Mifare operation	0x05
An abnormal bit-collision has been detected during bit wise anti-collision at 106 kbps	0x06
Communication buffer size insufficient	0x07
RF Buffer overflow has been detected by the CIU (bit BufferOvfl of the register *CIU_Error*)	0x09
In active communication mode, the RF field has not been switched on in time by the counterpart (as defined in NFCIP-1 standard)	0x0A
RF Protocol error (cf. **Error! Reference source not found.**, description of the *CIU_Error* register)	0x0B
Temperature error: the internal temperature sensor has detected overheating, and therefore has automatically switched off the antenna drivers	0x0D
Internal buffer overflow	0x0E
Invalid parameter (range, format, …)	0x10

Error cause	Error code
DEP Protocol: The PN532 configured in target mode does not support the command received from the initiator (the command received is not one of the following: ATR_REQ, WUP_REQ, PSL_REQ, DEP_REQ, DSL_REQ, RLS_REQ **Error! Reference source not found.**).	0x12
DEP Protocol, Mifare or ISO/IEC14443-4: The data format does not match to the specification. Depending on the RF protocol used, it can be: • Bad length of RF received frame, • Incorrect value of PCB or PFB, • Invalid or unexpected RF received frame, • NAD or DID incoherence.	0x13
Mifare: Authentication error	0x14
ISO/IEC14443-3: UID Check byte is wrong	0x23
DEP Protocol: Invalid device state, the system is in a state which does not allow the operation	0x25
Operation not allowed in this configuration (host controller interface)	0x26
This command is not acceptable due to the current context of the PN532 (Initiator vs. Target, unknown target number, Target not in the good state, …)	0x27
The PN532 configured as target has been released by its initiator	0x29
PN532 and ISO/IEC14443-3B only: the ID of the card does not match, meaning that the expected card has been exchanged with another one.	0x2A
PN532 and ISO/IEC14443-3B only: the card previously activated has disappeared.	0x2B
Mismatch between the NFCID3 initiator and the NFCID3 target in DEP 212/424 kbps passive.	0x2C
An over-current event has been detected	0x2D
NAD missing in DEP frame	0x2E

7.2 Miscellaneous commands

7.2.1 Diagnose

This command is designed for self-diagnosis.

Input:

D4	00	NumTst	[InParam]

- **NumTst** Test number to be executed by the PN532 (1 byte),
- **InParam** Input parameters needed for some of the tests.

Output:

D5	01	OutParam

- **OutParam** Parameters returned to the host controller (after execution of the test).

Syntax Error Conditions:

- Unknown test number (NumTst).

Description:

There are some parameters in command packet. The controller sends a command packet with parameter length and parameter itself.

The PN532 returns result (**OutParam**) with 1 to 262 bytes length parameters.

Processing time of this command varies depending on the content of the processing.

NumTst = 0x00: Communication Line Test

This test is for communication test between host controller and the PN532. “Parameter Length” and “Parameters” in response packet are same as “Parameter Length” and “Parameter” in command packet.

- Parameter Length : m (0 <= m <= 262),
- Parameter : Data,
- Result Length : Same value of m + 1.

OutParam consists of NumTst concatenate with InParam.

NumTst = 0x01: ROM Test

This test is for checking ROM data by 8 bits checksum.

- Parameter Length : 0,
- Result Length : 1,
- Result : 0x00 ➔ OK,
  0xFF ➔ Not Good.

NumTst = 0x02: RAM Test

This test is for checking RAM; 768 bytes of XRAM and 128 bytes of IDATA.
The test method used consists of saving original content, writing test data, checking test data and finally restore original data. So, this test is non destructive.

- Parameter Length : 0,
- Result Length : 1,
- Result : 0x00 ➔ OK,
  0xFF ➔ Not Good.

NumTst = 0x04 : Polling Test to Target

This test is for checking the percentage of failure regarding response packet receiving after polling command transmission. In this test, the PN532 sends a FeliCa polling command packet 128 times to target. The PN532 counts the number of fails and returns the failed number to host controller. This test doesn't require specific system code for target.
Polling is done with system code (0xFF, 0xFF). The baud rate used is either 212 kbps or 424 kbps.

One polling is considered as defective after no correct polling response within 4 ms.

During this test, the analog settings used are those defined in command **`RFConfiguration`** within the item n°7 (§7.3.1, p: 101).

- Parameter Length : 1,
- Parameter : 0x01 ➔ 212 kbps,
  0x02 ➔ 424 kbps.
- Result Length : 1,
- Result : Number of fails (Maximum 128).

NumTst = 0x05 : Echo Back Test

In this test, the PN532 is configured in target mode. The analog settings used are those defined by using the command **`RFConfiguration`** with the item n°6 (§7.3.1, p: 101).
This test is running as long as a new command is not received from the host controller.
The principle of this test is that the PN532 waits for a command frame coming from the initiator and after the Reply Delay, sends it back to it whatever its content and its length are.

- Parameter Length : 3,
- Parameter 1 : Reply Delay (step of 0.5 ms),
- Parameter 2 : Content of the CIU_TxMode (@0x6302) register defining the baud rate and the modulation type in transmission,
- Parameter 3 : Content of the CIU_RxMode (@0x6303) register defining the baud rate and the modulation type in reception,
- Result Length : no result, the test runs infinitely, so no output frame is sent to the host controller.

For example:

- The PN532 is configured to receive frame with passive 106 kbps modulation type. The frames are sent back immediately.
  The MSB bit (CRC enable) of CIU_TxMode and CIU_RxMode must be set to 0.

D4	00	05	00	00	00

- The PN532 is configured to receive frame with passive 212 kbps modulation type. The frames are sent back with a delay of 64 ms.
  The MSB bit (CRC enable) of CIU_TxMode and CIU_RxMode must be set to 1.

D4	00	05	80	92	92

- The PN532 is configured to receive frame with passive 424 kbps modulation type. The frames are sent back immediately.
  The MSB bit (CRC enable) of CIU_TxMode and CIU_RxMode must be set to 1.

D4	00	05	00	A2	A2

NumTst = 0x06 : Attention Request Test or ISO/IEC14443-4 card presence detection
This test can be used by an initiator to ensure that a target/card is still in the field:

- In case of DEP target, an Attention Request command is sent to the target, and it is expected to receive the same answer from the target. In that case, the test is declared as successful;
- In case of ISO/IEC14443-4 card, a R(NACK) block is sent to the card and it is expected to receive either a R(ACK) block or the last I-Block. In that case, the test is declared as successful (ISO/IEC14443-4 card is still in the RF field).

In case of no or incorrect response, the Result informs about the status of the transaction (refer. to §7.1, p:67)

- Parameter Length : 0,
- Result Length : 1,
- Result : 0x00 ➔ OK,
  different from 0x00 ➔ Not OK, Status byte.

NumTst = 0x07 : Self Antenna Test

This test is used to check the continuity of the transmission paths of the antenna.

- Parameter Length : 1,
- Parameter : Threshold used for antenna detection (applied in register **Andet_Control** (**@610C**), see **Error! Reference source not found.**),

7	6	5 4	3 2 1	0
andet_bot	andet_up	andet_ithl[1:0]	andet_ithh[1:0]	andet_en

- Result Length : 1,
- Result : 0x00 ➔ OK (antenna is detected),
  different from 0x00 ➔ not OK (no antenna is detected).

7.2.2 GetFirmwareVersion

The PN532 sends back the version of the embedded firmware.

Input:

D4	02

Output:

D5	03	IC	Ver	Rev	Support

- **IC** Version of the IC. For PN532, the contain of this byte is 0x32,
- **Ver** Version of the firmware,
- **Rev** Revision of the firmware,
- **Support** indicates which are the functionalities supported by the firmware. When the bits are set to 1, the functionality is supported, otherwise (bit set to 0) it is not.

RFU	RFU	RFU	RFU	RFU	ISO18092	ISO/IEC 14443 TypeB	ISO/IEC 14443 TypeA
7	6	5	4	3	2	1	0

Description:

In the case of the PN532, the version is 1.6.

That leads to **IC** : 0x32,
Ver : 0x01,
Rev : 0x06.
Support : 0x07.

7.2.3 GetGeneralStatus

This command allows the host controller to know at a given moment the complete situation of the PN532.

Input:

D4	04

Output:

D5	05	Err	Field	NbTg	$[Tg_1]$	$[BrRx_1]$	$[BrTx_1]$	$[Type_1]$
					$[Tg_2]$	$[BrRx_2]$	$[BrTx_2]$	$[Type_2]$
					SAM status			

- **Err** is an error code corresponding to the latest error detected by the PN532,
- **Field** indicates if an external RF field is present and detected by the PN532 (**Field = 0x01**) or not (**Field = 0x00**),
- **NbTg** is the number of targets currently controlled by the PN532 acting as initiator,
- For each target controlled by the PN532 (maximum 2 targets):
  - **Tg_i** : logical number
  - **$BrRx_i$** : bit rate in reception
    - 0x00 : 106 kbps
    - 0x01 : 212 kbps
    - 0x02 : 424 kbps
  - **BrTxi** : bit rate in transmission
    - 0x00 : 106 kbps
    - 0x01 : 212 kbps
    - 0x02 : 424 kbps
  - **Typei** : modulation type
    - 0x00 : Mifare, ISO/IEC14443-3 Type A, ISO/IEC14443-3 Type B, ISO/IEC18092 passive 106 kbps
    - 0x10 : FeliCa, ISO/IEC18092 passive 212/424 kbps
    - 0x01 : ISO/IEC18092 Active mode
    - 0x02 : Innovision Jewel tag
- **SAM status** informs on the status of the SAM connection (for more information, refer to §7.2.10, p:89).

Description:

Err:

Err contains error code as defined in the error code paragraph (§7.1, p:67). After the execution of the **`GetGeneralStatus`** command, the content of the latest error is cleared.

Field:

The PN532 scans the RF field to inform the host controller if an external field is detected or not.

Tg_i, $BrRx_i$, $BrTx_i$ and $Type_i$:

When the PN532 is configured as initiator, for all the targets handled by the PN532, the indication of the baud rate used and the modulation is given. The **Tgi** information corresponds to the logical target number attributed by the PN532 when a previous **`InListPassiveTarget`**, **`InJumpForDEP`** or **`InJumpForPSL`** command has been used[6].

SAM status:

This byte is cleared once read. It informs the host controller on the status of a possible transaction between an external reader and the SAM connected to the PN532.

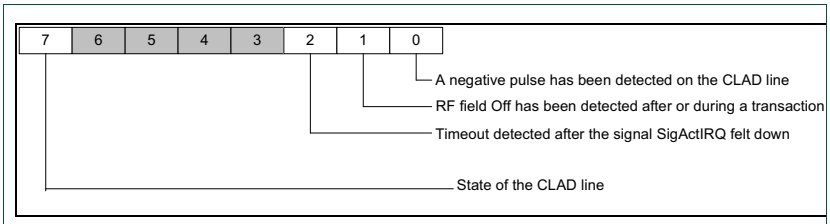


Fig 52. SAM status byte definition

When **bit 0** is set to 1, a full negative pulse has been detected on the CLAD line.

When **bit 1** is set to 1, an external RF field has been detected and switched off during or after a transaction.

When **bit 2** is set to 1, a timeout has been detected after SigActIRQ has felt down.

When **bit 7** is set to 1, the CLAD line is high level whereas when this bit is set to 0, the CLAD line is low level.

Warning: When the SAM is not powered, **bit 7** is not significant. In other words, for example when the PN532 is configured in virtual card mode, and if no external RF field is detected, this bit will be read as high, whatever the real level of the input.

[6] The only command capable of activating multiple targets is **`InListPassiveTarget`**, and in that case, the baud rate and the modulation type is the same for all the targets.

7.2.4 ReadRegister

This command is used to read the content of one or several internal registers of the PN532 (located either in the SFR area or in the XRAM memory space).

Input:

D4	06	$ADR1_H$	$ADR1_L$	…	$ADRn_H$	$ADRn_L$

- **$ADR1_H$ $ADR1_L$** First address (High and Low bytes),
- **$ADRn_H$ $ADRn_L$** n^{th} address (High and Low bytes).

Output:

D5	07	VAL1	…	VALn

- **VAL1** Value read in the register located at address ADR1,
- **VALn** Value read in the register located at address ADRn.

Syntax Error Conditions:

- Unknown SFR address.

Description:

For each address **ADR**, the PN532 performs a reading operation in the register located at address **ADR**. Then the value is returned in the **VAL** parameter.

- SFR registers.
  The host controller has to set the High Byte of the address to 0xFF, the real address of the register is given by the low byte.
  The list of the SFR registers accessible for the host controller is configured in the firmware. The firmware gives access control to the following SFR registers:

Table 14. List of SFR registers

Address	Register	Address	Register	Address	Register
0x87	PCON	0xA8	IE0	0xE8	IEN1
0x9A	RWL	0xA9	SPIcontrol	0xF4	P7CFGA
0x9B	TWL	0xAA	SPIstatus	0xF5	P7CFGB
0x9C	FIFOFS	0xAB	HSU_STA	0xF7	P7
0x9D	FIFOFF	0xAC	HSU_CTR	0xF8	IP1
0x9E	SFF	0xAD	HSU_PRE	0xFC	P3CFGA
0x9F	FIT	0xAE	HSU_CNT	0xFD	P3CFGB
0xA1	FITEN	0xB0	P3		
0xA2	FDATA	0xB8	IP0		
0xA3	FSIZE	0xD1	CIU_COMMAND		

- XRAM memory mapped registers
  The complete address is given by the high and low bytes of address.
  See **Error! Reference source not found.**.

Example:

The host controller reads the value 0x21 in the register *STATUS1* located at address 0x6337 and the value 0x07 in the *P7CFGA* register (SFR) located at address 0xF4.

The frame from the host controller to the PN532 is:

D4	06	63	37	FF	F4

And the one returned by the PN532 is:

D5	07	21	07

7.2.5 WriteRegister

This command is used to overwrite the content of one or several internal registers of the PN532 (located either in the SFR area or in the XRAM memory space).

Input:

D4	08	$ADR1_H$	$ADR1_L$	VAL1	…	$ADRn_H$	$ADRn_L$	VALn

- **$ADR1_H$ $ADR1_L$** First address (High and Low bytes),
- **VAL1** First value to be written,
- **$ADRn_H$ $ADRn_L$** n^{th} address (High and Low bytes),
- **VALn** n^{th} value to be written.

Output:

D5	09

Syntax Error Conditions:

- Unknown SFR address.

Description:

For each address **ADR**, the PN532 performs a writing operation of the value **VAL** in the register located at address **ADR**.

- SFR registers.
  The host controller has to set the High Byte of the address to 0xFF, the real address of the register is given by the low byte.
  The list of SFR registers that may be acceded is the same as the one defined in the **`ReadRegister`** command (§7.2.4, p:76).
- XRAM memory mapped registers.
  The complete address is given by the high and low bytes of address.
  See **Error! Reference source not found.**.

Example:

The host controller writes the value 0x39 in the register *CIU_CommIEn* located at address 0x6332 and the value 0x00 in the *P7CFGA* register (SFR) located at address 0xF4.

The frame from the host controller to the PN532 is:

D4	08	63	32	39	FF	F4	00

and the frame returned by the PN532 is:

D5	09

Warning:

The behavior of the PN532 may be altered by this command.

This command is only recommended for debug purposes.

7.2.6 ReadGPIO

The PN532 reads the value for each port and returns the information to the host controller.

Input:

D4	0C

Output:

D5	0D	P3	P7	I0I1

- The field **P3** contains the state of the GPIO located on the P3 port,

0	0	P35	P34	P33	P32	P31	P30
		5	4	3	2	1	0

- The field **P7** contains the state of the GPIO located on the P7 port,

0	0	0	0	0	P72	P71	0
					2	1	0

- The field **I0I1** contains the state of the GPIO located on the Interface Mode Lines.

0	0	0	0	0	0	I1	I0
						1	0

Description:

The GPIOs may be used with the following limitations of usage:

- **P32** corresponds to the pin P32_INT0.
  P32 can be used as standard GPIO and is therefore not used as external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§7.2.11, p:98), this pin can be used for the waking up.
  Moreover, when configured to use the handshake mechanism (§6.3, p:48), this pin may be used for the H_REQ line.
- **P33** corresponds to the pin P33_INT1.
  P33 can be used as standard GPIO and is therefore not used as external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§7.2.11, p:98), this pin can be used for the waking up.
    - **P34** corresponds to the pin P34/SIC_CLK.
      When configured to use the SAM companion chip (see **`SAMConfiguration`** command (§7.2.10 p:89)), **P34** is used for the CLAD line.

- **P71** and **P72** are the GPIOs corresponding to the pins MISO/P71 and SCK/P72 of the SPI bus. They can be used as GPIO when the PN532 is not configured to use the SPI interface to communicate with the host controller.
- **I0** and **I1** (see § 6.1.1, p:24) are the GPIOs used also to select the host controller interface. Once the selection has been done by the firmware, these two pins can be used as GPIOs.

7.2.7 WriteGPIO

The PN532 applies the value for each port that is validated by the host controller (bit **Val** of each port).

Input:

D4	0E	P3	P7

- The field **P3** contains the value to apply to the GPIO located on the P3 port,

Val	nu	P35	P34	P33	P32	P31	P30
7		5	4	3	2	1	0

- The field **P7** contains the value to apply to the GPIO located on the P7 port.

Val	nu	nu	nu	nu	P72	P71	0
7					2	1	0

Output:

D5	0F

Description:

For each port that is validated (bit **Val** = 1), all the bits are applied simultaneously. It is not possible for example to modify the state of the port P32 without applying a value to the ports P30, P31, P33, P34 and P35.

As for the command **`ReadGPIO`** (see §7.2.6, p:79), the GPIO may be used with the following limitations of usage:

- **P32** corresponds to the pin P32_INT0. It can be used as standard GPIO and is therefore not used as external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§7.2.11, p:98), this pin can be used for the waking up.
  Moreover, when configured to use the handshake mechanism (§6.3, p:48), this pin is used for the H_REQ line.
- **P33** corresponds to the pin P33_INT1. It can be used as standard GPIO and is therefore not used as external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§7.2.11, p:98), this pin can be used for the waking up.
- **P34** corresponds to the pin P34/SIC_CLK.
  P34 can be used as standard GPIO
  Moreover, when configured to use the SAM companion chip (**S** see **`SAMConfiguration`** command (§7.2.10 p:89)), this pin is used for the CLAD line.

- **P71** and **P72** are the GPIOs corresponding to the pins MISO/P71 and SCK/P72 of the SPI bus. The host controller shall not modify these GPIOs when the link selected to communicate with the host controller is the SPI bus.

Example:

The host controller wants to:

- set P30 and P31,
- reset P32, P33, P34 and P35,
- leave P70, P71 and P72 unchanged.

The frame from the host controller to the PN532 is:

D4	0E	83	00

And the frame returned by the PN532 is:

D5	0F

The host controller wants to set all the bits of P3 and P7:

The frame from the host controller to the PN532 is:

D4	0E	BF	87

And the frame returned by the PN532 is:

D5	0F

7.2.8 SetSerialBaudRate

This command is used to select the baud rate on the serial link between the host controller and the PN532 (HSU).

Input:

D4	10	BR

- **BR** is a byte indicating the baud rate requested by the host controller:
  - 0x00 9.6 kbaud,
  - 0x01 19.2 kbaud,
  - 0x02 38.4 kbaud,
  - 0x03 57.6 kbaud,
  - 0x04 115.2 kbaud,
  - 0x05 230.4 kbaud,
  - 0x06 460.8 kbaud,
  - 0x07 921.6 kbaud,
  - 0x08 1.288 Mbaud.

Output:

D5	11

Syntax Error Conditions:

- The requested baud rate is missing or not possible (BR > 8),
- The link used is not the HSU (High Speed UART),
- Incorrect command length.

Description:

When another link between the host controller and the PN532 is used (I2C or SPI), this command is not allowed. In that case, the PN532 will inform the host controller of an application level error.

The PN532 changes the baud rate from the old one to the new one **only** after having sent the Response of the command **and** having received one ACK frame sent by the host controller.

This ACK frame is usually optional, but in the case of this specific **`SetSerialBaudRate`** command, it is mandatory.

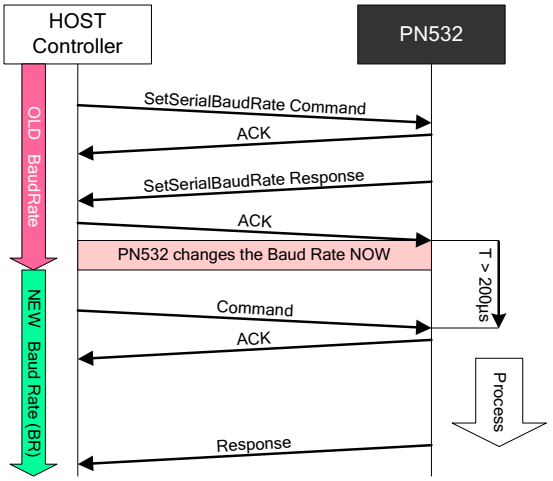


Fig 53. SetSerialBaudRate

The host controller shall send the next command at least 200µs (T period in Fig 53) after the ACK has been received.

7.2.9 SetParameters

This command is used to set internal parameters of the PN532, and then to configure its behavior regarding different cases.

Input:

D4	12	Flags

- **Flags** is a bit-field byte which individual definition is the following:

RFU	6	5	4	RFU	2	1	0

- bit 0: fNADUsed ➔ Use of the NAD information in case of initiator configuration (DEP and ISO/IEC14443-4 PCD).
- bit 1: fDIDUsed ➔ Use of the DID information in case of initiator configuration (or CID in case of ISO/IEC14443-4 PCD configuration).
- bit 2: fAutomaticATR_RES ➔ Automatic generation of the ATR_RES in case of target configuration.
- bit 3: RFU ➔ **Must** be set to 0.
- bit 4: fAutomaticRATS ➔ Automatic generation of the RATS in case of ISO/IEC14443-4 PCD mode.
- bit 5: fISO14443-4_PICC ➔ The emulation of a ISO/IEC14443-4 PICC is enabled.
- bit 6: fRemovePrePostAmble ➔ The PN532 does not send Preamble and Postamble.
- bit 7: RFU ➔ **Must** be set to 0.

Output:

D5	13

Syntax Error Conditions:

- **Flags** parameter is missing,
- Incorrect command length.

Description:

fNADUsed (DEP and ISO/IEC14443-4 PCD):

***In DEP mode*:**

By default, the PN532 initiator does not use the NAD byte in the Transport Protocol, so the host controller must set this flag in order to use NAD. The NAD value itself is set by the host controller directly in the **`InDataExchange`** command (see §7.3.8, p:127).

On the opposite side, when the PN532 configured as target is in front of an initiator using NAD byte, the NAD value received by the PN532 will be transmitted to the PN532 host controller for further analysis, and the NAD value to be returned to the initiator will be elaborated by the host controller of the PN532 (so, the NAD values are transported respectively within the **`TgGetData`** and **`TgSetData`** commands).

In both cases (PN532 initiator or PN532 target in Fig 54), the host controller (**A** or **B**) knows that the payload data of the transport commands include NAD values by checking the higher bit of the status byte returned (see §7.1: Error handling, p:67 and Fig 55).

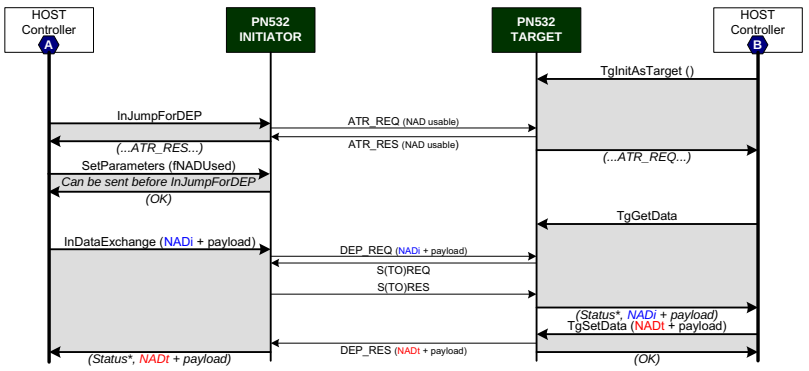


Fig 54. fNADUsed

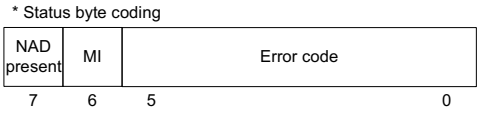


Fig 55. Status Byte definition

***In ISO/IEC14443-4 PCD mode*:**

By default, the PN532 initiator does not use the NAD byte in the Transport Protocol, so the host controller must set this flag in order to use NAD. The NAD value itself is set by the host controller directly in the **`InDataExchange`** command (see §7.3.8, p:127).

In reception mode, the NAD value received by the PN532 will be transmitted to the PN532 host controller.

fDIDUsed (DEP and ISO/IEC14443-4 PCD):

By default, the PN532 initiator does not use the DID byte for DEP and the CID byte for ISO/IEC14443-4 PCD in the Transport Protocol (not multi-target configuration). So the host controller must set this flag in order to use DID / CID. In that case, the DID / CID value itself is completely handled internally by the firmware.
The DID / CID has a fixed value of 0x01 when fDIDUsed is set to 1.

fAutomaticATR_RES:

By default, the PN532 target sends back to the initiator the ATR_RES after having received an ATR_REQ.
For various reasons, the host controller of the PN532 may want to choose the content of the Gt (general bytes of the target), only after having received the general bytes of the initiator. In that case, the host controller must set this flag to 0.

To have a detailed description of these two possibilities, see the commands **`TgSetGeneralBytes`** (§7.3.15, p:158) and **`TgInitAsTarget`** (§7.3.14, p:151).

fAutomaticRATS:

When executing the command **`InListPassiveTarget`** at 106 kbps, the PN532 may initialize cards supporting ISO/IEC14443-4 protocol (the PN532 knows that the target is ISO/IEC14443-4 compliant by analyzing the SEL-RES - SAK byte).

In that case, and if the flag fAutomaticRATS is set, the first command sent to the card is a RATS command. This command is automatically elaborated by the PN532.

If the user does not want to use the ISO/IEC14443-4 protocol with a card that is ISO/IEC14443-4 compliant, this flag must be set to 0.

fISO14443-4_PICC:

When the flag fISO14443-4_PICC is set, the PN532 is able to behave like a ISO/IEC14443-4 PICC. So, the PN532 answers a ATS after having received a RATS.

If the user does not want to use the possibility to emulate a ISO/IEC14443-4 PICC, this flag must be set to 0.

fRemovePrePostAmble:

When the flag fRemovePrePostAmble is set to one, the PN532 does not use the preamble and postamble fields in the messages it sends to the host controller. Normally, these two fields have a fixed length (1 byte) and a fixed content (0x00); with the fRemovePrePostAmble flag set to one, 2 bytes are then suppressed from the message (ACK frame or INFORMATION frame), allowing saving a bit in overall data throughput.

Example of the **`GetFirmwareVersion`** command:

with Preamble and Postamble:

00 00 FF 06 FA D5 03 32 01 05 07 E9 00

without Preamble and Postamble:

00 FF 06 FA D5 03 32 01 05 07 E9

Summary of the default settings:

If the user does not use the **`SetParameters`** command, the following settings apply:

Table 15. Default values of internal flags

Property	Default value
fNADUsed	Not used
fDIDUsed	Not used
fAutomaticATR_RES	Yes, automatic
fAutomaticRATS	Yes, automatic
fISO14443-4_PICC	Yes, enabled
fRemovePrePostAmble	No

7.2.10 SAMConfiguration

This command is used to select the data flow path by configuring the internal serial data switch.

Input:

D4	14	Mode	Timeout	[IRQ]

- **Mode** defines the way of using the SAM (Security Access Module):
  - 0x01: **Normal mode**, the SAM is not used; this is the default mode,
  - 0x02: **Virtual Card**, the couple PN532+SAM is seen as only one contactless SAM card from the external world,
  - 0x03: **Wired Card**, the host controller can access to the SAM with standard PCD commands (**`InListPassiveTarget`**, **`InDataExchange`**, …),
  - 0x04: **Dual Card**, both the PN532 and the SAM are visible from the external world as two separated targets.

  Virtual, Wired and Dual Card mode are only valid with 106kbps ISO14443-3 and 4 type A and Mifare.
- **Timeout** defines the time-out only in Virtual card configuration (**Mode** = 0x02). In Virtual Card mode, this field is mandatory; whereas in the other mode, it is optional.
  This parameter indicates the timeout value with a LSB of 50ms.
  There is no timeout control if the value is null (**Timeout** = 0).
  The maximum value for the timeout is 12.75 sec (**Timeout** = 0xFF).
- **IRQ** specifies if the PN532 takes care of the P70_IRQ pin or not. If the value is null (IRQ =0x00), the P70_IRQ pin remains at high level; whereas if the value is 0x01, the P70_IRQ pin is driven by the PN532. If the P70_IRQ parameter is not present, the default value is 0x01.

Output:

D5	15

Syntax Error Conditions:

- Incorrect values for **Mode** and **Timeout** parameters,
- **Mode** parameter is missing,
- **Timeout** parameter missing in Virtual Card Mode.

Description:

A SAM companion chip can be used to bring security. It is connected to the PN532 by using a S2C interface (SigIn (pin #36), SigOut (pin #35) and CLAD (pin #34)). The CLAD line is optional.

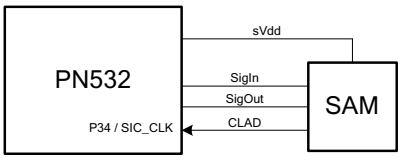


Fig 56. SAM electrical connection

There are four possible configurations and three of them allow using a companion SAM (either internally or externally).

- The **Normal mode** is the default mode. It is used when no communication with the SAM is needed, either from the host controller or from an external PCD. In this mode, the PN532 can act either as initiator or as target.

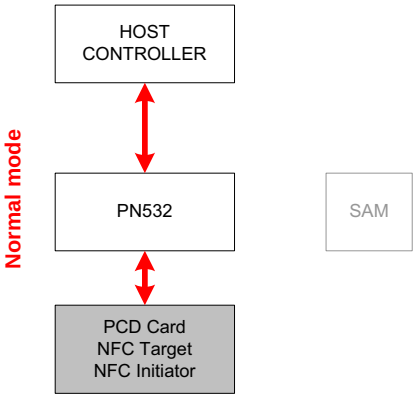


Fig 57. SAM: Normal mode

- The **Wired Card** mode is used to communicate with the SAM internally. No RF field is emitted. The PN532 does not react to an external PCD.

  In this mode, the PN532 acts as PCD of the SAM card and the commands to be used are:
  - **`InListPassiveTarget`** to activate the SAM.
  - **`InDataExchange`** to exchange data with the SAM
    Depending on the type of the SAM, the correct protocol will be applied (Mifare or ISO/IEC14443-4).
  - **`InDeselect`** to deselect the SAM properly.

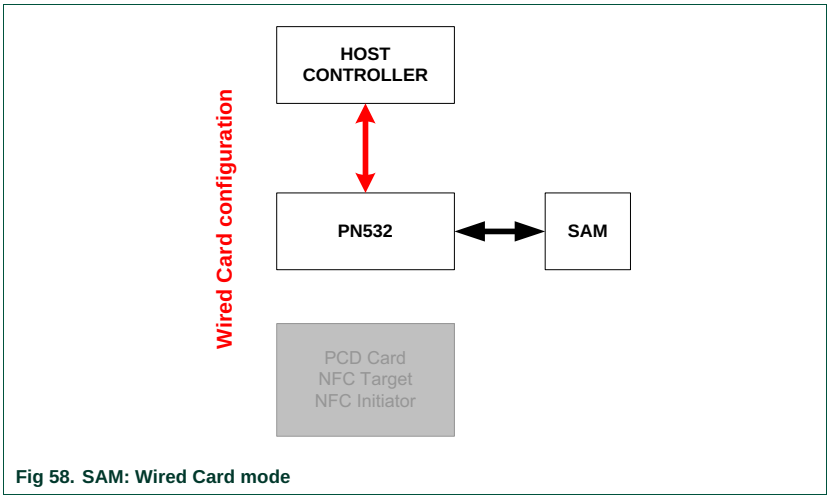


Fig 58. SAM: Wired Card mode

- In the **Virtual Card** mode, the PN532 acts as real contactless card, and all the responses to external requests are handled by the SAM itself; neither the PN532 nor the host controller have to take care of the data exchanges.
  The PN532 acts just as bridge (analog front-end + antenna) between the PCD and the SAM.

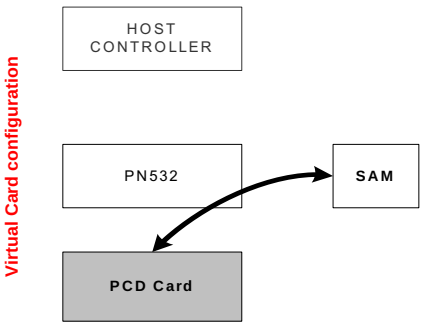


Fig 59. SAM: Virtual Card mode

The PN532 is in charge of monitoring the link between the external PCD and the SAM in order to inform the host controller when a contactless transaction is potentially finished[7].
Then, the host controller may react in a way it chooses (e.g. switch to Wired Card mode to check internally in the SAM card the result of the contactless transaction).

Once in virtual card mode, the PN532 monitors the SigInActIRQ interrupt permanently. This interrupt is generated internally in the PN532 by the CIU, for every byte sent by the SAM (SigIn). In this way, if one interrupt appears, that necessarily means that a real transaction between the external PCD and the SAM is started.

The firmware is in charge of clearing this interrupt, to be able to detect any further data exchanged between the SAM and the PCD.

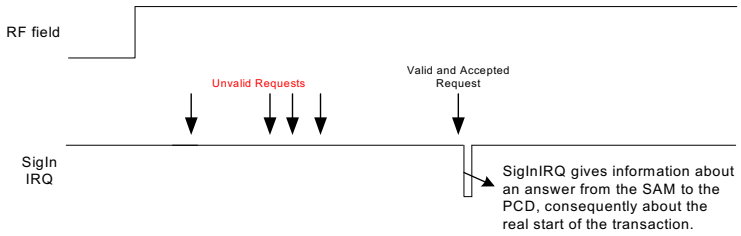


Fig 60. SAM: Detection of the start of a transaction

Then, there are three different events to detect the end of the transaction.

[7] There is no way for the PN532 to be certain of the completion of a contacless transaction in virtual card mode. Making use of different mechanisms (monitoring of the SigInIRQ signal and the CLAD line, timeout), the PN532 can only help the host controller to be informed about the real end of a transaction, but the final check has to be done by the host controller itself.

- *CLAD rising edge*:

  The CLAD line enables the PN532 to know when the contactless transaction between the external PCD and the SAM is completed[8].
  This line is active low. The SAM asserts this line after having received a REQUEST from the external PCD. It stays active as long as the transaction is not completed (HALT in ISO/IEC14443-3 or DESELECT in ISO/IEC14443-4).
  In that case, the PN532 informs immediately the host controller by using the P70_IRQ pin, because the fact that a complete negative pulse has been detected on the CLAD line with the RF field still active is the best indicator that a complete contacless transaction has occurred.

  The warning of the host controller by the PN532 is done in **handshake mode** as described in the §6.3, p:48.

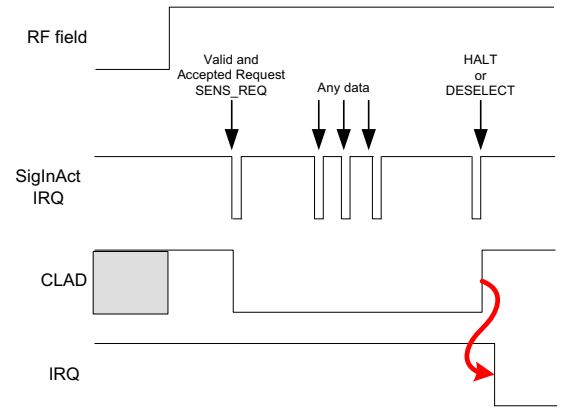


Fig 61. SAM: P70_IRQ triggered by the CLAD line

[8] The CLAD line may be not connected between the SAM and the PN532 and/or not managed by the SAM. Therefore, this is an optional line.

- *RF Off detected:*

  Another event that leads the PN532 to consider that the transaction between the PCD and the SAM is finished is the external RF field cut (a PCD may switch the RF field off after a transaction).

  To consider that the RF field cut is synonymous of an end of transaction, the PN532 must have first detected some activities between the SAM and the external PCD by using SigInActIRQ.

  The warning of the host controller by the PN532 is done **in handshake mode** as described in §6.3, p:48.

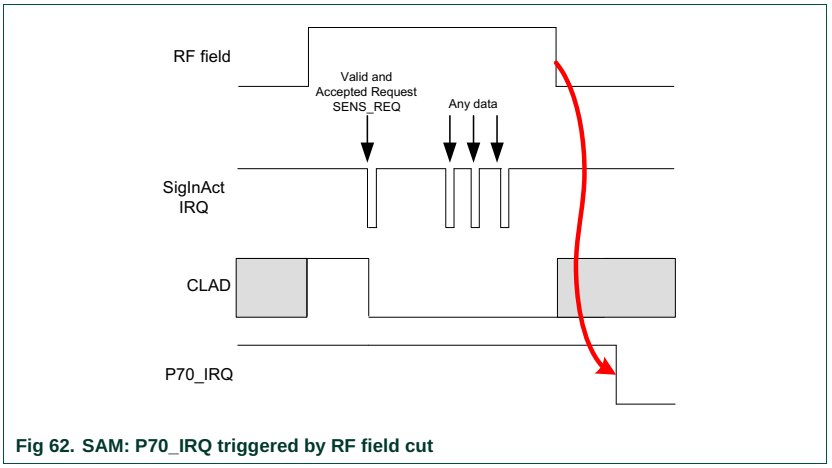


Fig 62. SAM: P70_IRQ triggered by RF field cut

- *Timeout:*

  In order to detect a potential end of transaction, when CLAD signal is not available and when the RF field is not switched off at the end of the transaction, a timeout monitoring is used.
  Each time a SigInActIRQ interrupt is detected, a counter is re-started and if no other SigInActIRQ is detected before the counter has reached the value zero, the PN532 considers that the transaction is finished and informs the host controller.

  The timeout value is chosen by the host controller with the **Timeout** optional parameter. If no **Timeout** parameter is given, no control is done based on timeout; only CLAD and RF field cut are used to detect an end of transaction.

  The warning of the host controller by the PN532 is done **in handshake mode** as described in the §6.3, p:48.

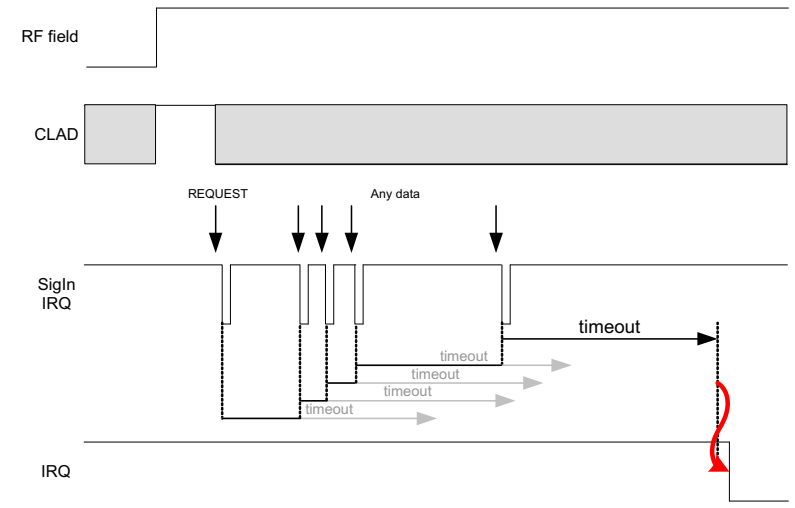


Fig 63. SAM: P70_IRQ triggered by Timeout

Whatever the reason for what it has been informed by the P70_IRQ pin that something happened with the SAM, the host controller is invited to use the **`GetGeneralStatus`** command (§7.2.3, p:74) to have more details. It will know by this way, if a complete pulse has been detected on the CLAD line, if the timeout occurred …

- The **Dual Card** configuration allows presenting two different cards to the external world: the SAM and the target functionality of the PN532 itself.

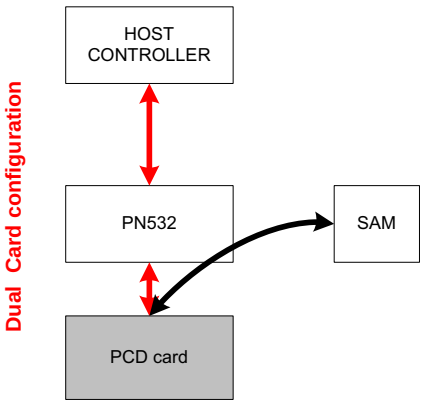


Fig 64. SAM: Dual Card mode

In this mode, an external R/W can communicate alternatively with the SAM or with the PN532 internal Contactless UART by using normal passive 106kbps selection methods (DLS_REQ / WUPA).

Example:

One possible scenario using the SAM is described below:

- Configure the PN532 in **Virtual Card** mode to allow transaction from external PCD to the SAM.
  The PN532 goes into Power Down mode waiting for external RF field.
  - SAMConfiguration (0x02, 0x00) *Virtual Card mode (No timeout used here)*
  - As soon as the host controller is informed that a transaction has occurred, it can switch to **Wired Card** mode and access to the SAM directly to check the result of the transaction.
    Before switching into this mode, the host controller must use the normal mode first.
    - GetGeneralStatus();
    - SAMConfiguration (0x03); *Wired Card mode*
    - InListPassiveTarget (…); *To access to the SAM internally*
    - InDataExchange (…); *To exchange data with the SAM*
  - Either returns in **Normal mode** or to **Virtual Card** mode to wait for a new transaction.
    - SAMConfiguration (0x01); *Normal mode*
    - SAMConfiguration (0x02, 0x00); *No timeout is used in this example*

7.2.11 PowerDown

This command can be used to put the PN532 (including the contactless analog front end) into Power Down mode in order to save power consumption.

Input:

D4	16	WakeUpEnable	[GenerateIRQ]

- **WakeUpEnable** defines the authorized sources to wake up the PN532 from Power Down mode,
- Optional parameter **GenerateIRQ** defines whether once waken up, the PN532 handles or not the P70_IRQ pin.

Output:

D5	17	Status

- **Status** indicates if the command is accepted or not (see §7.1, p: 67).

Syntax Error Conditions:

- **WakeUpEnable** parameter missing,
- Incorrect command length.

Description:

Different sources of wake up may be selected with this command with the **WakeUpEnable** parameter.

I2C	GPIO	SPI	HSU	RF Level Detector	RFU	INT1	INT0
7	6	5	4	3	2	1	0

Of course, it is possible to select more than one individual wake up source, and then the user may combine for example RFLevelDetector and HSU.

The “RFU” bit (bit 2) **must** be 0.

If the PN532 is currently activated as DEP target or ISO/IEC14443 PICC, the session will be lost with this command and the internal state returned by a **`TgGetTargetStatus`** command will be *TG_RELEASED* or *PICC_RELEASED* see §7.3.21, p:172)

Remarks:

- If the handshake mode is used, the PN532 goes automatically into Power Down mode after **`TgInitAsTarget`** command is launched. It wakes up only when an external RF field is detected or when the host controller sends a new command (§6.3, p: 48).
- The PN532 needs approximately **1 ms** to get into Power Down mode, after the command response. Sending host commands during this time is not recommended

GenerateIRQ parameter:

This parameter is only useful when **RFLevelDetector** bit is set to 1. The host controller sets this parameter to 1 when it wants to be warned that the PN532 has been awake by external RF field detection.
In that case, as soon as it is waken up, the PN532 asserts the P70_IRQ pin connected to the host controller, informing that it has left the Power Down mode.
Then, the next command received from the host controller makes the PN532 to release the P70_IRQ pin to level 1 (the host controller can use a **`GetGeneralStatus`** command to check that the RF field is present).
If the handshake mechanism is used, the same description than the one done in the "*SAMConfiguration: Virtual Card mode*" paragraph applies (See §6.3.2.5, p: 53 or §6.3.3.5, p: 59 or §6.3.4.5, p: 64).

HSU wake up condition:

When the host controller sends a command to the PN532 on the HSU link in order to exit from Power Down mode, the PN532 needs some delay to be fully operational (the real waking up condition is the 5th rising edge on the serial line, see **Error! Reference source not found.**).

As a consequence, if the host controller wants to be sure that the command will not be lost or partially received, some precautions must be taken:

- Either send a command with large preamble containing dummy data,
- Or send first a 0x55 dummy byte and wait for the waking up delay ($T_{wake\ up\ time}$) before sending the command frame.

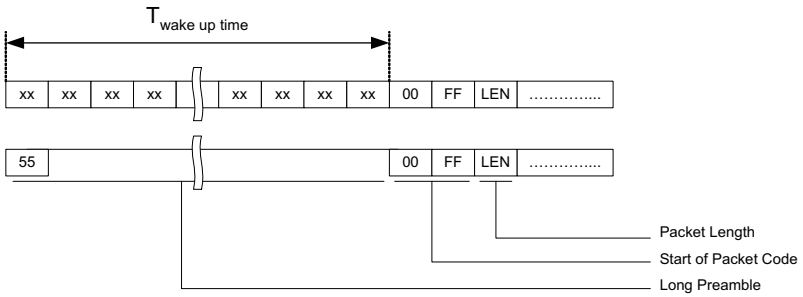


Fig 65. HSU Wake up

I2C wake up condition:

When the host controller sends a command to the PN532 on the I2C link in order to exit from Power Down mode, the PN532 will stretch the SCL line after having recognized its own slave address and releases the SCL line after T_osc_start[9].

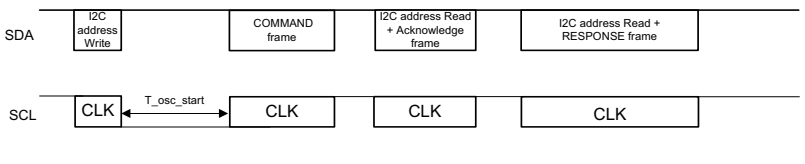


Fig 66. I2C Wake up

SPI wake up condition:

When the host controller sends a command to the PN532 on the SPI link in order to exit from Power Down mode, the PN532 needs some delay to be fully operational (T_osc_start).

In order that the command is acknowledged and executed, the Host controller has to maintain the NSS line to low level before sending the SPI command.

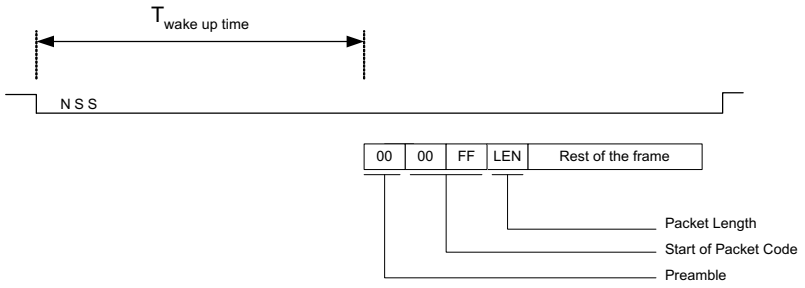


Fig 67. SPI wake up

[9] This is the time needed by the FW to be ready to operate when the oscillator is shut down. It is smaller than 2ms with appropriate quartz and layout.

7.3 RF Communication command

7.3.1 RFConfiguration

This command is used to configure the different settings of the PN532 as described in the input section of this command.

Input:

D4	32	CfgItem	ConfigurationData [ ]

The **ConfigurationData [ ]** field length and content depend on the **CfgItem** as follows:

- **CfgItem = 0x01: RF field** (**ConfigurationData [ ]** length is 1 byte)

  RFConfiguration allows switching **on** or **off** the RF field immediately.

7	6	5	4	3	2	1	0
nu	nu	nu	nu	nu	nu	Auto RFCA	RF on/off
						0: Off 1: On	0: Off 1: On

  When the bit **AutoRFCA** is off, the PN532 does not need to take care of external field before switching on its own field.
  In other words, if the bit **AutoRFCA** is off and **RFon/off** is on, the PN532 will generate RF field whatever external field is (present or not).

- **CfgItem = 0x02: Various timings** (3 bytes)

Table 16. Various timings

Byte #	Variable definition	Variable name
Byte 1	RFU	
Byte 2	ATR_RES TimeOut	fATR_RES_Timeout
Byte 3	TimeOut during non-DEP communications	fRetryTimeout

The **first byte** is RFU.

The **second byte** in this item defines the timeout between ATR_REQ and ATR_RES when the PN532 is in initiator mode. A target is considered as mute if no valid ATR_RES is received within this timeout value. In this way, the PN532 can easily detect non TPE target in passive 212-424 kbps mode.
The default value for this parameter is 0x0B (102.4 ms).

The **third byte** defines the timeout value that the PN532 uses in the **`InCommunicateThru`** (§7.3.9, p: 136) command to give up reception from the target in case of no answer.
The default value for this parameter is 0x0A (51.2 ms).
This timeout definition is also used with **`InDataExchange`** (§7.3.8, p: 127) when the target is a FeliCa or a Mifare card (Ultralight, Standard …).

For the timings of CfgItem 0x02, the coding is the following:

In case n = 0 No timeout

In case 1 ≤ n ≤ 16 $T_{(\mu s)} = 100 \times 2^{(n-1)}$

Table 17.Timings definition for RFConfiguration command

Byte Value (n)	Timeout Value
0x00	no timeout
0x01	100 µs
0x02	200 µs
0x03	400 µs
0x04	800 µs
0x05	1.6 ms
0x06	3.2 ms
0x07	6.4 ms
0x08	12.8 ms
0x09	25.6 ms
0x0A	51.2 ms
0x0B	102.4 ms
0x0C	204.8 ms
0x0D	409.6 ms
0x0E	819.2 ms
0x0F	1.64 sec
0x10	3.28 sec

- **CfgItem = 0x04: MaxRtyCOM** (1 byte)

  The information **MaxRtyCOM** (1 byte) defines the number of retry that the PN532 will use in the **`InCommunicateThru`** (§7.3.9, p:136) command in case of mute target or error detected. This information is also used with **`InDataExchange`** (§7.3.8, p:127) when the target is a FeliCa or a Mifare card.
  The specific value 0xFF means that the PN532 retries eternally.
  The default value of this parameter is 0x00 (no retry, only one try).

- **CfgItem = 0x05: MaxRetries** (3 bytes)

Table 18.Maximum retries

Byte #	Variable name
Byte 1	MxRtyATR
Byte 2	MxRtyPSL
Byte 3	MxRtyPassiveActivation

The parameters **MxRtyATR**, **MxRtyPSL** and **MxRtyPassiveActivation** define the number of retries that the PN532 will use in case of the following processes:

– **MxRtyATR** is a byte containing the number of times that the PN532 will retry to send the ATR_REQ in case of incorrect reception of the ATR_RES (or no reception at all - timeout).
For **active mode**, value 0xFF means to try eternally, 0x00 means only once (no retry, only one try). The default value of this parameter is 0xFF (infinitely).
For **passive mode**, the value is always overruled with 0x02 (two retries).

– **MxRtyPSL** is a byte containing the number of times that:

- The PN532 will retry to send the PSL_REQ in case of incorrect reception of the PSL_RES (or no reception at all) for the NFC IP1 protocol,
- The PN532 will retry to send the PPS request in case of incorrect reception of the PPS response (or no reception at all) for the ISO/IEC14443-4 protocol.

Value 0xFF means to try eternally, 0x00 means only once (no retry, only one try).The default value of this parameter is 0x01 (the PSL_REQ/PPS request is sent twice in case of need).

– **MxRtyPassiveActivation** is a byte containing the number of times that the PN532 will retry to activate a target in **`InListPassiveTarget`** command (§7.3.5, p: 115).
Value 0xFF means to try eternally, 0x00 means only once (no retry, only one try).
The default value of this parameter is 0xFF (infinitely).

- **CfgItem = 0x0A: Analog settings for the baudrate 106 kbps type A** (11 bytes)

This CfgItem is used to choose the analog settings that the PN532 will use for the baudrate 106kbps type A.

When using this command, the host controller has to provide 11 values (**ConfigurationData [ ]**) for the following internal registers:

Table 19. Analog settings for the baudrate 106 kbps type A

Byte #	Register	Default values
Byte 1	CIU_RFCfg	0x59
Byte 2	CIU_GsNOn	0xF4
Byte 3	CIU_CWGsP	0x3F
Byte 4	CIU_ModGsP	0x11
Byte 5	CIU_Demod when own RF is On	0x4D
Byte 6	CIU_RxThreshold	0x85
Byte 7	CIU_Demod when own RF is Off	0x61
Byte 8	CIU_GsNOff	0x6F
Byte 9	CIU_ModWidth	0x26
Byte 10	CIU_MifNFC	0x62
Byte 11	CIU_TxBitPhase	0x87

Note:

Actually, there is only one **CIU_Demod** register which defines a setting used by the reader in reception only. But depending on the RF condition, two different settings can be used for this register:

- **CIU_Demod when own RF is On** defines a setting when its RF field is on during a reception i.e. initiator passive mode,
- **CIU_Demod when own RF is Off** defines a setting when its RF field is off during a reception i.e. initiator active mode.

- **CfgItem = 0x0B: Analog settings for the baudrate 212/424 kbps** (8 bytes)

This CfgItem is used to choose the analog settings that the PN532 will use for the baudrates 212/424kbps.
When using this command, the host controller has to provide 8 values (**ConfigurationData [ ]**) for the following internal registers:

Table 20. Analog settings for the baudrate 212/424 kbps

Byte #	Register	Default values
Byte 1	CIU_RFCfg	0x69
Byte 2	CIU_GsNOn	0xFF
Byte 3	CIU_CWGsP	0x3F
Byte 4	CIU_ModGsP	0x11
Byte 5	CIU_Demod when own RF is On	0x41
Byte 6	CIU_RxThreshold	0x85
Byte 7	CIU_Demod when own RF is Off	0x61
Byte 8	CIU_GsNOff	0x6F

Note:

Actually, there is only one **CIU_Demod** register which defines a setting used by the reader in reception only. But depending on the RF condition, two different settings can be used for this register:

- **CIU_Demod when own RF is On** defines a setting when its RF field is on during a reception i.e. initiator passive mode,
- **CIU_Demod when own RF is Off** defines a setting when its RF field is off during a reception i.e. initiator active mode.

- **CfgItem = 0x0C: Analog settings for the type B** (3 bytes)

This CfgItem is used to choose the analog settings that the PN532 will use for the type B when configured as PCD.

When using this command, the host controller has to provide 3 new values (**ConfigurationData [ ]**) for the following internal registers:

Table 21. Analog settings for the type B

Byte #	Register	Default values
Byte 1	CIU_GsNOn	0xFF
Byte 2	CIU_ModGsP	0x17
Byte 3	CIU_RxThreshold	0x85

Except for these two specific settings, the 8 remaining analog settings are the same as the CfgItem 106 kbps type A.

- **CfgItem = 0x0D: Analog settings for baudrates 212/424 and 848 kbps with ISO/IEC14443-4 protocol** (9 bytes)

  This CfgItem is used to choose the analog settings that the PN532 will use for the baudrates 212/424/848 kbps with ISO/IEC14443-4 cards.
  When using this command, the host controller has to provide 9 values (**ConfigurationData [ ]**) for the following internal registers:

Table 22. Analog settings for the baudrate 212/424 and 848 kbps with ISO/IEC14443-4 protocol

Byte #	Register	Default values	Baudrate
Byte 1	CIU_RxThreshold	0x85	212 kbps
Byte 2	CIU_ModWidth	0x15	
Byte 3	CIU_MifNFC	0x8A	
Byte 4	CIU_RxThreshold	0x85	424 kbps
Byte 5	CIU_ModWidth	0x08	
Byte 6	CIU_MifNFC	0xB2	
Byte 7	CIU_RxThreshold	0x85	848 kbps
Byte 8	CIU_ModWidth	0x01	
Byte 9	CIU_MifNFC	0xDA	

Except for these three specific registers (CIU_RxThreshold, CIU_ModWidth and CIU_MifNFC), the 8 remaining analog registers are the same as the previous CfgItem 0x0A.

Output:

D5	33

Syntax Error Conditions:

- Various timings values greater than 0x10 (item 2),
- Incorrect **CfgItem** value (0x00, 0x03, 0x06, 0x07, 0x08, 0x09 and greater than 0x0D),
- Incorrect command length.

7.3.2 RFRegulationTest

This command is used for radio regulation test.

Input:

D4	58	TxMode

- **TxMode** is the definition of the bit rate and of the framing used for data transmission.

7	6	5	4	3	2	1	0
nu	TxSpeed			nu	nu	TxFraming	

000: 106 kbps
001: 212 kbps
010: 424 kbps
011: 848 kbps

00: Mifare
10: FeliCa

Output:

This command never stops, so no output frame is sent.

Description:

The PN532 makes RF transmission with pseudo random numbers by using the PRBS15 bit of the *CIU_TestSel2* register (see **Error! Reference source not found.**). The transmission speed and the framing are indicated by the host controller with the **TxMode** parameter.

The **TxMode.TxSpeed** parameter defines the bit rate that is used during the transmission (106, 212, 424 kbps or 848 kbps).
The **TxMode.TxFraming** parameter defines the type of modulation (Mifare or FeliCa modulation).

The PN532 transmits data until a new command comes from the host controller.

7.3.3 InJumpForDEP

This command is used by a host controller to activate a target using either active or passive communication mode.
If a target is in the field, it will then be ready for DEP exchanges.

Input:

D4	56	ActPass	BR	Next	[PassiveInitiatorData] (4 or 5 bytes)
					[NFCID3i [0..9]]
					[Gi [0..n]]

- **ActPass** is the communication mode requested by the host controller
  - 0x00 : Passive mode,
  - 0x01 : Active Mode.
- **BR** is the Baud Rate to be used during the activation
  - 0x00 : 106 kbps,
  - 0x01 : 212 kbps,
  - 0x02 : 424 kbps.
- **Next** indicates if the optional fields of the command (**PassiveInitiatorData**, **NFCID3i** and **Gi**) are present (bit = 1) or not (bit = 0).
  - bit 0 : PassiveInitiatorData is present in the command frame,
  - bit 1 : NFCID3i is present in the command frame,
  - bit 2 : Gi is present in the command frame.
- **PassiveInitiatorData [ ]** is an array of data to be used during the initialization of the target in case of passive communication mode (**ActPass**). Depending on the Baud Rate specified, the content of this field is different:
  - **106 kbps:**
    The field is optional and is present only when the host controller wants to initialize a target with a known ID (according to **Error! Reference source not found.**, the first byte of this ID should be 0x08 for a TPE target). In that case, **PassiveInitiatorData [ ]** contains the ID of the target (4 bytes).
  - **212/424 kbps:**
    In that case, this field is mandatory in passive communication mode and contains the complete payload information that should be used in the polling request command (5 bytes, length byte is excluded) as defined in **Error! Reference source not found.**, §11.2.2.5.
- **NFCID3i** is the NFCID3 of the initiator that is used by the PN532 within the ATR_REQ. Depending on the baud rate specified and the communication mode, the use of this field is different:
  - **106/212/424 kbps, active mode:**
    The field is used to build the ATR_REQ frame. If not present, the PN532 will use a random value.
  - **106 kbps, passive mode:**
    The field is used to build the ATR_REQ frame. If not present, the PN532 will use a random value.

– **212/424 kbps, passive mode:**
This field is not used. The NFCID3i field of the ATR_REQ is filled with the value of the NFCID2t of the target received in the POL_RES frame (refer to process description in passive mode).

- **Gi** contains the general bytes for the ATR_REQ (optional, max. 48 bytes).

Output:

D5	57	Status	Tg	NFCID3t[0..9]	DIDt	BSt	BRt
					TO	PPt	[Gt [0..n]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p: 67),
- **Tg** is the logical number attributed to the activated target.
The target number returned within this output frame is always 0x01 (only one TPE target is supported).

The following parameters are part of the **ATR_RES** sent by the target:

- **NFCID3t[0..9]** is an array of bytes containing the random identifier of the target,
- **DIDt** is the DID byte sent by the target,
- **BSt** specifies the supported send-bit rate by the target,
- **BRt** specifies the supported receive-bit rate of the target,
- **TO** specifies the timeout value of the target in transport protocol,
- **PPt** specifies the optional parameters of the target (Length reduction, NAD usable and General bytes),
- **Gt [0..n]** are the optional general bytes (max. 47 bytes). They contain general information.

Syntax Error Conditions:

- Incorrect Baud Rate (**BR**),
- Incorrect **ActPass** parameter,
- Bad TSN (Time Slot Number) value in **PassiveInitiatorData**, in passive 212/424 kbps mode (different from 0x00, 0x01, 0x03, 0x07 or 0x0F),
- Incorrect command length.

Description:

The process is different depending on the communication mode (Active or Passive):

ACTIVE MODE

- Do Initial RFCA
- ATR_REQ
  - Set the communication settings (Active mode, baud rate **BR**)
  - Send ATR_REQ constructed with **NFCID3i [ ]** and **Gi [ ]**.
    Depending on the value of the internal parameter fDIDUsed (set by the host controller with **`SetParameters`** command (§7.2.9, p: 85)), the PN532 constructs the ATR_REQ with or without a DID parameter. If used, the DID value is fixed by the PN532 to 0x01.
  - Receive ATR_RES.
    The PN532 waits for this answer from the target a maximum time (timeout defined with the **`RFConfiguration`** command (§7.3.1, p: 101)). In case of incorrect ATR_RES received, the PN532 sends again ATR_REQ (MxRtyATR times)
- If a correct ATR_RES is received then the PN532 stores the NFCID3t and attributes a logical number for this new target (**Tg**). The target number returned within this output frame is always 0x01.
  The complete **ATR_RES** (CMD0 CMD1 = 0xD5 0x01 excepted) is sent back to the host controller

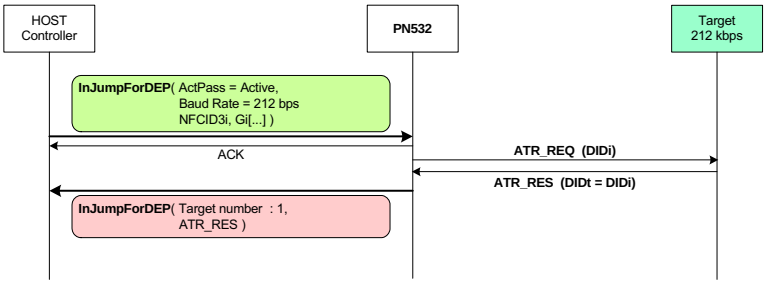


Fig 68. InJumpForDEP – Active communication mode – DID used

PASSIVE MODE

- Do Initial RFCA
- If BR = 106 kbps
  - Select one target (SENS_REQ, SDD_REQ, SEL_REQ)
    - If no Transport Protocol Equipped (TPE) target is detected, try again the process (SENS_REQ …). The absence of target is detected with a ~5 ms timeout value,
  - Store the NFCID1t for further use,
  - Send a ATR_REQ constructed with **NFCID3i [ ]** and **Gi [ ]**.
    Depending on the value of the internal parameter fDIDUsed, the ATR_REQ contains or not a DID parameter. If used, the DID value is fixed by the PN532 to 0x01.
- Else if BR = 212 or 424 kbps
  - Process a time slot SDD: send a POL_REQ with a **PassiveInitiatorData** given by the host controller. This POL_REQ command is sent under a timeout control which value depends on the Time Slot Number (see **`InListPassiveTarget`** §7.3.5, p: 115).
    - If a correct POL_RES answer is received, then store the NFCID2t for further use.
    - If no valid POL_RES is received in due time, try again the process (POL_REQ).
  - Send an ATR_REQ constructed with an overruled **NFCID3i [ ]** and **Gi [ ]**.
    Depending on the value of the internal parameter fDIDUsed, the ATR_REQ contains or not a DID parameter. If used, the DID value is fixed by the PN532 to 0x01.
    To allow selection between several targets, the NFCID3i field of the ATR_REQ is filled with the value of the NFCID2t of the target received in the POL_RES.
    The sizes of NFCID3i and NFCID2t are different, so following padding is used:

NFCID3i									
0	1	2	3	4	5	6	7	8	9
NFCID2t0	NFCID2t1	NFCID2t2	NFCID2t3	NFCID2t4	NFCID2t5	NFCID2t6	NFCID2t7	0x00 (padding)	0x00 (padding)

- Receive ATR_RES

  In case of success (no timeout), the target is Transport Protocol Equipped.
  In that case, the NFCID3t is memorized in the PN532 and the ATR_RES is sent back to the host controller (CMD0 CMD1 = 0xD5 0x01 excepted).
  The PN532 waits for this ATR_RES coming from the target a maximum time (timeout defined with the **`SetParameters`** command (§7.2.9, p: 85)).
  Twice ATR_REQ are sent by PN532, in case of incorrect ATR_RES received. The MxRtyATR parameter from the **`RFConfiguration`** command (§7.3.1, p: 101) is not take into account.
- If correct ATR_RES is received then the PN532 stores the NFCID3t and attributes a logical number for this new target (**Tg**).
  The target number returned within this output frame is always 0x01.

The following figure depicts the **InJumpForDEP** process in case of passive mode activation at 106 kbps.

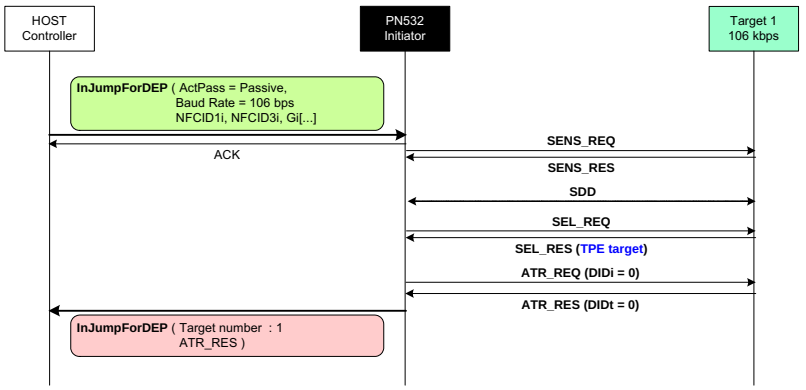


Fig 69. InJumpForDEP – Passive Communication Mode – DID not used

Remark: In any cases (active or passive mode), if this command is aborted by the host controller without any target activated, the RF field is automatically switched off to decrease power consumption (see chapter §3.1.3.10, p: 20).

7.3.4 InJumpForPSL

This command is used by a host controller to activate a target using either active or passive communication mode.
If a target is in the field, it will then be ready for PSL or DEP exchanges.

Input:

D4	46	ActPass	BR	Next	[PassiveInitiatorData] (4 or 5 bytes)
					[NFCID3i [0..9]]
					[Gi [0..n]]

- **ActPass** is the communication mode requested by the host controller
  - 0x00 : Passive mode
  - 0x01 : Active Mode
- **BR** is the Baud Rate to be used during the activation
  - 0x00 : 106 kbps
  - 0x01 : 212 kbps
  - 0x02 : 424 kbps
- **Next** indicates if the optional fields of the command (**PassiveInitiatorData**, **NFCID3i** and **Gi**) are present (bit = 1) or not (bit = 0).
  - bit 0 : PassiveInitiatorData is present in the command frame
  - bit 1 : NFCID3i is present in the command frame
  - bit 2 : Gi is present in the command frame
- **PassiveInitiatorData [ ]** is an array of data to be used during the initialization of the target in case of passive communication mode (**ActPass**). Depending on the Baud Rate specified, the content of this field is different:
  - **106 kbps:**
    The field is optional and is present only when the host controller wants to initialize a target with a known ID. In that case, **PassiveInitiatorData [ ]** contains the ID of the target (4 bytes).
  - **212/424 kbps:**
    In that case, this field is mandatory in passive communication mode and contains the complete payload information that should be used in the polling request command (5 bytes, length byte is excluded) as defined in **Error! Reference source not found.**, §11.2.2.5.
- **NFCID3i** is the NFCID3 of the initiator that is used by the PN532 within the ATR_REQ. Depending on the baud rate specified and the communication mode, the use of this field is different:
  - **106/212/424 kbps, active mode:**
    The field is used to build the ATR_REQ frame. If not present, the PN532 will use a random value.
  - **106 kbps, passive mode:**
    The field is used to build the ATR_REQ frame. If not present, the PN532 will use a random value.

– **212/424 kbps, passive mode:**
This field is not used. The NFCID3i field of the ATR_REQ is filled with the value of the NFCID2t of the target received in the POL_RES frame. Refer to **`InJumpForDEP`** (§7.3.3, p: 108).

- **Gi** contains the general bytes for the ATR_REQ (optional, max. 48 bytes).

Output:

D5	47	Status	Tg	NFCID3t[0..9]	DIDt	BSt	BRt
					TO	PPt	[Gt [0..n]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67),
- **Tg** is the logical number attributed to the activated target.
The target number returned within this output frame is always 0x01 (only one TPE target is supported).

The following parameters are part of the **ATR_RES** sent by the target:

- **NFCID3t[0..9]** is an array of bytes containing the random identifier of the target,
- **DIDt** is the DID byte sent by the target,
- **BSt** specifies the supported send-bit rate by the target,
- **BRt** specifies the supported receive-bit rate of the target,
- **TO** specifies the timeout value of the target in transport protocol,
- **PPt** specifies the optional parameters of the target (Length reduction, NAD usable and General bytes),
- **Gt [0..n]** are the optional general bytes. They contain general information.

Syntax Error Conditions:

- Incorrect Baud Rate (**BR**),
- Incorrect **ActPass** parameter,
- Bad TSN (Time Slot Number) value in **PassiveInitiatorData**, in passive 212/424 kbps mode (different from 0x00, 0x01, 0x03, 0x07 or 0x0F),
- Incorrect command length.

Description:

The process of this command is exactly the same than the one of **`InJumpForDEP`** (§7.3.3, p: 108).

7.3.5 InListPassiveTarget

The goal of this command is to detect as many targets (maximum **MaxTg**) as possible in passive mode.

Input:

D4	4A	MaxTg	BrTy	[InitiatorData[ ]]

- **MaxTg** is the maximum number of targets to be initialized by the PN532. The PN532 is capable of handling 2 targets maximum at once, so this field should not exceed 0x02. For Jewel card, only one target can be initialized.
- **BrTy** is the baud rate and the modulation type to be used during the initialization
  - 0x00 : 106 kbps type A (ISO/IEC14443 Type A),
  - 0x01 : 212 kbps (FeliCa polling),
  - 0x02 : 424 kbps (FeliCa polling),
  - 0x03 : 106 kbps type B (ISO/IEC14443-3B),
  - 0x04 : 106 kbps Innovision Jewel tag.
- **InitiatorData [ ]** is an array of data to be used during the initialization of the target(s). Depending on the Baud Rate specified, the content of this field is different:

– 106 kbps type A:

The field is optional and is present only when the host controller wants to initialize a target with a known UID.
In that case, **InitiatorData [ ]** contains the UID of the card (or part of it). **The UID must include the cascade tag CT if it is cascaded level 2 or 3**.

Cascade Level 1

UID1	UID2	UID3	UID4

Cascade Level 2

CT	UID1	UID2	UID3	UID4	UID5	UID6	UID7

Cascade Level 3

CT	UID1	UID2	UID3	CT	UID4	UID5	UID6	UID7	UID8	UID9	UID10

– **106 kbps type B[10]:**
In this case, **InitiatorData[ ]** is formatted as following:

AFI (1 byte)	[Polling Method]

- **AFI:**
The AFI (Application Family Identifier) parameter represents the type of application targeted by the PN532 and is used to pre-select the PICCs before the ATQB.
This field is mandatory.
- **Polling Method:**
This field is optional. It indicates the approach to be used in the ISO/IEC14443-3B initialization:

[10] This NXP IC is licensed under Innovatron's ISO/IEC 14443 Type B patent license.

- If bit 0 = 1: Probabilistic approach (option 1) in the ISO/IEC14443-3B initialization,
- If bit 0 = 0: Timeslot approach (option 2) in the ISO/IEC14443-3B initialization,
- If this field is absent, the timeslot approach will be used.

– **212/424 kbps:**
In that case, this field is mandatory and contains the complete payload information that should be used in the polling request command (5 bytes, length byte is excluded) as defined in **Error! Reference source not found.** §11.2.2.5.

– **106 kbps Innovision Jewel tag:**
This field is not used.

Output:

D5	4B	NbTg	[$TargetData_1$ [ ]]	[$TargetData_2$ [ ]]

- **NbTg** is the Number of initialized Targets (minimum 0, maximum 2 targets),
- **$TargetData_i$ [ ]** contains the information about the detected targets and depends on the baud rate selected. The following information is given for one target, it is repeated for each target initialized (**NbTg** times).

– **106 kbps Type A:**

Tg	SENS_RES[11] (2 bytes)	SEL_RES (1 byte)	NFCIDLength (1 byte)	NFCID1[ ] (NFCIDLength bytes)	[ATS[ ]] (ATSLength bytes[12])

– **106 kbps Type B:**

Tg	ATQB Response (12 bytes)	ATTRIB_RES Length (1 byte)	ATTRIB_RES[ ] (ATTRIB_RES Length)

***Remark**: This mode is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*

– **212/424 kbps:**

Tg	POL_RES length	0x01 (response code)	NFCID2t	Pad	SYST_CODE (optional)
1 byte	1 byte	1 byte	8 bytes	8 bytes	2 bytes
	POL_RES (18 or 20 bytes)				

[11] The first byte is the MSB, the second one the LSB.

[12] The first byte of the ATS frame sent by an ISO14443-4 card in response to a RATS is the length of the complete ATS (cf. **Error! Reference source not found.**). Thus here, the ATS structure is [ATSLength xx_1 xx_2 xx_3 xx_4 … $xx_{ATSLength-2}$ $xx_{ATSLength-1}$]

– **106 kbps Innovision Jewel tag:**

Tg	SENS_RES (2 bytes)	JEWELID[ ] (4 bytes)

Syntax Error Conditions:

- **MaxTg** value is incorrect (0 or higher than 2) for the targets 106 kbps type A, 106 kbps type B and for 212/424 kbps,
- **MaxTg** value is incorrect (0 or higher than 1) for the target 106 kbps Innovision Jewel tag,
- **BrTy** value is incorrect,
- TSN number in the **InitiatorData** is incorrect in passive 212/424 kbps mode (different from 0x00, 0x01, 0x03, 0x07 or 0x0F),
- AFI parameter missing for Type B,
- Incorrect command length.

Description:

Depending on the baud rate requested, the PN532 will use a Mifare, FeliCa, ISO/IEC14443-3B or Innovision Jewel initialization.

The default analog settings or those that have been modified by the host controller with the **`RFConfiguration`** command (CfgItem 0x0A, 0x0B, 0x0C, 0x0D) are used during the activation.

- **if BrTy = 0x00 (106 kbps Type A)**
  – As long as there is no target detected (maximum = **MaxTg**),
  (*This process is done MxRtyPassiveActivation times (§7.3.1, p: 101, CfgItem 0x05), if no answer is detected the command is terminated and the field **NbTg** in the output buffer contains 0x00, meaning that no target has been detected with the number of allowed trials).*
    o Probe the field for targets using either SENS_REQ or ALL_REQ command with timeout control of ~5 ms.
    The ALL_REQ command is sent if the **InitiatorData[ ]** input data contains a UID of a card. The ID includes the cascade tag CT if it is cascaded level 2 or 3.
    o If one of several target(s) has been detected with the previous command, handle the anti-collision (SDD_REQ) and then select one target (SEL_REQ command).
    o If the selection is successful, the PN532 attributes a logical number for the current target. This logical number will then be used by the host controller in all the target-related commands (**`InDataExchange`**, **`InATR`**, **`InPSL`**, **`InSelect`** …) to identify the target.
    The first target found when executing this command will have the number Tg=1 and the second one the number Tg=2.
    The information relative to previously initialized targets (stored inside the PN532) is lost.
    o Fill the **TargetData$_i$[ ]** output buffer with all the information relative to the target (Tg, SENS_RES, SEL_RES, length of the NFCID1t field, NFCID1t)
    o If the target indicates that it is ISO/IEC14443-4 compliant, then the PN532 carries out the ISO/IEC14443-4 activation, sending a RATS and waiting for a ATS response from the card.

In that case, the complete ATS response frame is sent back to the host controller in **TargetData$_i$ [ ]**.

- If more than one target is requested by the host controller (**MaxTg** input parameter), put the initialized target in HALT state (SLP_REQ) so that other targets can be initialized. In case of an ISO/IEC14443-4 target compliant, a S(deselect)REQ operation is performed instead of a SLP_REQ.

– The real number of initialized target is indicated to the host controller in the **NbTg** field (0 ≤ **NbTg** ≤ MaxTg).

– The latest target initialized remains active and is not put in HALT state. Thus, the host controller is able to exchange data with this target more quickly.

Remark 1: The NFCID1t does not include the cascade tag CT if it is cascaded level 2 or 3.

Remark 2: In case of multiple targets activation, when a collision is detected on the SENS_RES, the SENS_RES field in **TargetData$_i$ [ ]** is filled with value (0x00, 0x00).

- **if BrTy = 0x03 (106 kbps Type B)**

  ***Remark**: This mode is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*

  – As long as there is no target detected (maximum = **MaxTg**),
  (*This process is done MxRtyPassiveActivation times (§7.3.1, p: 101, CfgItem 5), if no answer is detected the command is terminated and the field **NbTg** in the output buffer contains 0x00, meaning that no target has been detected with the number of allowed trials).*

  - Probe the field for targets using either REQB or WUPB command with timeout control of ~5 ms.
    The WUPB command is sent if the **InitiatorData[ ]** input data indicates that the card is initially in HALT state.
  - If one of several target(s) have been detected with the previous command, handle the anti-collision using of REQB command if the probabilistic approach is used (ISO144443-3B option 1), SLOT_MARKER command in the timeslot approach (ISO/IEC14443-3B option 2) and then select one target (ATTRIB command).
  - If the selection is successful, the PN532 attributes a logical number for the current target. This logical number will then be used by the host controller in all the target-related commands (InDataExchange, InATR, InPSL, InSelect …) to identify the target.
    The first target found when executing this command will have the number Tg =1 and the second one the number Tg=2.
    The information relative to previously initialized targets (stored inside the PN532) is lost.
  - Fill the **TargetData [ ]** output buffer with all the information relative to the target (Tg, ATQB_RES, length of the ATTRIB_RES and the ATTRIB_RES.
  - If more than one target is requested by the host controller (**MaxTg** input parameter), put the initialized target in HALT state by the command S(deselect)REQ so that other targets can be initialized.

  – The real number of initialized target is indicated to the host controller in the **NbTg** field (0 ≤ **NbTg** ≤ MaxTg).

  – The latest target initialized remains active and is not put in HALT state. Thus, the host controller is able to exchange data with this target more quickly.

- **if BR = 0x01 or 0x02 (212 kbps or 424 kbps)**
  - As long as there is no target detected (with a maximum number MxRtyPassiveActivation of retries), process a time slot SDD: send a POL_REQ with the PolReqPayload information given by the host controller.
    This command is sent with a timeout control whose value depends on the Time Slot Number (TSN) chosen by the host controller in the **InitiatorData [ ]** field:
    TOvalue = Td + (TSN+1) x Ts
    TOvalue = 512 x 64/fc + (TSN+1) x 256 x 64/fc
    TOvalue = 2.42 ms + (TSN+1) x 1.21 ms
    *(This process is done MxRtyPassiveActivation times (§7.3.1, p: 101, CfgItem 5), if no answer is detected the command is terminated and the field **NbTg** in the output buffer contains 0x00, meaning that no target has been detected with the number of allowed trials).*
  - When one or several targets have answered to the polling request command, the PN532 checks the coherence of the answer(s):
    - The command byte in the polling response has to be equal to 1,
    - The PN532 has to receive 18 bytes or 20 bytes of polling response frame. The response frame length depends on POL_REQ type and Card model.

  All the targets that do not satisfy these conditions are rejected.

  If the POL_RES is correct, the PN532 attributes a logical number for the current target. This logical number will then be used by the host controller in all the target-related commands (**`InDataExchange`**, **`InATR`**, **`InPSL`**, **`InSelect`** …) to identify the target.

  The first target found when executing this command will have the number Tg=1 and the second one the number Tg=2.
  If previous targets were initialized previously, the information relative to these targets (stored inside the PN532) is lost.

Fill the **TargetData$_i$ [ ]** output buffer with the answers of the valid targets:

- 1 byte containing the logical number attributed (Tg),
- 1 byte indicating the length of the POL_RES (2 + IDm (NFCID2t) + Pad + [SystCode] ➔ 18 or 20 bytes),
- 1 response code byte, fixed value 0x01,
- 8 bytes for NFCID2t (IDm),
- 8 bytes for the Pad,
- 2 possible bytes for the System Code of the target when the polling response frame is 20 bytes long.

– The real number of initialized target is indicated to the host controller in the **NbTg** field (0 ≤ **NbTg** ≤ MaxTg).

- **If BrTy = 0x04 (106 kbps Innovision Jewel tag)**
  – As long as there is no target detected (maximum 1 target),
    - Probe the field for targets using the ATQA_REQ command with timeout control of ~5 ms,
    - If one target has been detected with the previous command, read the identification of the Jewel tag using the RID command,
    - If the reading of identification is successful, the PN532 attributes the logical number 1 for the current target. This logical number will then be used by the host controller in all the target-related commands (**`InDataExchange`**, **`InSelect`** …) to identify the target.
    - Fill the **TargetData$_i$ [ ]** output buffer with all the information relative to the target (ATQA_RES, RID_RES).

Remark: If this command is aborted by the host controller without any target activated, the RF field is automatically switched off to decrease power consumption (see chapter §3.1.3.10, p: 20).

Examples:

- In the first example, the host controller requires the initialization of one target at 106 kbps type A.

```
➔ D4 4A  01  00
➔ D5 4B  01  01  04 00  08  04  92 2E 58 32
```

  In the answer frame, it is indicated that one target has been initialized with the following parameters:

  – Logical Number **`01`**
  – SENS_RES **`04 00`**
  – SEL_RES **`08`**
  – NFCID1t length **`04`**
  – NFCID1t content **`92 2E 58 32`**

- In the second example, the host requires the initialization of one ISO/IEC14443-3B card with the default parameters (AFI = 0x00).

```
➔ D4 4A  01  03  00     (deterministic approach)
➔ D4 4A  01  03  00  01 (probabilistic approach)
➔ D5 4B  01  01  50  01 02 03 04 00 00 00 00 00 00 00 01 01
                 ←--------------- ATQB ------------->
```

  One target of 106 kbps type B is detected in the field and gives the following responses:

  - ATQB_RES[12] **50 01 02 03 04 00 00 00 00 00 00 00**
  - ATTRIB_RES length **01**
  - ATTRIB_RES **01**

- In the third example, the host controller requires the initialization of one target at 212 kbps with the POL_REQ payload 00 FF FF 01 00 (system code requested).

```
➔ D4 4A  01 01 00 FF FF 01 00
➔ D5 4B  01 01 14 01 01 01 06 01 67 02 A5 15
                        03 00 4B 02 4F 49 8A 8A  FF FF
```

  In the answer frame, it is indicated that one target has been initialized with the following parameters:

  - Logical number **01**
  - POL_RES length **14**
  - response code byte **01**
  - NFCID2t **01 01 06 01 67 02 A5 15**
  - PAD **03 00 4B 02 4F 49 8A 8A**
  - System Code **FF FF**

- In this fourth example, the host controller requires to detect and to initialize an Innovision Jewel tag

```
➔ D4 4A  01 04
➔ D5 4B  01 01 04 00 92 2E 58 32
```

  In the answer frame, it is indicated that one target has been initialized with the following parameters:

  - Logical Number **01**
  - ATQA_RES **04 00**
  - RID content **92 2E 58 32**

7.3.6 InATR

This command is used by a host controller to launch an activation of a target in case of passive mode.

Input:

D4	50	Tg	Next	[NFCID3i [0..9]]	[Gi [0..n]]

- **Tg** is the logical number of the relevant target,
- **Next** indicates if the optional fields of the command (**NFCID3i** and **Gi**) are present (bit = 1) or not (bit = 0).
  - Bit 0 : NFCID3i is present,
  - Bit 1 : Gi is present.
- **NFCID3i** is the NFCID3 of the initiator that is used by the PN532 within the ATR_REQ. Depending on the baud rate, the use of this field is different:
  - **106 kbps, passive mode:**
    The field is used to build the ATR_REQ frame. If not present, the PN532 will use a random value.
  - **212/424 kbps, passive mode:**
    This field is not used. The NFCID3i field of the ATR_REQ is filled with the value of the NFCID2t of the target received in the POL_RES frame. Refer to **`InJumpForDEP`** (§7.3.3, p: 108).
- **Gi** contains the general bytes for the ATR_REQ (optional, max. 48 bytes)

Output:

D5	51	Status	NFCID3t [0..9]	DIDt	BSt	BRt	TO	PPt	[Gt [0..n]]

- **Status** is a byte indicating if the process has been terminated successfully or not. (see §7.1, p:67)

  The following parameters are part of the **ATR_RES** sent by the target:
- **NFCID3t [0..9]** is an array of bytes containing the random identifier of the target,
- **DIDt** is the DID byte sent by the target,
- **BSt** specifies the supported send-bit rate by the target,
- **BRt** specifies the supported receive-bit rate of the target,
- **TO** specifies the timeout value of the target in transport protocol,
- **PPt** specifies the optional parameters of the target (Length reduction, NAD usable and General bytes),
- **Gt [0..n]** are the optional general bytes (max. 47 bytes).
  They contain general information.

Syntax Error Conditions:

- **Tg** and **Next** parameters are missing.

Description:

It is assumed that the target **Tg** has been initialized before (see command **`InListPassiveTarget`** §7.3.5, p: 115).
If the **Tg** number is unknown, the PN532 informs the host controller with a specific error code (**Status** = 0x27).

The Baud Rate and the modulation type defined for the target **Tg** in the former **`InListPassiveTarget`** command are re-used.

The following process is performed:

- the process of activation of the target is the same whatever the mode of activation or the baud rate are:
  - Set the communication settings (Passive mode, baud rate **BR**),
  - Depending of the baud rate:
    106 kbps: send ATR_REQ constructed by means of **NFCID3i** and **Gi** at **BR**. If the NFCID3i is not present, a random value is used.
    212/424 kbps: send ATR_REQ with the NFCID3i field filled with the value of the NFCID2t of the target received in the POL_RES frame during the initialization with 0x00 padding (two last bytes).

    Depending on the value of the internal parameter fDIDUsed (set by the host controller with the **`SetParameters`** command (§7.2.9, p: 85)), the PN532 constructs the ATR_REQ with or without a DID parameter. If used, the DID value is fixed by the PN532 to 0x01.
    If **Gi** is not present in the command, the PN532 constructs the ATR_REQ without **Gi** bytes.
  - Receive ATR_RES.
    The complete ATR_RES received from the target is returned back to the host controller in the **ATR_RES** field for further decision (CMD0 CMD1 = 0xD5 0x01 excepted).
    The NFCID3t is stored internally in the PN532, as part of the total information relative to the target number **Tg**.

    The reception of the ATR_RES is done under conditions of timeout. This timeout value is defined in the **`SetParameters`** command (§7.2.9, p: 85).
    If no valid ATR_RES is received (MxRtyATR attempts), the PN532 returns in the **Status** byte an error code.

    The PN532 waits for this ATR_RES coming from the target a maximum time (timeout defined with the **`SetParameters`** command (§7.2.9, p: 85)).
    In case of incorrect ATR_RES received, the PN532 sends again ATR_REQ (MxRtyATR times).

Example:

The **InATR** command may be used in a step-by-step process when the host controller wants to activate a target.

The following sequence:

```
Tg = InListPassiveTarget (1, 106)
InATR (Tg …)
InPSL (Tg …)
```

Is equivalent to

```
InJumpForDEP (Passive, 106)
```

7.3.7 InPSL

This command is used by a host controller to change the defined bit rates either with a TPE target or with a ISO/IEC14443-4 target.

Input:

D4	4E	Tg	BRit	BRti

- **Tg** is the logical number of the relevant target,
- **BRit** is the Baud Rate to be negotiated for communication from the initiator to the target
  - 0x00 : 106 kbps
  - 0x01 : 212 kbps
  - 0x02 : 424 kbps
- **BRti** is the Baud Rate to be negotiated for communication from the target to the initiator
  - 0x00 : 106 kbps
  - 0x01 : 212 kbps
  - 0x02 : 424 kbps

Output:

D5	4F	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p: 67).

Syntax Error Conditions:

- **BRit** value is incorrect,
- **BRti** value is incorrect,
- Incorrect command length (not equal to 5).

Description:

It is assumed that the target **Tg** has been activated before (see commands **`InListPassiveTarget`** (§7.3.5, p: 115), **`InATR`** (§7.3.6, p: 122) and **`InJumpForPSL`** (§7.3.4, p: 113)). If this is not the case, or if the **Tg** number is unknown, the PN532 informs the host controller with a specific error code (**Status** = 0x27).

In the case of a **TPE target**, a Parameter Selection is launched with the target **Tg**:

- Send a PSL_REQ (based on the parameters **BRti** and **BRit**).
  The FSL parameter of the PSL_REQ is fixed to 0x00, meaning that the maximum length of the Transport protocol field is 64 bytes.
- If a PSL_RES is correctly received (**MxRtyPSL** retries: §7.3.1, p: 101)
  - The PN532 takes internally into account the parameters changes,
  - The **Status** byte is set to 0x00 and sent back to the host controller.
- Else the PN532 gives up and answers to the host controller with a **status** byte different from 0x00.

In the case of a ISO/IEC14443**-4 card**, a Protocol and Parameter Selection (PPS) is launched with the target **Tg**:

- Send a PPS request based on the parameters **BRti** = **DRi** and **BRit** = **DSi**.
- If a PPS response is correctly received (**MxRtyPSL** retries: §7.3.1, p: 101)
  - The PN532 takes internally into account the parameters changes,
  - The **Status** byte is set to 0x00 and sent back to the host controller,
  - The register values contained in the RFConfiguration CfgItem 0x0D are then applied to the respective registers (CIU_RxThreshold, CIU_ModWidth and CIU_MifNFC) depending on the baud rate chosen.
- Else the PN532 gives up and answers to the host controller with a **Status** byte different from 0x00.

This command is only valid for Type A cards (not Type B).

Possible errors returned (Status byte):

- Negotiation already performed with the relevant target
  ➔ *Operation not allowed error code is returned (**Status** = 0x26)*
- The target is neither a TPE nor ISO/IEC14443-4
  ➔ *Operation not allowed error code is returned (**Status** = 0x26)*
- Unknown target number
  ➔ *Command not acceptable error code is returned (**Status** = 0x27)*
- InPSL command was sent to a Type B card
  ➔ *Command not acceptable error code is returned (**Status** = 0x27)*

7.3.8 InDataExchange

This command is used to support protocol data exchanges between the PN532 as initiator and a target.

Input:

D4	40	Tg	[DataOut [ ]]

- **Tg** is a byte containing the logical number of the relevant target.
  This byte contains also a *More Information* (MI) bit (bit 6) indicating, when set to 1, that the host controller wants to send more data that all the data contained in the **DataOut [ ]** array (see Chaining mechanism §7.4.5, p: 178). This bit is only valid for a TPE target.
- **DataOut** is an array of raw data (from 0 up to 262 bytes) to be sent to the target by the PN532 (see §7.4.7, p: 186).

Output:

D5	41	Status	[DataIn [ ]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p: 67).
  When either in DEP or in ISO/IEC14443-4 PCD mode, this byte indicates also if *NAD* is used and if the transfer of data is not completed with bit *More Information* (see §7.4.5, p: 178).
- **DataIn** is an array of raw data (from 0 up to 262 bytes) received by the PN532.

Syntax Error Conditions:

- In case of Mifare card:
  - **Cmd** value is incorrect,
  - Bad number of data for Authentication command (Data 0..9),
  - Bad number of data for 16 bytes writing command (Data 0..15),
  - Bad number of data for 4 bytes writing command (Data 0..3).
- In case of Jewel tag:
  - **Cmd** value is incorrect.

Description:

When using this command, it is assumed that a target has been first activated. The baud rate and the modulation type that have been chosen by using one of the 3 possible commands: **`InListPassiveTarget`**, **`InJumpForDEP`** or **`InJumpForPSL`**.

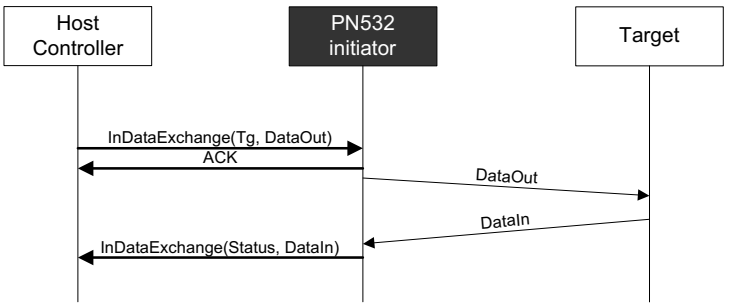


Fig 70. InDataExchange – General context

If the target number is unknown, the PN532 returns a specific error code (**Status** = *0x*27).

The PN532 has stored internally all the necessary information needed about all the initialized targets:

- Type (DEP, Mifare card, Jewel tag, ISO/IEC14443-4 card or FeliCa card),
- Baud rate, communication mode (active or passive),
- Internal state (selected, deselected, released …).

So, first of all, the PN532 applies all the correct configuration settings corresponding to the target **`Tg`** (Baud Rate, modulation type …).

Then, if the target **`Tg`** is not currently selected (current state memorized inside the PN532), the PN532 initiates the selection.
The detailed process of the selection depends on the type of the target; it is given in the **`InSelect`** (§7.3.12, p: 141) command description.

After this selection stage, the PN532 takes in charge the data exchange.

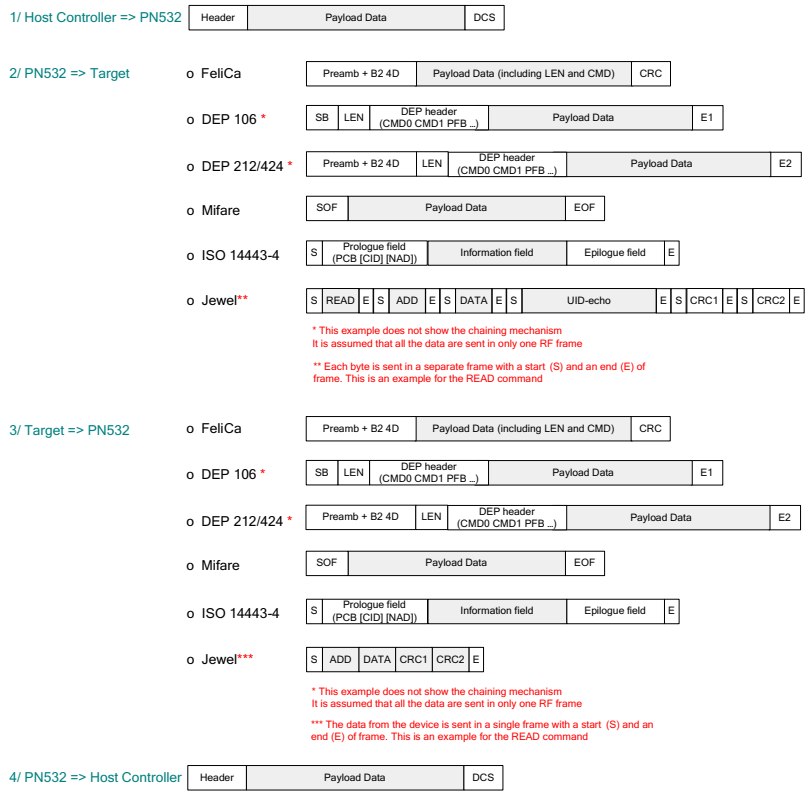


Fig 71. InDataExchange – Different target types

The way of exchanging data is different depending on the real nature of the target (DEP, Mifare card, Jewel tag, FeliCa card or ISO/IEC14443-4 card):

- **Mifare card**

When the target **Tg** is Mifare compliant, the input parameters are interpreted by the PN532 to execute a Mifare exchange. The PN532 sends the command and waits for the answer with a default timeout value of 51.2 ms.

This value can be changed by using the command **`RFConfiguration`** §7.3.1, p: 101.

The **DataOut [ ]** data must be formatted in the following way:

Cmd	Addr	[Data 0..15]

- **Cmd** is the Mifare specific command byte (see),
- **Addr** is the address associated with the Mifare command,
- **Data 0..15** is an array of maximum 16 bytes containing either
  - the data to be sent to the card during a writing operation,
  - or the data to be used during an authentication operation:
    - Data 0..5 contain the 6 bytes key,
    - Data 6..9 contain the 4 bytes serial number of the card.

The **DataIn [ ]** data are formatted in the same way:

[Data 0..15]

- **Data 0..15** is an array of maximum 16 bytes containing data read from the card in case of a reading command.

The Mifare specific command byte **Cmd** may take one of the possible values:

0x60 / 0x61	Authentication A / Authentication B
0x30	16 bytes reading
0xA0	16 bytes writing
0xA2	4 bytes writing
0xC1	Incrementation
0xC0	Decrementation
0xB0	Transfer
0xC2	Restore

Refer to Mifare cards (Classic and Ultralight) documentation and **Error! Reference source not found.** to have a more detailed description of the Mifare command set.

Examples:

It is assumed in these examples that the logical number attributed by the PN532 to the card is #01.

- **`D4 40 01 60 02 FF FF FF FF FF FF E2 3F B8 1E`**

Authenticate using the keys 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF to the address 0x02 of a Mifare Standard card whose UID number is 0xE2 0X3F 0xB8 0x1E.

- **`D4 40 01 30 02`**

Read 16 bytes from the address 0x02.

- **`D4 40 01 A0 02 01 02 03 04 05 06 07 08`**
  `09 0A 0B 0C 0D 0E 0F 10`

Write 16 bytes 0x01 to 0x10 from the address 0x02.

- **ISO/IEC14443-4 card**

When the target **Tg** is ISO/IEC14443-4 compliant, the input parameters are interpreted by the PN532 to execute a ISO/IEC14443-4 exchange.

The PN532 uses the data contained in the **DataOut [ ]** buffer to build the frames.

The main ISO/IEC14443-4 protocol mechanisms are implemented:

- Chaining,
- Waiting time Extension,
- Error handling.

The payload data returned by the target are sent back to the host controller in **DataIn [ ]**.

The C-APDU command length can be up to 261 bytes (CLA-INS-P1-P2-P3-255 data bytes-Le) and the R-APDU returned to the host controller can have a length of 258 bytes (256 data bytes-SW1-SW2).

Remark: Both **DataIn [ ]** and **DataOut [ ]** can contain NAD information. See **`SetParameters`** command §7.2.9, p:85 to have a complete description.

Example:

It is assumed in this example that the logical number attributed by the PN532 to the card is 0x01. The command sent to the card is a “read” command, 16 bytes are read.

```
➔ D4  40  01  00 B0 82 00 10
➔ D5  41  00  00 01 02 03 04 … 0F 90 00
```

The value of the status byte is 0x00, indicating that the RF exchange is correct.

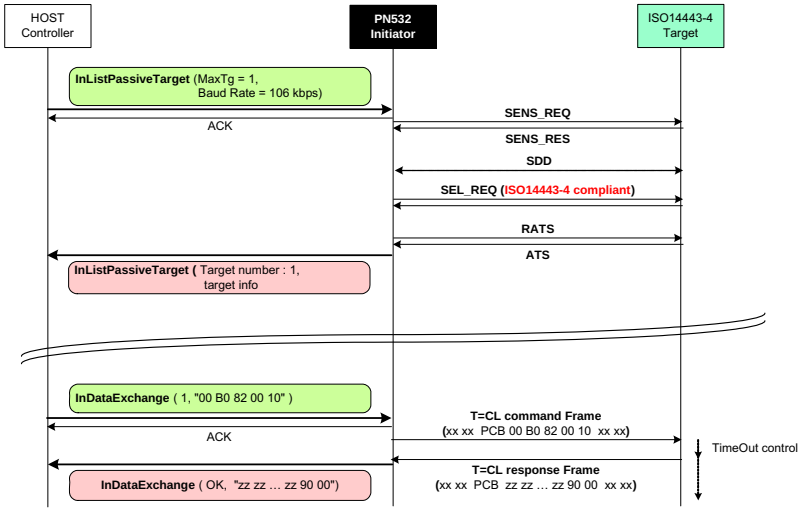


Fig 72. InDataExchange – Example of a ISO/IEC14443-4 exchange

During the exchanges, the following timeout control is used between the ISO/IEC14443-4 command and the response from the card:

$$TimeOut = FWT + 3*2^{FWI} \text{ etu}$$

Concerning the error handling, the PN532 tolerates up to 3 errors detected in the communication flow before returning an error code to the host controller. Also, a S(DESELECT) request is automatically send to fulfill ISO/IEC14443-4 standard.

- **FeliCa card**

When the target **Tg** is a FeliCa card, the PN532 just transfers the data contained in the **DataOut [ ]** buffer as they are.

The **Len** and **Cmd** bytes of the FeliCa protocol must be present in this buffer (the frame is completely built by the host controller).

Len	Cmd	[Data]

- **Len** is the length of the total **DataOut [ ]** buffer,
- **Cmd** is the FeliCa specific command byte,
- **Data** is an optional array of data bytes depending on the command used.

After having sent the command frame, the PN532 waits for a reply from the card and sends back the received frame to the host controller in **DataIn [ ]**.

A mute target can be detected by using a timeout mechanism after the transmitted frame (default value is 102.4 ms). The configuration of this timeout is done with the **`RFConfiguration`** command (§7.3.1, p:101), CfgItem 0x02 (fRetryTimeout) and 0x04 (MaxRtyCOM).

Examples:

It is assumed in this example that the logical number attributed by the PN532 to the card is 0x01. The card does an echo of the received frame.

➔ **`D4 40 01 06 F0 00 FF 11 22`**
➔ **`D5 41 00 06 F0 00 FF 11 22`**

The value of the status byte is 0x00, indicating that the RF exchange is correct.

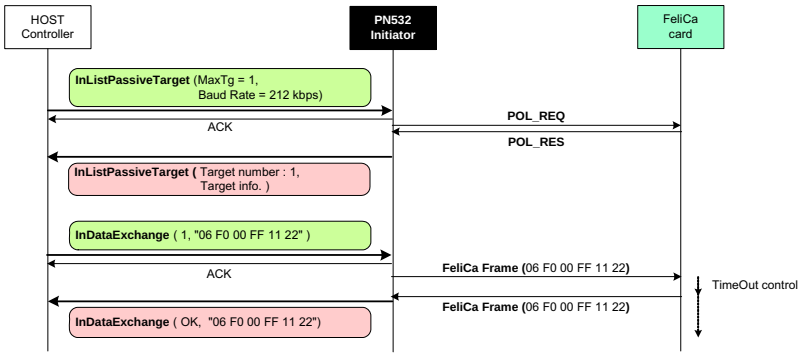


Fig 73. InDataExchange – Example of a FeliCa exchange

➔ **`D4 40 01 06 F0 00 FF 11 22`**
➔ **`D5 41 01`**

Here, the status byte informs of a timeout detected by the PN532.

- **DEP target**

When the target **Tg** is a NFC-DEP, the PN532 takes care of the protocol internally.

The PN532 sends the data contained in the **DataOut [ ]** array either in one or several stages (chaining mechanism as described in **Error! Reference source not found.** and §7.4.5, p:178) depending of the total length of the frame to send.

The PN532 uses a fixed value of Length Reduction of 64 bytes, even if the target indicates a higher capability.

The error handling and the timeout extensions ($S(TO)_{REQ}$ and $S(TO)_{RES}$) are also completely internally managed by the PN532.

If the *More Information* bit is not set in the **Tg** field of the host controller command, the PN532 waits for a DEP_RES. After having received a complete frame from the target, the PN532 sends back the data received in the **DataIn [ ]** array.

If the *More Information* bit is set in the **Tg** field of the host controller command, the PN532 returns no data to its host controller but takes care of the target with the timeout extensions request and response.

Remark: Both **DataIn [ ]** and **DataOut [ ]** can contain NAD information. See **`SetParameters`** command §7.2.9, p:85 to have a complete description.

Example:

It is assumed in this example that the logical number attributed by the PN532 to the target is #01. The target does an echo of the received frame.

➔ **`D4 40 01 11 22 33 44`**
➔ **`D5 41 00 11 22 33 44`**

The value of the status byte is 0x00, indicating that the RF exchange is correct.

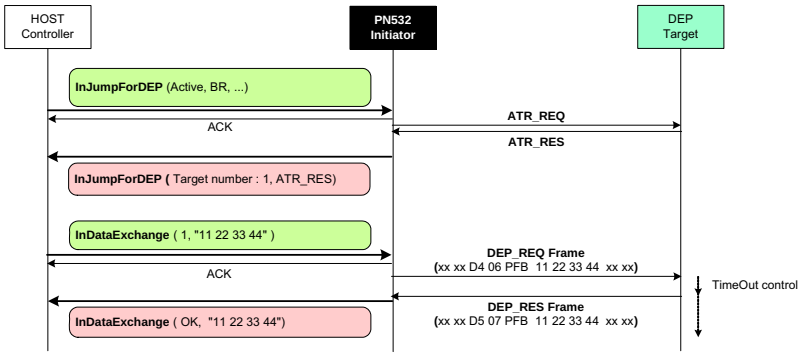


Fig 74. InDataExchange – Example of a DEP exchange

To simplify the figure, DID and NAD are not used in this example.

- **Innovision Jewel tag**

When the target is an Innovision Jewel tag, the input parameters are interpreted by the PN532 to execute a Jewel exchange. The PN532 sends the command and waits for the answer with a default timeout value of 51.2 ms.

This value can be changed by using the command **`RFConfiguration`** §7.3.1, p:101.

The **DataOut [ ]** data must be formatted in the following way:

Cmd	Addr	[Data1..8]

- **Cmd** is the Jewel specific command byte,
- **Addr** is the address associated with the Jewel command,
- **Data1..8** is an array of maximum 8 bytes containing the data to be sent to the card during a writing operation.

The **DataIn [ ]** data are formatted in the same way:

[Data1..255]

- **Data1..255** is an array of maximum 255 bytes containing data read from the card.

The Jewel specific command byte **Cmd** may take one of the possible values:

0x00	Read all bytes (maximum 255 bytes including HR, UID, data, LOCK and OTP bytes)
0x01	Read a single byte
0x10	Read segment (RSEG)
0x1A	Write-no-Erase a single byte
0x1C	Write-no-Erase 8 bytes
0x53	Write-with-Erase a single byte
0x55	Write-with-Erase 8 bytes
0x72	Read ID – Use to read the metal-mask ROM and UID0-3 from block 0

Refer to Jewel tag documentation to have a more detailed description of the Jewel command set and on the frames structure (7or 8 bit data and CRC).

Examples:

Read all bytes –read from 0x00 to 0x79 (there are 122 bytes on this tag)

```
➔  D4  40  01  00
➔  D5  41  00  01 3C CC 15 30 FF 01 01 25 00 12 …
                  AA 00 00 00 00 01 60 00 00 OO 00 00 00
```

Write-no-Erase a single byte 55 at the address 0x02 of block 0

```
➔  D4  40  01  1A 02 55
➔  D5  41  00
```

Write-with-Erase 8 bytes 0x01 …0x08 from the address 0x02 of block 0

```
➔  D4  40  01  55 02 01 02 03 04 05 06 07 08
➔  D5  41  00
```

7.3.9 InCommunicateThru

This command is used to support basic data exchanges between the PN532 and a target.

Input:

D4	42	[DataOut [ ]]

- **DataOut** is an array of raw data to be sent to the target by the PN532 (max. 264 bytes, cf. §7.4.7, p:186).

Output:

D5	43	Status	[DataIn [ ]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67),
- **DataIn** is an array of raw data received by the PN532 (coming from the target).

Description:

When using this command, it is assumed that a target has been first activated. The baud rate and the modulation type that have been chosen by a former **`InListPassiveTarget`** command (§7.3.5, p115) are used for transmitting the **DataOut [ ]** and for receiving the **DataIn [ ]** bytes.

This command is complementary of the **`InDataExchange`** command (§7.3.8, p:127). The main difference compared to **`InDataExchange`** is that here the PN532 does not handle with all the protocol features (chaining, error handling, …).

The host controller has to take care of the selection of the target it wants to reach (whereas when using the **`InDataExchange`** command, it is done automatically).

The process performed by this function is:

- Send the data by encapsulating the raw data (**DataOut [ ]**) in accordance with the current baud rate used. The CRC is automatically calculated and added by the PN532:

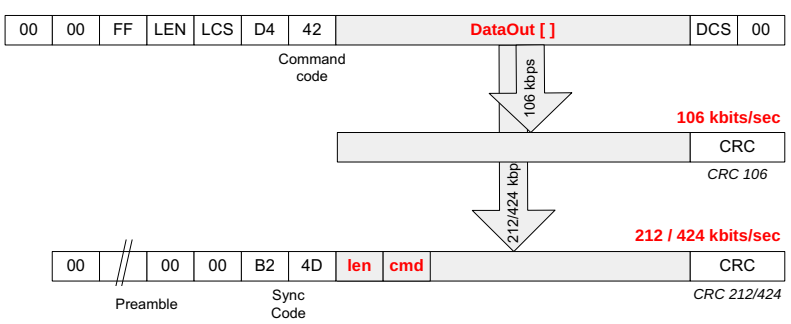


Fig 75. InCommunicateThru (1)

- Receive the data coming from the target in accordance with the current baud rate in use and de-encapsulate them (**DataIn [ ]**). The received CRC is checked but does not appear in **DataIn [ ]**:

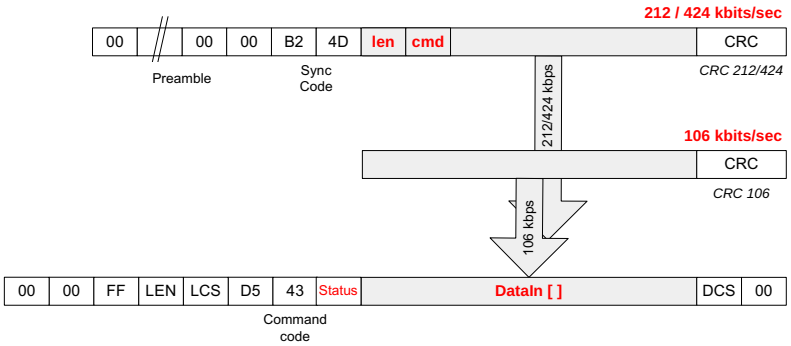


Fig 76. InCommunicateThru (2)

The following figure depicts the complete exchange of data between the host controller and the target.

In that case, the PN532 acts only as RF-transceiver between the host controller and the selected target.

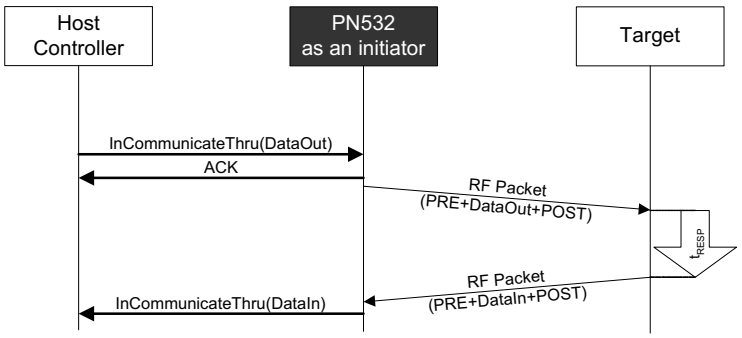


Fig 77. InCommunicateThru (3)

If the parameter **fRetryTimeout** of the command **`RFConfiguration`** (§7.3.1, p:101) is 0x00, no time out is managed on the delay (t_{RESP}) used by the target to send back its answer. The host controller has to manage timeout by itself.

Otherwise (**fRetryTimeout** is different from 0x00), the PN532 checks the response delay (t_{RESP}) to detect mute target (delay greater than **fRetryTimeout** parameter).

In case of error (either mute target or communication error), the PN532 sends again the RF packet to the target as many times as defined in the **MaxRtyCOM** parameter (cf. **`RFConfiguration`** command in §7.3.1, p:101).

In any case, the host controller can stop the current **`InCommunicateThru`** command by using one of the two dedicated ways of stopping: ACK frame or new command frame.

7.3.10 InDeselect

The goal of this command is to deselect the target(s) **Tg**. The PN532 keeps all the information relative to this target.

Input:

D4	44	Tg

- **Tg** is a byte containing the logical number of the relevant target (0x00 is a specific value indicating all targets).

Output:

D5	45	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67).

Description:

If the target is unknown (**Tg** number not attributed by the PN532) a specific error code is returned (**Status** = 0x27).

If the target is already deselected, no action is performed and **Status** OK is returned.

The process depends on the way that the target or the targets has or have been initialized.
In case of **Tg** equals to 0x00, this process is done for all the known targets.

Table 23. InDeselect RF actions

Target Type	Action
DEP compliant target (whatever the baud rate and the communication mode are)	Send **DSL_REQ**
106 kbps Type A card	Send **HLTA**
106 kbps Type B card ***Remark**: This target type is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*	Send **HLTB**
Mifare card	Send **HLTA**
ISO/IEC14443-4 compliant card	Send **DESELECT**
FeliCa card	No action
Innovision Jewel tag	No action

7.3.11 InRelease

The goal of this command is to release the target(s) **Tg**.

Input:

D4	52	Tg

- **Tg** is the logical number of the relevant target.
  (0x00 is a specific value indicating all available targets).

Output:

D5	53	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67).

Description:

Releasing a target means that the host controller has finished the communication with the target(s), so the PN532 erases all the information relative to it (them).

This command is used whatever the targets types and their current state (initialized, activated, deselected) are.

The process depends on the way that the target or the targets has or have been initialized.
In case of **Tg** equals to 0x00, this process is done for all the known targets.

Table 24. InRelease RF actions

Target Type	Action
DEP compliant target (whatever the baud rate and the communication mode are)	Send **RLS_REQ**
106 kbps Type A card	Send **HLTA**
106 kbps Type B card ***Remark**: This target type is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*	Send **HLTB**
Mifare card	Send **HLTA**
ISO/IEC14443-4 compliant card	Send **DESELECT**
FeliCa card	No action
Innovision Jewel tag	No action

In all the cases (DEP compliant or not), the logical numbers of the released targets are freed, meaning that no further data exchanges will be possible with the target(s).

If there is no more activated target after the release of the **Tg** one, the PN532 automatically returns in the ***Standby mode*** as defined in §0, p:12, meaning that the RF field is switched off and the CL frond end is put in low power mode.

7.3.12 InSelect

The goal of this command is to select the target **Tg**.

Input:

D4	54	Tg

- **Tg** is the logical number of the target to be selected.

Output:

D5	55	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67).

Description:

If this target is unknown (**Tg** number not attributed by the PN532) a specific error code is returned (**Status** = 0x27).

If the target is already selected, no action is performed and **Status** OK is returned.

The process depends on the way that the target or the targets has or have been initialized (see Table 25, p:141).

Table 25. InSelect RF actions

Target Type	Action
DEP compliant target Active communication mode (whatever the baud rate is)	• Wake up of the deselected target (**WUP_REQ**).
DEP compliant target Passive communication mode 106 kbps	• Initialization and Single Device Detection (**ALL_REQ**, **SEL_REQ**) using the *NFCID1t* of the target that has been stored during the initial activation of this target. • Send a **ATR_REQ** in which the *NFCID3i* is replaced by the *NFCID3t* of the target that has been stored during the initial activation of this target.
DEP compliant target Passive communication mode 212/424 kbps	• Initialization and Single Device Detection (**POL_REQ**). • Send a **ATR_REQ** in which the *NFCID3i* is replaced by the *NFCID2t* of the target that has been stored during the initial activation of this target (using padding as described in **`InJumpForDEP`**, §7.3.3, p:108).
106 kbps Type A card	• Initialization, anti-collision loop and Selection (**WUPA**, **Anti-collision** and **SELECT**) using the UID (*NFCID1* field of **InListPassiveTarget** see §7.3.5, p115) of the target that has been stored during the initial activation of this target.
106 kbps Type B card ***Remark**: This target type is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*	• Initialization (**WUPB**, **ATTRIB** commands) using N =1 and indicating the card is initially in HALT state.

Target Type	Action
Mifare card	• Initialization, anti-collision loop and Selection (**REQA**, **Anti-collision** and **SELECT**) using the UID (*NFCID1* field of **`InListPassiveTarget`** see §7.3.5, p115)
ISO/IEC14443-4 Type A compliant card	• Initialization, anti-collision loop and Selection (**WUPA**, **Anti-collision** and **SELECT**) using the UID (*NFCID1* field of **`InListPassiveTarget`** see §7.3.5, p115) of the target that has been stored during the first activation of this target. • Send an **RATS**.
ISO/IEC14443-4 Type B compliant card ***Remark**: This target type is accepted only with the PN532, which is the PN532's version supporting Type B PCD functionality.*	• Initialization (**WUPB**, **ATTRIB** commands) using N =1 and indicating the card is initially in HALT state.
FeliCa card	• Initialization and Single Device Detection (**POL_REQ**).
Innovision Jewel tag	• No action

Note: This command can be used in combination with **`InDeselect`** command (§7.3.10, p:139), as described in the following figure.
When using the **`InDataExchange`** command (§7.3.8, p:127), the Select/Deselect sequence is done automatically.

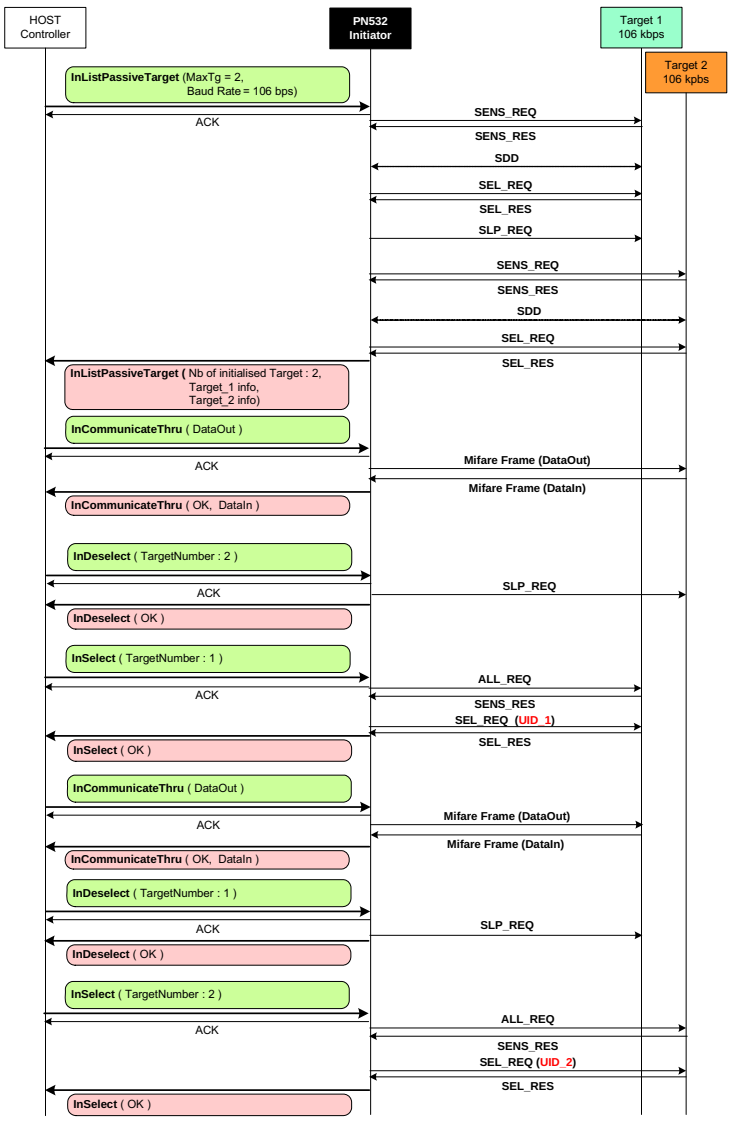


Fig 78. InSelect

7.3.13 InAutoPoll

This command is used to poll card(s) / target(s) of specified Type present in the RF field.

Input:

D4	60	PollNr	Period	Type 1	[Type 2]	…	[Type N]

- **PollNr** specifies the number of polling (one polling is a polling for each Type j).
  0x01 – 0xFE : 1 up to 254 polling
  0xFF : Endless polling.
- **Period** (0x01 – 0x0F) indicates the polling period in units of 150 ms,
- **Type 1** indicates the mandatory target type to be polled at the 1st time,
- **Type 2 to Type N** indicate the optional target types to be polled at the 2nd up to the Nth time (N ≤ 15).
  The format used for these fields is:

Act	DEP	TCL	Mf_Fe	0	BrMod		
7	6	5	4	3	2	1	0
Specific Types				Generic Types			

BrMod: Baudrate and modulation
0: 106 kbps ISO/IEC14443 type A,
1: 212 kbps,
2: 424 kbps,
3: 106 kbps ISO/IEC14443 type B,
4: Innovision Jewel tag,
Mf_Fe: Mifare or FeliCa card if set to '1',
TCL : ISO/IEC14443-4 compliant if set to '1',
DEP : DEP if set to '1',
Act : active mode if set to '1', passive mode if set to '0'.

The possible values are listed below:
0x00 : Generic passive 106 kbps (ISO/IEC14443-4A, Mifare and DEP),
0x01 : Generic passive 212 kbps (FeliCa and DEP)
0x02 : Generic passive 424 kbps (FeliCa and DEP),
0x03 : Passive 106 kbps ISO/IEC14443-4B,
0x04 : Innovision Jewel tag,

0x10 : Mifare card,
0x11 : FeliCa 212 kbps card,
0x12 : FeliCa 424 kbps card,
0x20 : Passive 106 kbps ISO/IEC14443-4A,
0x23 : Passive 106 kbps ISO/IEC14443-4B,

0x40 : DEP passive 106 kbps,
0x41 : DEP passive 212 kbps,
0x42 : DEP passive 424 kbps,

0x80 : DEP active 106 kbps,
0x81 : DEP active 212 kbps,
0x82 : DEP active 424 kbps.

Output:

D5	61	NbTg	[Type 1]	[Ln 1]	[AutoPollTargetData 1[ ]]
			[Type 2]	[Ln 2]	[AutoPollTargetData 2[ ]]

The response is variable depending on the found targets and on their type.

- **NbTg** is the number of target(s) found (maximum is two targets, only one of them can be DEP compliant),
- **Type 1 – Type 2** indicates the polled target type if any.
  The format of the output **Type** field is the same as the one of the input **Type** field. All the values listed above are possible except the generics (0x00, 0x01, and 0x02).
- **Ln 1 – Ln 2** indicates the length of the following/corresponding **AutoPollTargetData i[ ]** ($i \in 1 .. 2$),
- **AutoPollTargetData i[ ]** contains information about the i^{th} found target ($i \in 1.. 2$). Its number and its features are given as follow:
  - For non-DEP targets, the array **AutoPollTargetData i[ ]** is the same as the response of the **`InListPassiveTarget`** command (§7.3.5, p115):

TargetData i[ ]

  - For DEP compliant passive targets, **AutoPollTargetData i[ ]** is a concatenation of the response of the **`InListPassiveTarget`** command (§7.3.5, p115) and the one of the **`InATR`** command (§7.3.6, p:122):

TargetData i[ ]	NFCID3t[0..9]	DIDt	BSt	BRt	TO	PPt	[Gt[0..n]]

  - For DEP compliant active targets, **AutoPollTargetData i [ ]** is the response of the **`InATR`** command (§7.3.6, p:122):

NFCID3t [0..9]	DIDt	BSt	BRt	TO	PPt	[Gt [0..n]]

Syntax Error Conditions:

- The **PollNr**, **Period**, **Type 1** parameters are missing,
- Two out of the **Act**, **DEP**, **TCL** and **Mf_Fe** bits are set (Type value is incorrect),
- **Period** value is equal to zero,
- The number of polling (**PollNr**) is zero,
- Incorrect command length.

Remark 1: As the polling command is fixed for FeliCa card within **`InAutoPoll`** command (Command Code = 0x00, System Code = 0xFFFF, Reserved = 0x00 and TimeSlot = 0x00; see FeliCa card user's Manual for more details), only one FeliCa card can be polled.

Remark 2: If two targets are found, put the first initialized target in HALT or SLEEP state (depending on the target type) so that a second target can be initialized.
The latest target initialized remains active. Thus, the host controller is able to exchange data with this target more quickly.

Description:

This command is used to start an auto-polling process with a number of polling PollNr, an interval time of Period * 150 ms and a list of target types Type j (j ∈ 1 .. 15). The polling process is as follows (Fig 79):

1. Check **Type** validity
2. Check external RF field,
3. Initialize **j** = 1 to poll for the first **Type**,
4. Start timer for **Period** * **150 ms**,
5. Poll for targets according to the target type specified in the field **Type j** (j ∈ 1 .. 15). If a generic type is requested (e.g. **Type** = 0x00 => all passive targets 106 kbps type A), all the possible types (i.e. Mifare, ISO/IEC14443-4 or DEP passive 106 kbps) will be polled. According to the used protocols, all targets response time should be less than 150 ms, which is the minimum start value for the timer. That means if target(s) is(are) present, the answer is received before the timer end.
6. If target(s) is(are) found during the response time, exit the polling process and build the response with:
   a. The number **NbTg** of found target,
   b. The corresponding **Type**, **Ln** and **AutoPollTargetData i [ ]** fields for each found target,
7. If no target is found, switch off the RF field and wait for timer end.
8. If the next **Type j** field is present then increment **j** and process again the sequences 4 up to 8 to poll for the next **Type**. Otherwise, all the **Type j** (j ∈ 1 .. 15) fields have been scanned,
9. Decrement **PollNr** and execute again the sequences 3 up to 9 for a new polling process. If the number **PollNr** of polling is run out, exit the polling process and build the response with **NbTg** = 0x00.

This process is over when:

- Targets are found at the end of sequence 6. In this case, the response containing information on the targets is sent back to the host controller,
- The number **PollNr** of polling is run out at the end of sequence 9. A response with **NbTg** = 0x00 is returned,
- A new command is entered.

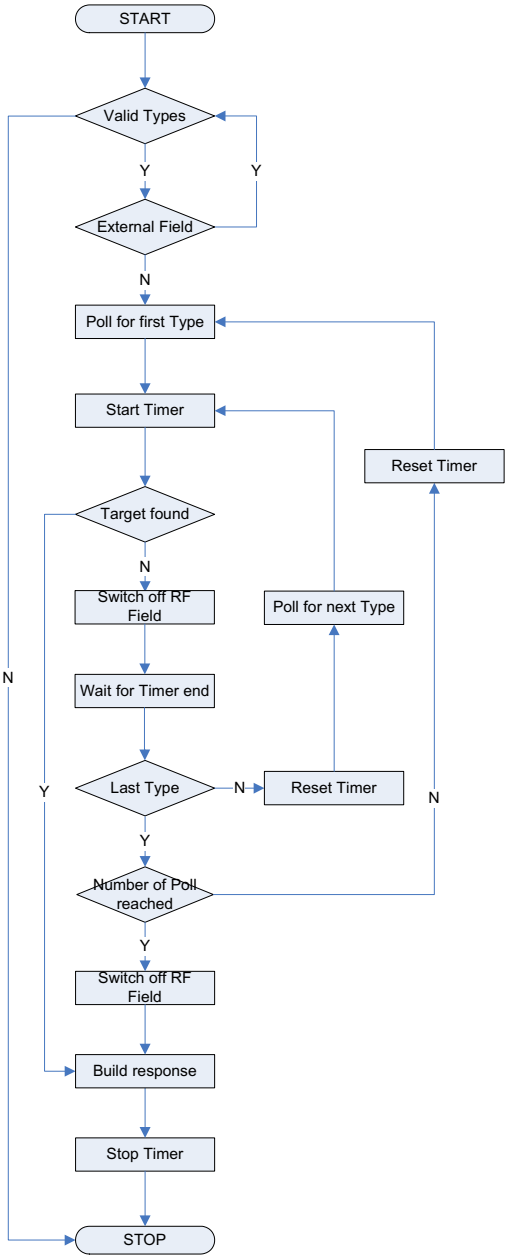


Fig 79. Auto-polling process

The Fig 80 shows the activation times during an auto-polling process. It shows that the activation time t is always less than the polling period T. After the activation time has been reached, the PN532 switch off its RF field until the polling period is elapsed.

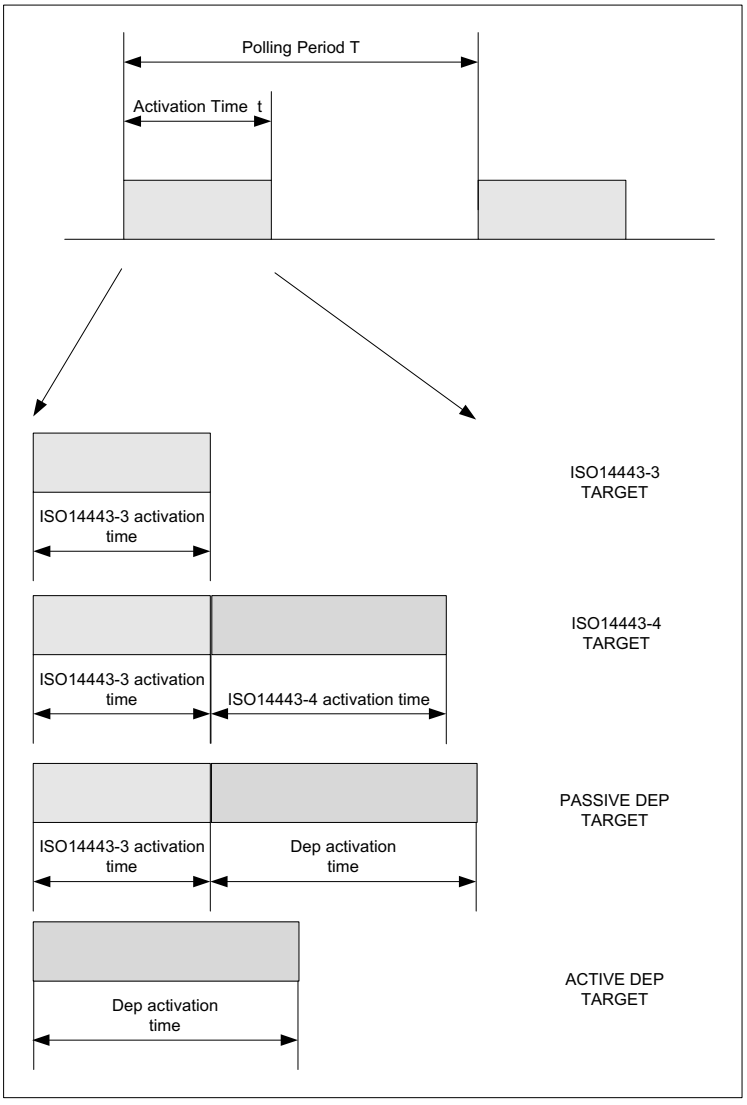


Fig 80. Some target activation times during an auto-polling process

Examples:

In this first example, the host controller polls for 212 kbps targets. One FeliCa 212 kbps card is present.

```
➔ D4 60  01  01  01
➔ D5 61  01  11  13  01  12  01   01 01 06 01 46 05 8F 1A
                                  04 01 4B 02 4F 49 93 FF
```

The answer frame indicates that one target has been detected and initialized with the following parameters:

- NbTg **01**
- Type 1 (FeliCa) **11**
- Ln 1 **13**
- AutoPollTargetData 1:
  - Logical Number **01**
  - POL_RES **12**
  - Response Code **01**
  - NFCID2t **01 01 06 01 46 05 8F 1A**
  - Pad **04 01 4B 02 4F 49 93 FF**

In this second example, the host controller polls for 106 kbps targets. One Mifare card and a DEP compliant card are present.

```
➔ D4 60  01   01  00
➔ D5 61  02   10  09  01  04 00  08  04  92 2E 58 32
              40  18  02  00 08  40  04  08 12 34 56
              00 11 22 33 44 55 66 77 88 99  00  00  00  09 01
```

The answer frame indicates that two targets have been detected and initialized with the following parameters:

- NbTg **02**
- Type 1 (Mifare) **10**
- Ln 1 **09**
- AutoPollTargetData 1:
  - Logical Number **01**
  - SENS_RES **04 00**
  - SEL_RES **08**
  - NFCID1t length **04**
  - NFCID1t **92 2E 58 32**
- Type 2 (DEP) **40**
- Ln 2 **18**
- AutoPollTargetData 2:
  - Logical Number **02**
  - SENS_RES **00 08**
  - SEL_RES **40**
  - NFCID1t length **04**
  - NFCID1t **08 12 34 56**
  - NFCID3t **00 11 22 33 44 55 66 77 88 99**
  - DIDt **00**
  - BSt **00**
  - BRt **00**
  - TO **09**
  - PPt **01**

7.3.14 TgInitAsTarget

The host controller uses this command to configure the PN532 as target.

Input:

D4	8C	Mode	MifareParams[ ] (6 bytes)		FeliCaParams[ ] (18 bytes)		NFCID3t (10 bytes)
				LEN Gt	[Gt[0..n]]	LEN Tk	[Tk[0..n]]

- **Mode** is a byte indicating which mode the PN532 should respect

7	6	5	4	3	2	1	0
Nu	nu	nu	nu	nu	PICC only	DEP only	Passive only
					0: no 1: yes	0: no 1: yes	0: no 1: yes

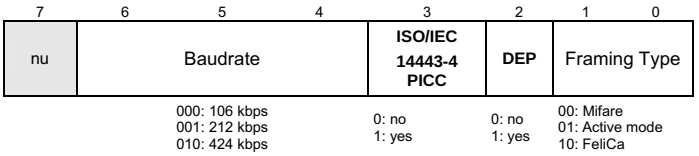
  - *PassiveOnly* flag is used to configure the PN532 to accept to be initialized only in passive mode, i.e. to refuse active communication mode;
  - *DEPOnly* flag is used to configure the PN532 to accept to be initialized only as DEP target, i.e. receiving an ATR_REQ frame. The PN532 can be activated either in passive or active mode, but if the PN532 receives a proprietary command frame as first command following AutoColl process, it will be rejected and the PN532 returns automatically in the AutoColl state;
  - *PICCOnly* flag is used to configure the PN532 to accept to be initialized only as ISO/IEC14443-4 PICC, i.e. receiving an RATS frame.
    If the PN532 receives another command frame as first command following AutoColl process, it will be rejected and the PN532 returns automatically in the AutoColl state.
- **MifareParams[ ]** is the information needed to be able to be activated at 106 kbps in passive mode. **MifareParams[ ]** is composed of:
  - *SENS_RES (2 bytes LSB first, as defined in ISO/IEC14443-3).*
  - *NFCID1t has a fixed length of 3 bytes containing the nfcid11 to nfcid13 bytes. Indeed, the PN532 can handle only NFCID1t in single size,*
  - *SEL_RES (1 byte), typical value = 0x40 (for DEP)*
    = 0x20 (for ISO/IEC14443-4 PICC emulation)
    = 0x60 (for both DEP and emulation of ISO/IEC14443-4 PICC)
- **FeliCaParams[ ]** contain the information to be able to respond to a polling request at 212/424 kbps in passive mode. **FeliCaParams[ ]** is composed of:
  - *NFCID2t (8 bytes),*
  - *PAD (8 bytes),*
  - *System Code (2 bytes), these two bytes are returned in the POL_RES frame if the 4th byte of the incoming POL_REQ command frame is 0x01.*
- **NFCID3t** is used in the ATR_RES in case of ATR_REQ received from the initiator,
- **LEN Gt** codes the number of general bytes (max. 47 bytes). This field is mandatory. When set to 0, there are no general bytes following,
- **Gt[ ]** is an array containing the general bytes to be used in the ATR_RES. This information is optional and the length is not fixed (max. 47 bytes),

- **LEN Tk** codes the number of historical bytes (max. 48 bytes). This field is mandatory. When set to 0, there are no historical bytes following,
- **Tk[ ]** is an array containing the historical bytes to be used in the ATS when PN532 is in ISO/IEC14443-4 PICC emulation mode. This information is optional.

Output:

D5	8D	Mode	InitiatorCommand[ ]

- **Mode** is a byte indicating in which mode the PN532 has been activated:



- **InitiatorCommand** is an array containing the first valid frame received by the PN532 once the PN532 has been initialized.
  This frame is different depending on the mode in which the PN532 has been initialized (ISO/IEC14443-4 PICC emulation, DEP, passive 106 kbps or 212/424 kbps).

Syntax Error Conditions:

- ***LEN Gt*** exceeds 47 bytes,
- ***LEN Tk*** exceeds 48 bytes,
- Both ***PICCOnly*** and ***DEPOnly*** are set to 1,
- ***PICCOnly*** is set to 1 whereas fISO14443-4_PICC is set to 0 (by using the **`SetParameters`** command (§7.2.9, p:85)),
- Incorrect command length.

Description:

When this command is used by the host controller, the PN532 first stores the input parameters in the dedicated area of the internal CIU and then activates the AutoColl command.
This AutoColl command handles FeliCa polling and Mifare anti-collision automatically.

Thus, **`TgInitAsTarget`** is ended when a complete command frame has been received from the external initiator. Depending on the initialization type, the 3 following scenarios are possible:

- 106 kbps passive (BR=106, Framing=Mifare):
  - When a SENS_REQ command is detected, the PN532 sends back the SENS_RES contained in **MifareParams**,
  - Then the PN532 uses the NFCID1t part of **MifareParams** during the anti-collision process,
  - At the end of the selection, the PN532 sends the SEL_RES to the initiator,
  - Then the PN532 waits for a command coming from the initiator that closes the AutoColl internal command,
  - This command may be an ATR_REQ, a SLP_REQ, a RATS or a proprietary command.
    - *ATR_REQ*: if the flag *fAutomaticATR_RES* is set (§7.2.9, p:85), the PN532 sends back automatically the ATR_RES frame to the initiator (example in Fig 81).
      Otherwise, the ATR_RES will be sent back to the initiator only after having received a **`TgSetGeneralBytes`** (§7.3.15, p:158) from the host controller.
      The complete ATR_REQ is returned to the host controller.

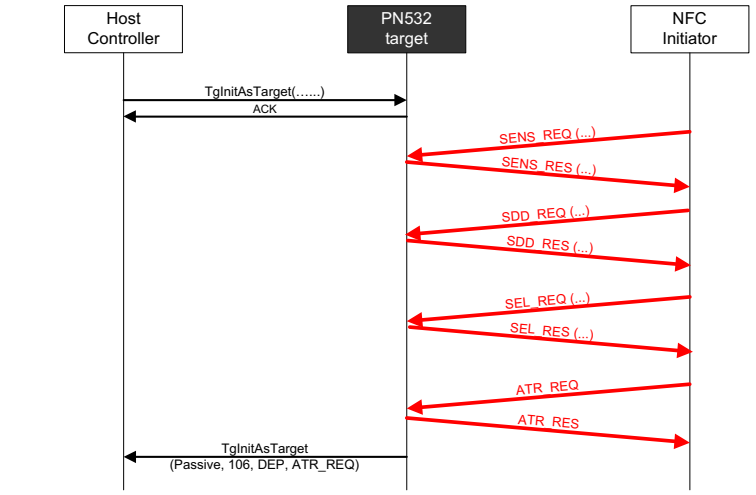


Fig 81. TgInitAsTarget – Passive DEP 106 kbps

- *SLP_REQ:* the PN532 starts again an AutoColl sequence and therefore is ready to receive a new activation command (**`TgInitAsTarget`** process is still running).

- *RATS:* if the flag fISO14443-4_PICC is set (See **`SetParameters`** command (§7.2.9, p:85), the PN532 responds with a predefined ATS containing historical bytes Tk if available (whatever the **MifareParams** SEL_RES byte value is).
  The *RATS* is returned to the Host Controller.

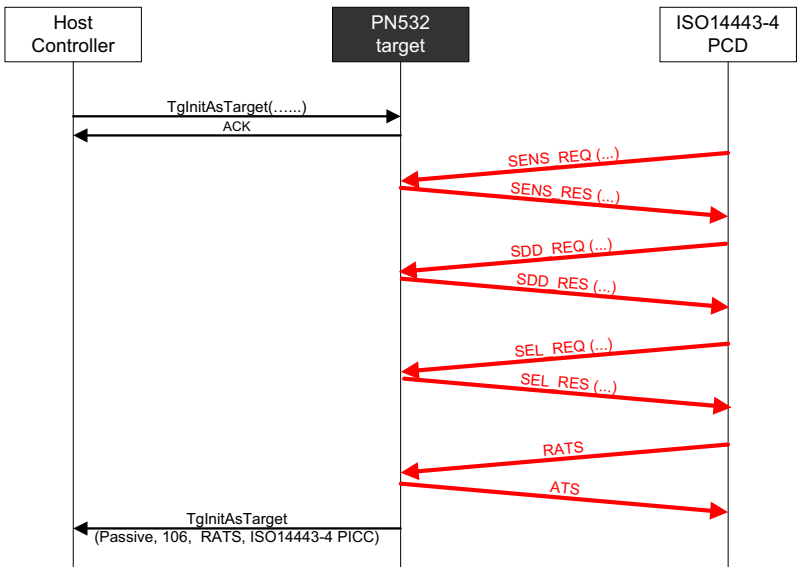


Fig 82. TgInitAsTarget – ISO/IEC14443-4 PICC emulated

- *proprietary command:* the PN532 does nothing with this command (example in Fig 83).
  If the bit *DEPOnly* is not set, the complete proprietary command is returned to the host controller.
  If the bit *DEPOnly* is set, the command is refused and the PN532 starts a new AutoColl sequence.

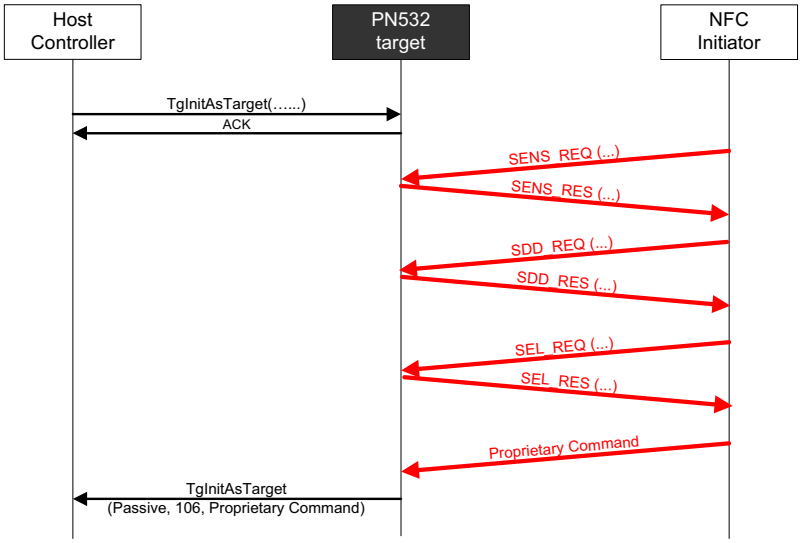


Fig 83. TgInitAsTarget – Proprietary command

- 212/424 kbps passive (BR=212/424, Framing=FeliCa):
  - When a POL_REQ command is detected, the PN532 sends back the POL_RES contained in **FeliCaParams**.
    If requested by the initiator (4th byte of the POL_REQ = 0x01), the system code information is added in the POL_RES.
  - Then the PN532 waits for a command coming from the initiator that closes the AutoColl process.
  - This command may be an ATR_REQ (Fig 84) or a proprietary command (Fig 85). In case of the reception of an ATR_REQ, the NFCID3i **must** be the specified NFCID2t with 0x00 padding (last two bytes):
    - If not, the PN532 rejects the command and starts a new AutoColl sequence,
    - If yes, the PN532 sends automatically the ATR_RES frame (except if the flag *fAutomaticATR_RES* is not set (§7.2.9, p:85); this is the case in Fig 84).

If the bit *DEPOnly* is set, the command received must be an ATR_REQ.
The PN532 refuses all other commands and starts a new AutoColl sequence.

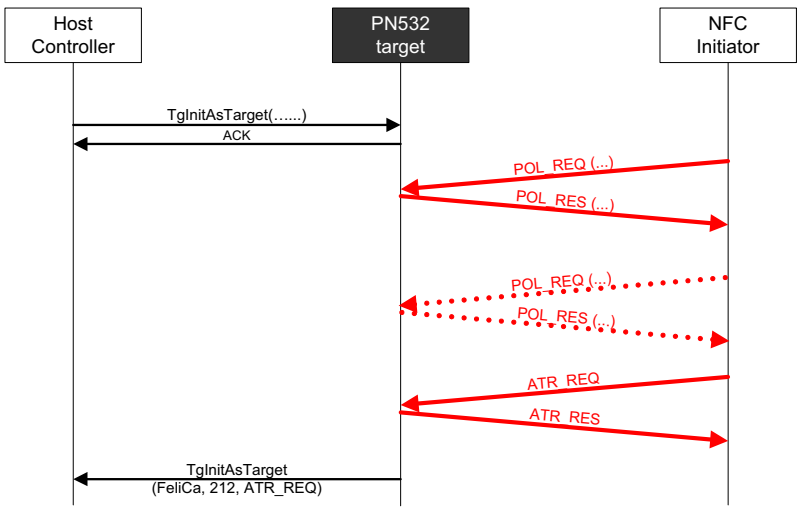


Fig 84. TgInitAsTarget – Passive DEP 212/424 kbps

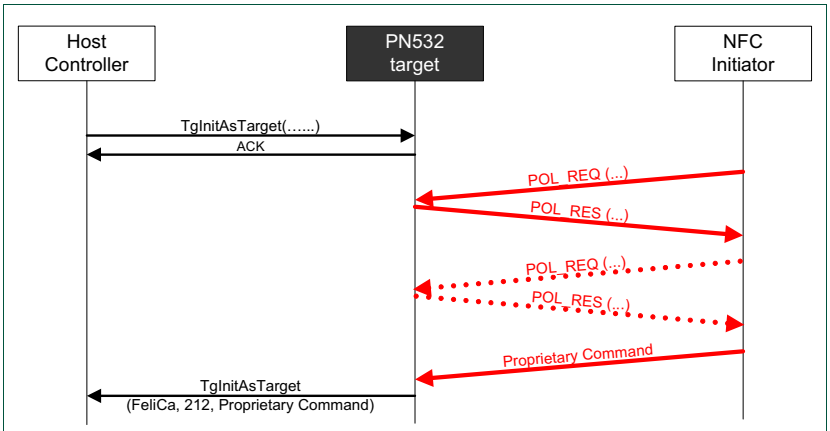


Fig 85. TgInitAsTarget – Passive 212/424 kbps – Proprietary command

- 106/212/424 kbps active (BR=106/212/424, Framing=Active mode):
  - The PN532 waits for a command coming from the initiator that closes the AutoColl process (at this stage, the baud rate and the communication mode are now determined and sent back to the host controller within **Mode** parameter);
  - The command received should be an ATR_REQ. If the incoming RF frame does not fit with ATR_REQ, the PN532 starts a new sequence of AutoColl;
  - Depending on the flag *fAutomaticATR_RES* (§7.2.9, p:85), the PN532 sends automatically the ATR_RES frame (Fig 86) or not. If not, the host controller shall use the **`TgSetGeneralBytes`** (§7.3.15, p:158);
  - The *PassiveOnly* bit shall not be set.

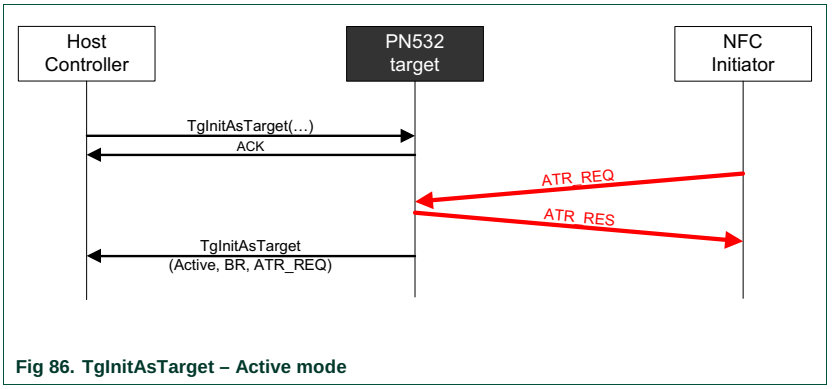


Fig 86. TgInitAsTarget – Active mode

Once the PN532 is configured as target, it can handle some of the DEP and ISO/IEC14443-4 commands without any help of its host controller.
The PN532 builds the corresponding answer frame and updates its internal state (released, deselected, activated, …). This is the case for the following command frame:

Table 26. Target configuration – Automatic response

Command Received	Automatic Response	DEP mode	ISO/IEC14443-4 PICC mode
ATR_REQ	ATR_RES may need **`TgSetGeneralBytes`** if *fAutomaticATR_RES* is not set	Y	
PSL_REQ	PSL_RES	Y	
DSL_REQ	DSL_RES	Y	
RLS_REQ	RLS_RES	Y	
WUP_REQ	WUP_RES	Y	
PPS request	PPS response		Y
S(DESELECT) request	S(DESELECT) response		Y

7.3.15 TgSetGeneralBytes

This command is used in combination with the **`TgInitAsTarget`** command (§7.3.14, p:151) to give the General Bytes.

The PN532 uses them to build the ATR_RES sent to the initiator.

Input:

D4	92	Gt[0..n]

- **Gt[ ]** is an array containing the general bytes to be used in the ATR_RES. The length of this field is not fixed (max. 47 bytes).

Output:

D5	93	Status

- **Status** is a byte indicating if the process has been terminated successfully or not
(see §7.1, p:67).

Syntax Error Conditions:

- **Gt[ ]** length exceeds 47 bytes,
- The PN532 is currently activated as ISO/IEC14443-4 PICC, and therefore this command is not supported.

Description:

By default (flag *fAutomaticATR_RES* set, §7.2.9, p:85), the PN532 uses the general bytes given in **`TgInitAsTarget`** command (present or not in the input parameters).

The command **`TgSetGeneralBytes`** allows the host controller to build the General Bytes of the target after having analyzed the ATR_REQ coming from the initiator.

When used, the command **`TgSetGeneralBytes`** must follow the **`TgInitAsTarget`**, as described in the following figure (Fig 87).

The PN532 does not send ATR_RES before the command **`TgSetGeneralBytes`**. Then, the PN532 prepares the ATR_RES with the **Gt[ ]** bytes and sends the complete ATR_RES to the initiator.

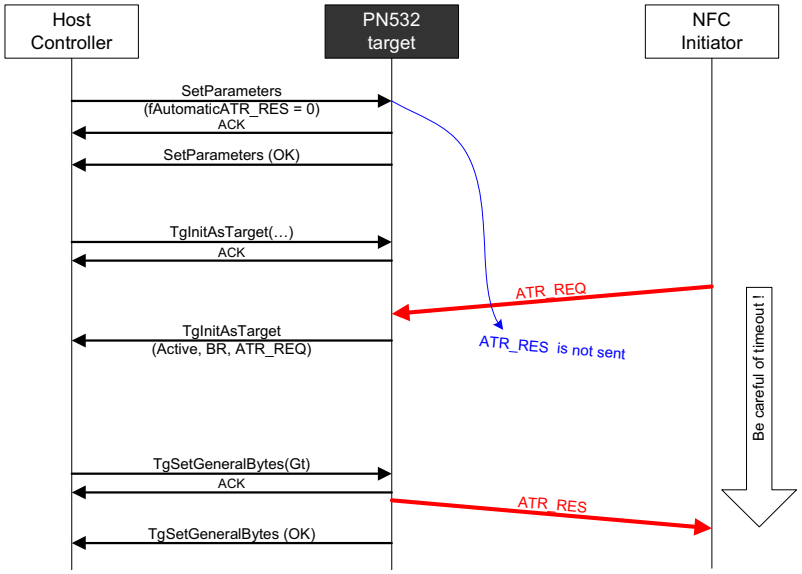


Fig 87. TgSetGeneralBytes

Remark: The NFC initiator controls a timeout after having sent an ATR_REQ (see **Error! Reference source not found.**), so the host controller of the PN532 must take care of that, meaning that if the ATR_RES is not ready in time, the initiator will stop the transaction with the target.

7.3.16 TgGetData

This command is used in case of the PN532 configured as target for Data Exchange Protocol (DEP) or for ISO/IEC14443-4 protocol when PN532 is activated in ISO/IEC14443-4 PICC emulated (see §4, p:21).

Input:

D4	86

Output:

D5	87	Status	[DataIn [ ]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67).
  When in DEP mode, this byte indicates also if *NAD* is used and if the transfer of data is not completed with bit *More Information* (see §7.4.5, p:178).
- **DataIn[ ]** is an array of data (from 0 up to 262 bytes) received by the PN532 coming from the initiator (see §7.4.7, p:186).

Description:

This command allows the host controller to get back the data received by the PN532 from its initiator (in **DataIn [ ]** array).

The delay between a reception from its initiator and the transmission of the corresponding response elaborated by the host controller is not completely under the PN532 control (the host controller may take a long time to prepare the data to be returned).
To bypass this potential problem, the PN532 automatically generates the necessary Supervisory pdu ($S(TO)_{REQ}$) for DEP or the S(WTX) request for ISO/IEC14443-4 protocol.

a. DEP Protocol

Regarding the DEP protocol, a typical data exchange between the PN532 as target and a NFC Initiator can be represented as follows (Fig 88):

- When the host controller wants to retrieve a command message coming from the initiator, it uses the **`TgGetData`** command,
- In that case, the PN532 sends back an ACK frame to the host controller and then waits for available data from the initiator. It may take a long time before data are available (T_{cmd}),
- As soon as it has received a complete RF frame from the initiator, the PN532 uses a supervisory frame to ask for time extension to the initiator (7 x 154 ms = 1.078s)[13],
- Then, the PN532 sends the received RF frame back to the host controller,
- After having processed these data, the host controller shall use the **`TgSetData`** command (§7.3.17, p:164) to complete the exchange.

[13] 7 is the default value of the RTOX parameter sent by the PN532 in an $S(TO)_{REQ}$.
154 ms corresponds to the default value of RWT (Response Waiting Time) for the PN532 configured as target.

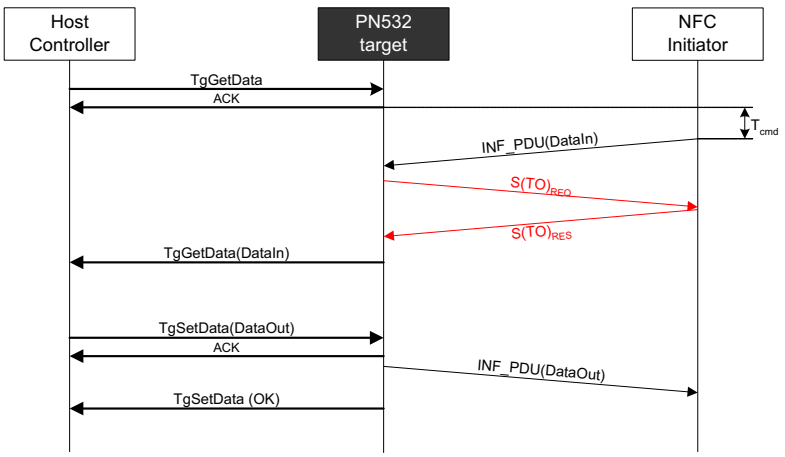


Fig 88. TgGetData (1)

In this schematic representation, no chaining is shown.
Refer to §**7.4.5**, p:178 to have a more detailed explanation of the chaining mechanism in case of the PN532 configured as target.

The PN532 can also accept a INF_PDU incoming frame from the initiator even if it has not received yet a **`TgGetData`** command from the host controller. The protocol exchange with the initiator is then preserved by using $S(TO)_{REQ}$.

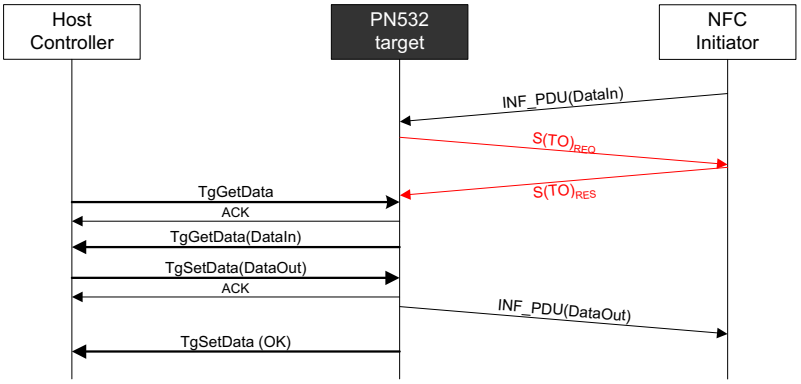


Fig 89. TgGetData (2)

b. ISO/IEC14443-4 protocol

Regarding the ISO/IEC14443-4 protocol, a typical data exchange between the PN532 as emulated ISO/IEC14443-4 PICC and a ISO/IEC14443-4 PCD can be represented as:

- When the host controller wants to retrieve a command message coming from the PCD, it uses the **`TgGetData`** command,
- In that case, the PN532 sends back an ACK frame to the host controller and then waits for available data from the PCD. It may take a long time before data are available (T_{cmd}),
- As soon as it has received a complete RF frame from the initiator, the PN532 uses a supervisory frame to ask for time extension to the initiator (7 x 154 ms = 1.078s)[14],
- Then, the PN532 sends the received RF frame back to the host controller,
- After having processed these data, the host controller shall use the **`TgSetData`** command to complete the exchange.

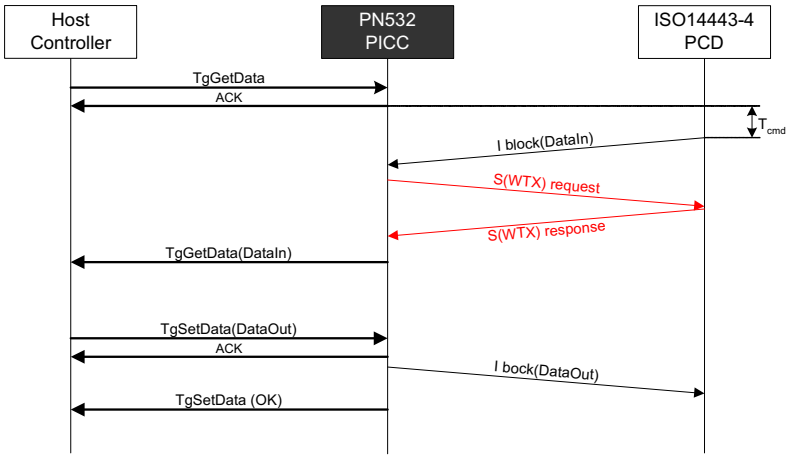


Fig 90. TgGetData for ISO/IEC14443-4 PICC emulated (1)

In this schematic representation, no chaining is shown.
Refer to §7.4.6, p: 184 to have a more detailed explanation of the chaining mechanism in case of the PN532 configured as PICC.

[14] 7 is the default value of the WTXM parameter sent by the PN532 in an $S(WTX)_{REQ.}$
154 ms corresponds to the default value of FWT for the PN532 configured as PICC.

The PN532 can also accept a I-block incoming frame from the PCD even if it has not received a **`TgGetData`** command yet from the host controller. The protocol exchange with the ISO/IEC14443-4 PCD is then preserved by using S(WTX) request.

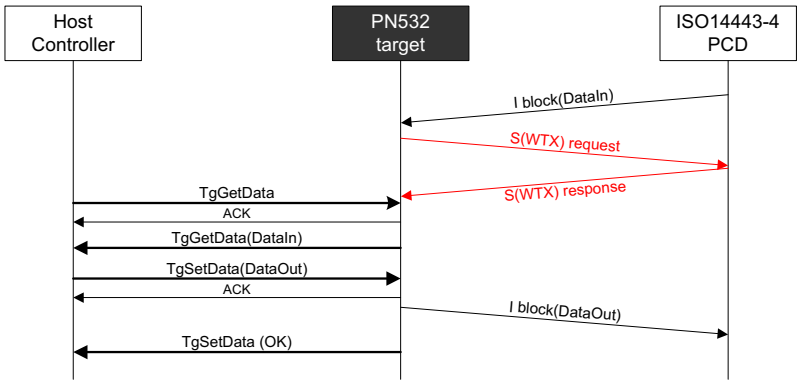


Fig 91. TgGetData for ISO/IEC14443-4 PICC emulated (2)

Possible errors returned:

- Target is not in a correct state to perform this operation (not in DEP protocol, nor in PICC emulation)

  ➔ A specific error code is returned (**Status** = 0x25)

- Target has been released

  ➔ A specific error code is returned (**Status** = 0x29)

7.3.17 TgSetData

This command is used in case of the PN532 configured as target for Data Exchange Protocol (DEP) or for ISO/IEC14443-4 protocol when PN532 is activated in ISO/IEC14443-4 PICC emulated (see §4, p:21). The overall amount of data to be sent can be transmitted in one frame (262 bytes maximum).

Input:

D4	8E	[DataOut [ ]]

- **DataOut [ ]** is an array of data (from 0 up to 262 bytes) to be sent by the PN532 as response to its initiator (see §7.4.7, p:186).

Output:

D5	8F	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67).

Description:

It allows the host controller to supply the PN532 with the data that it wants to send back to the initiator (in response of the previous RF DEP_REQ frame(s) for DEP and I-block for ISO/IEC14443-4 PICC emulation).

The PN532 sends in the RF link the data contained in **DataOut [ ]** array.
The protocol management (chaining, error handling) is completely managed internally by the PN532.

A typical data exchange between the PN532 as target and a NFC Initiator is represented in the **`TgGetData`** command description.

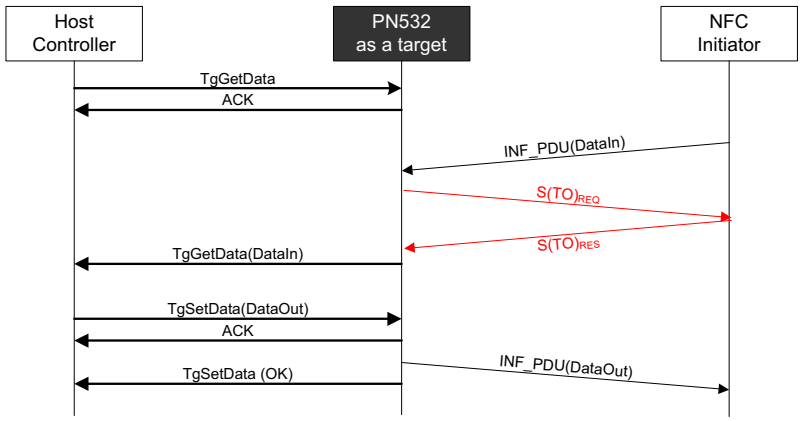


Fig 92. TgSetData

The examples given in §7.4.5, p:178 show how the chaining is handled either by the initiator or by the target.

A typical data exchange between the PN532 as ISO/IEC14443-4 PICC emulation target and a ISO/IEC14443-4 PCD is represented in the **`TgGetData`** command description.

DataOut [ ] contains the corresponding R-APDU.

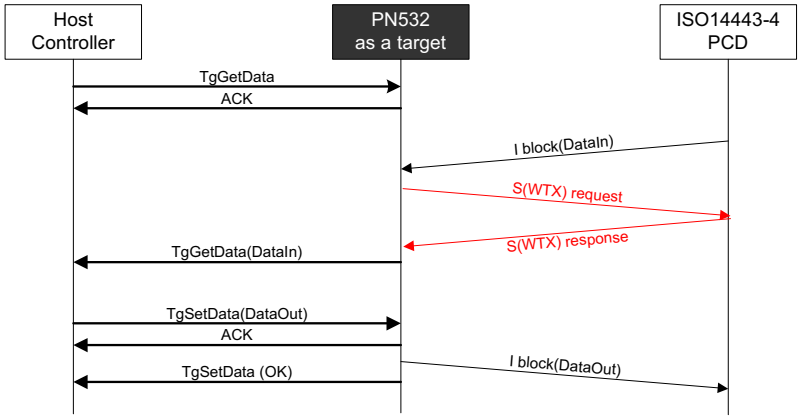


Fig 93. TgSetData for ISO/IEC14443-4 PICC emulation

The examples given in §7.4.6, p:184 show how the chaining is handled either by the ISO/IEC14443-4 PCD or by the ISO/IEC14443-4 PICC emulated target.

7.3.18 TgSetMetaData

This command is used in case of the PN532 configured as target for Data Exchange Protocol (DEP) if the overall amount of data to be sent cannot be transmitted in one frame (more than 262 bytes).

Input:

D4	94	DataOut [ ]

- **DataOut [ ]**is an array of data (from 0 up to 262 bytes) to be sent by the PN532 as response to its initiator.

Output:

D5	95	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67)

Syntax Error Conditions:

- The PN532 is currently activated as ISO/IEC14443-4 PICC, and therefore this command is not supported.

Description:

The main difference compared to the **`TgSetData`** command (see §7.3.17, p:164) is therefore that in the last chained packet sent by the PN532 to the initiator, the PFB control byte will contain the More Information bit set to one.

A typical data exchange using this command is shown in the following figure:

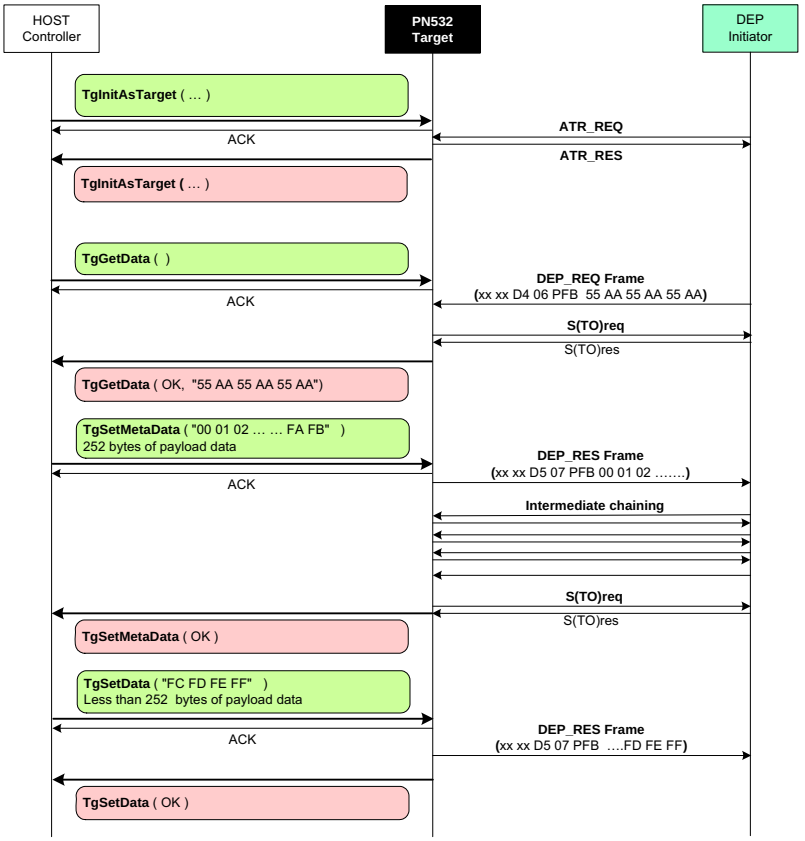


Fig 94. TgSetMetaData

7.3.19 TgGetInitiatorCommand

This command is used to get a packet of data from an initiator and to send it back to the host controller.

Input:

D4	88

Output:

D5	89	Status	InCommand[ ]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67
- **InCommand** is an array of raw data (from 0 up to 262 bytes) received by the PN532 (command from the initiator).

Description:

This command is used when the PN532 is configured as target (see **`TgInitAsTarget`**, §7.3.14, p:151).

The received data are simply returned to the host controller (in **`InCommand[]`** array) that will process them and then use the **`TgResponseToInitiator`** command (§7.3.20, p:170) to give the response to the initiator.

Depending on the mode and the baud rate used, the frame received from the initiator will be de-encapsulated before sending back data to the host controller.

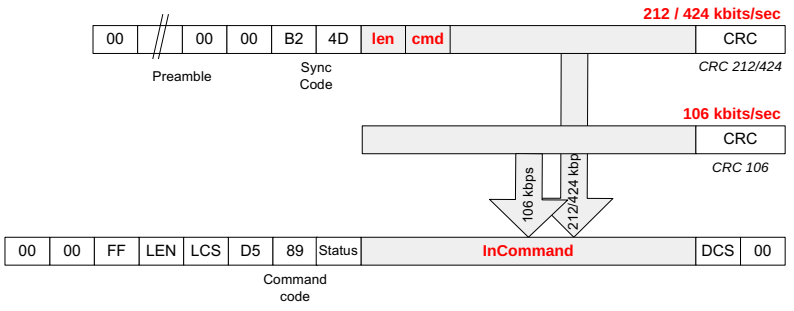


Fig 95. TgGetInitiatorCommand (1)

This command is complementary of the **`TgGetData`** command.
The main difference compared to **`TgGetData`** is that here the PN532 does not handle at all the protocol (DEP) features (Supervisory, chaining, error handling …). As a consequence, the **`TgGetData`** command may be used to carry information whatever the protocol used.

So, this command combined with **`TgResponseToInitiator`** (§7.3.20, p:170) may be used to build exchanges without using the protocol handled by the PN532 (DEP).

The following figure depicts the linking of the exchanges between the initiator and the PN532 and between the host controller and the PN532.

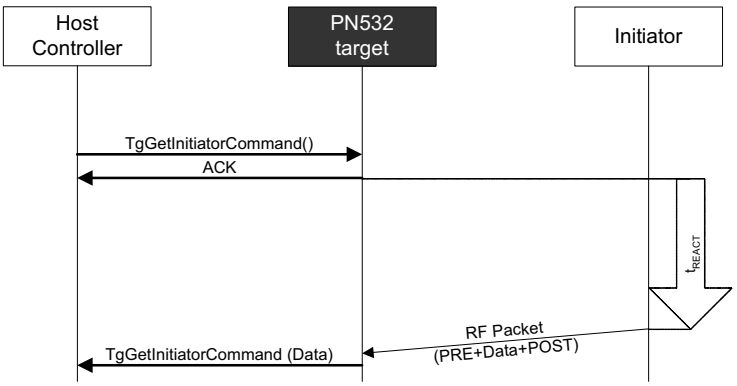


Fig 96. TgGetInitiatorCommand (2)

No control is done on the delay (t_{REACT}) used by the initiator to send its command frame. The host controller has to manage timeout by itself (it can stop the current **`TgGetInitiatorCommand`** command by using one of the two dedicated ways of stopping: ACK frame or new command frame).

7.3.20 TgResponseToInitiator

This command is used to send a response packet of data to an initiator.

Input:

D4	90	TgResponse[ ]

- **TgResponse** is an array of raw data (from 0 up to 262 bytes) to be sent by the PN532 (response to the initiator).

Output:

D5	91	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §7.1, p:67)**.**

Description:

This command is usually used in co-operation with **`TgGetInitiatorCommand`** (§7.3.19, p:168).

The received data from the host controller (**TgResponse [ ]** array) are simply encapsulated in the RF frame which format depends on the mode and the RF baud rate used.

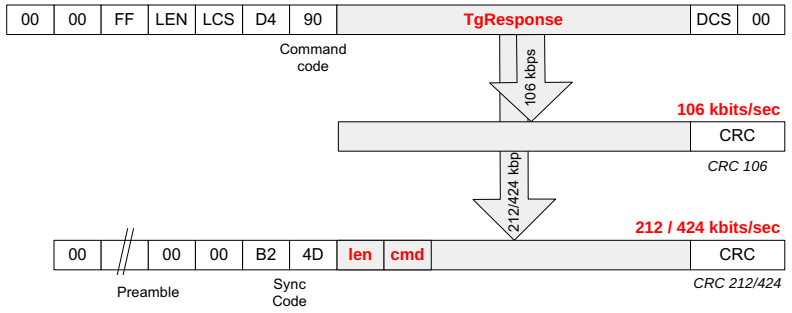


Fig 97. TgResponseToInitiator (1)

This command is complementary of the **`TgSetData`** command.
The main difference compared to **`TgSetData`** is that here the PN532 does not handle at all the protocol (DEP) features (Supervisory, chaining, error handling …).

This command, coupled with the **`TgGetInitiatorCommand,`** is the counterpart of the **`InCommunicateThru`** (see §7.3.9, p:136) command from the initiator side.

The following figure depicts how the exchanges between the initiator and the PN532, but also between the host controller and the PN532 are cascaded.

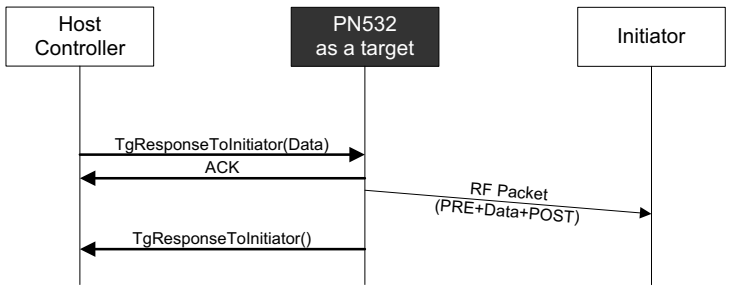


Fig 98. TgResponseToInitiator (2)

The **`TgResponseToInitiator`** command ends as soon as the RF frame is sent to the initiator.
If an error is detected, **Status** byte is filled in with an error code indicating the communication error.

7.3.21 TgGetTargetStatus

This command is used by the host controller to know what the current state of the PN532 is.

Input:

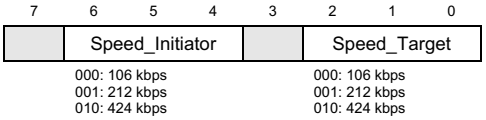
D4	8A

Output:

D5	8B	State	BRit

- **State** gives information about the current state of the PN532.
  - 0x00 TG_IDLE / TG_RELEASED
    the PN532 (acting as NFCIP-1 target) waits for an initiator or has been released by its initiator,
  - 0x01 TG_ACTIVATED
    the PN532 is activated as NFCIP-1 target,
  - 0x02 TG_DESELECTED
    the PN532 (acting as NFCIP-1 target) has been de-selected by its initiator,
  - 0x80 PICC_RELEASED
    the PN532 (acting as ISO/IEC14443-4 PICC) has been released by its PCD (no more RF field is detected),
  - 0x81 PICC_ACTIVATED
    the PN532 is activated as ISO/IEC14443-4 PICC,
  - 0x82 PICC_DESELECTED
    the PN532 (acting as ISO/IEC14443-4 PICC) has been de-selected by its PDC.

- **BRit** gives information about the baud rate used only when in TG_ACTIVATED state:



Description:

The goal of this command is to offer the possibility for the host controller to know if the PN532 target has been either de-selected or released.

In addition, the baud rates used in both directions are also returned.

7.4 Commands summary

The host controller has several commands that can be used when the PN532 is configured either as initiator or as target:

7.4.1 Commands for Initiator mode

The Fig 99 summarizes all the possible commands that can be used when the PN532 is configured as initiator.

Initialization / Activation:

- InJumpForDEP
- InJumpForPSL
- InListPassiveTarget
- InAutoPoll
- InATR
- InPSL

Data Exchange:

- InDataExchange
- InCommunicateThru

Selection / De-Selection / Release:

- InSelect
- InDeselect
- InRelease

7.4.2 Commands for Target mode

The Fig 100 summarizes all the possible commands that can be used when the PN532 is configured as target.

Initialization:

- TgInitAsTarget
- TgSetGeneralBytes

Data Exchange:

- TgGetData
- TgSetData
- TgSetMetaData
- TgGetInitiatorCommand
- TgResponseToInitiator

7.4.3 Commands for ISO/IEC14443-4 PICC mode

All the possible commands that can be used when the PN532 is configured as ISO/IEC14443-4 PICC are:
Initialization:

- TgInitAsTarget

Data Exchange:

- TgGetData
- TgSetData
- TgGetInitiatorCommand
- TgResponseToInitiator

7.4.4 Target states summary

The Fig 101 details all the possible states for the PN532 configured as target in passive communication mode.

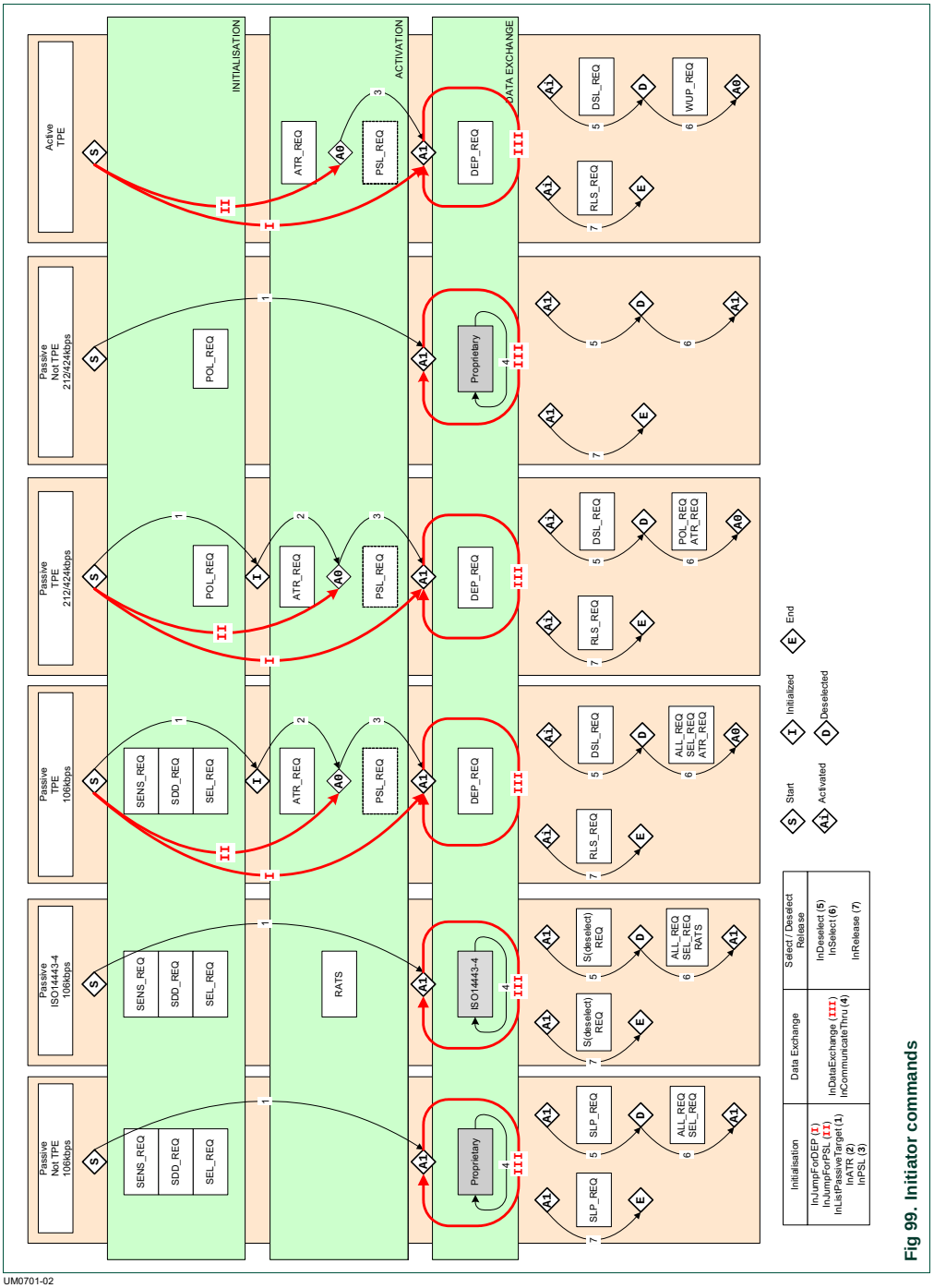


Fig 99. Initiator commands

UM0701-02

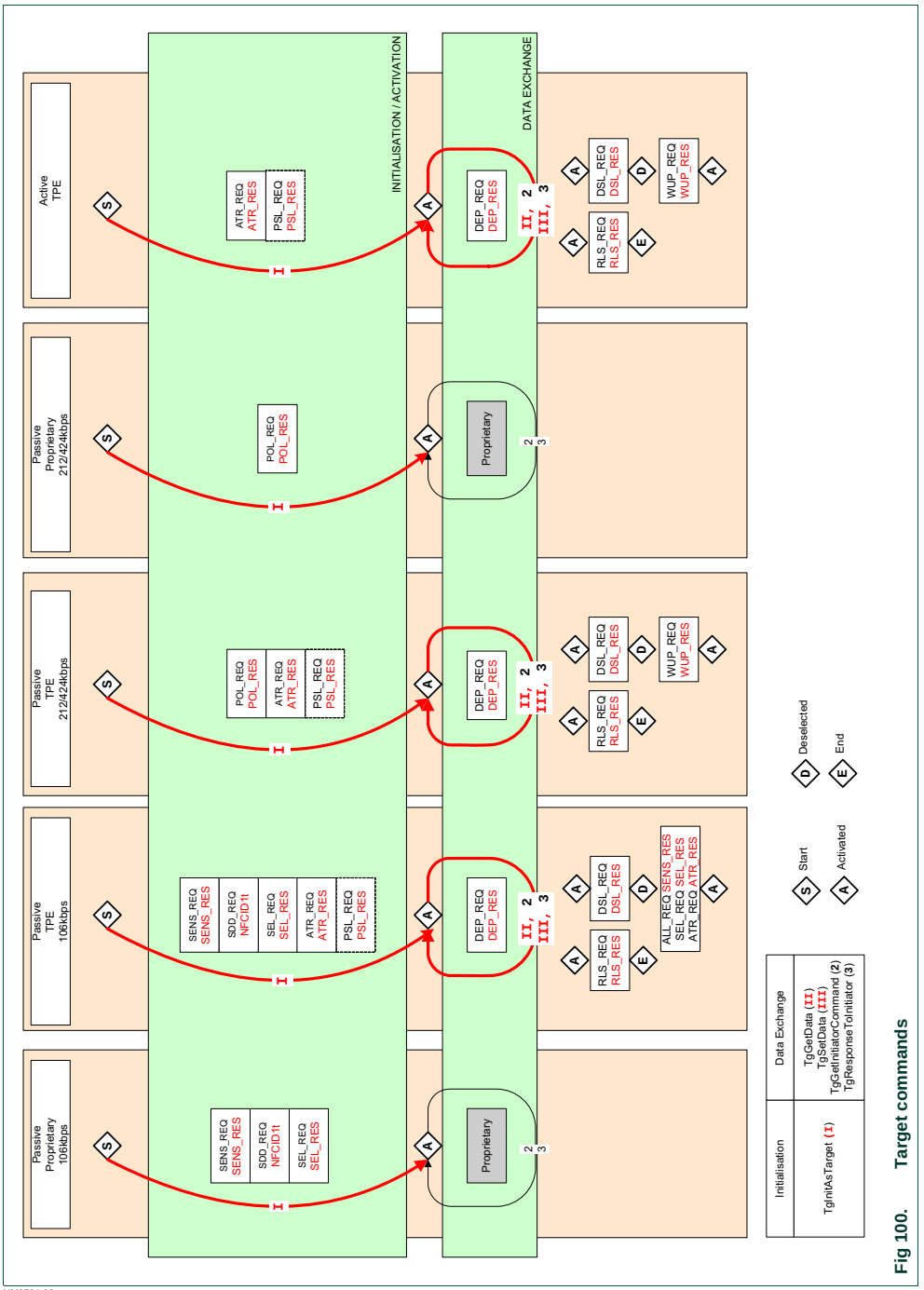


Fig 100. Target commands

UM0701-02

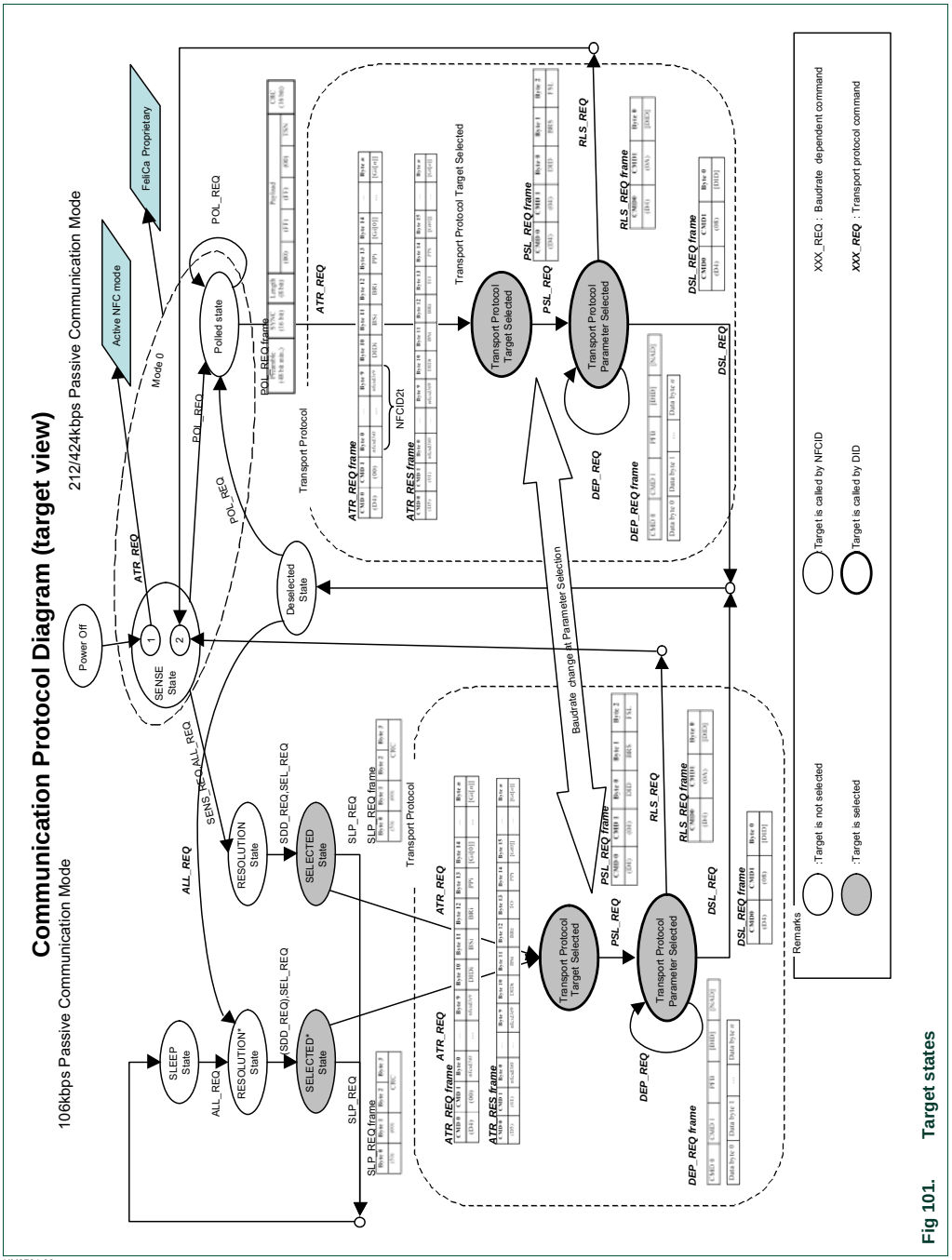


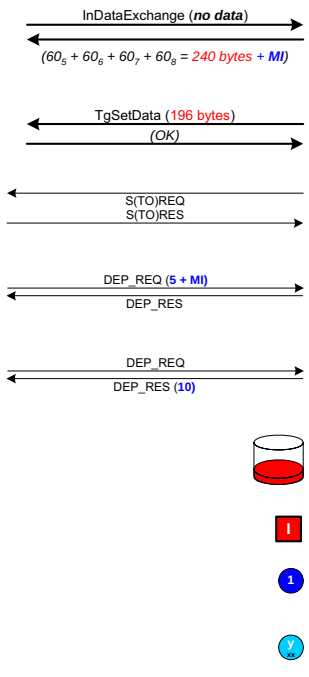
Fig 101. Target states

UM0701-02

7.4.5 DEP chaining mechanism

This chapter details how the PN532 configured as initiator or as target handles the DEP chaining mechanism. Four examples are given.

The following symbolic representation is used:



InDataExchange command without any data to send and with *240* bytes received. The ***MI*** bit is set in the status byte informing that additional data is available.
The subscript (5, 6, 7 and 8) represents the number of a RF packet of data exchanged.

TgSetData command with *196* bytes to send.

Time-Out extension request from the target, accepted by the initiator.

Data Exchange between the initiator and the target. The initiator sends in the **DEP_REQ** the ***5th*** packet of data and indicates (***MI***) that chaining is on-going. The target acknowledges this packet exchange with the **DEP_RES**.

Data Exchange between the initiator and the target. The initiator sends no data in the **DEP_REQ** whereas the target sends in the **DEP_RES** the ***10th*** packet of data. No chaining is on-going.

Container indicating the progress of the transfer either from the initiator to the target or from the target to the initiator.
The more red the container is, the more data it contains.

Stage number when filling up / emptying the container

Packet of 60 bytes of data that are conveyed with individual **DEP_REQ** / **DEP_RES** exchanges. The white number written inside corresponds to the packet number.

Packet of xx data bytes that are conveyed with individual **DEP_REQ** / **DEP_RES** exchanges. The white number y written inside corresponds to the packet number.

Fig 102. Legend used for the figures describing the chaining of data

In the *first example* shown (Fig 103), both the initiator and the target are supposed to be a PN532 which are **not using** Meta-Chaining.
The host controller of the initiator (A) sends packets of 262 bytes (maximal capacity of the **InDataExchange** command, see §7.3.8, p:127).
The target indicates a length reduction of 64 bytes, i.e. payload of 60 bytes[15].

The PN532 initiator cuts out the 262 bytes packet of data into individual packets of 60 bytes and sends these packets to the target.

After having received the 5th individual packet, the PN532 target sends back a $S(TO)_{REQ}$ RF frame to the initiator, and sends the information data (262 bytes) to its host controller (**B**).

In this example, **B** knows that these first 262 bytes are only a part of the complete "file" to transfer (header information for example) and then has no data to send back to **A**. Consequently, it uses **TgSetData** without data.

When the PN532 target receives this **TgSetData** command, it can send back a DEP_RES frame to the initiator.

This mechanism goes on until all the data are transferred.

In the *second example* (Fig 104), the initiator has a large memory area, meaning that the complete "file" to be transferred is ready in its memory.

The scenario is then a little bit different; the initiator maintains the **MI** bit in the PFB byte to 1 until the complete data are transferred.

The difference compared to the first example is that the PN532 target returns always the **MI** information to the host controller **B**. In that case, **B** does not need to use **TgSetData** as in the first example.

In the *third and fourth examples*, the PN532 uses Meta-Chaining functionality to allow transfer of "large" amount of data.

The Fig 105 represents the case of data to be transferred from the initiator to the target and the Fig 106 details the opposite case (data to be transferred from target to initiator).

In the third example, one will notice how the **MI** bit is used both in the **InDataExchange** and the **TgGetData** commands.

On the other hand, in the fourth example (Fig 106), the comparative use of the **TgSetMetaData** and **TgSetData** commands is shown.

[15] The total Transport Frame length indicated by the target is maximum **64** bytes. Thus the maximum payload data length is then **60** bytes, as there are **CMD0**, **CMD1** and **PFB** bytes to deduct (see **Error! Reference source not found.**, §12.1 and Fig.23).

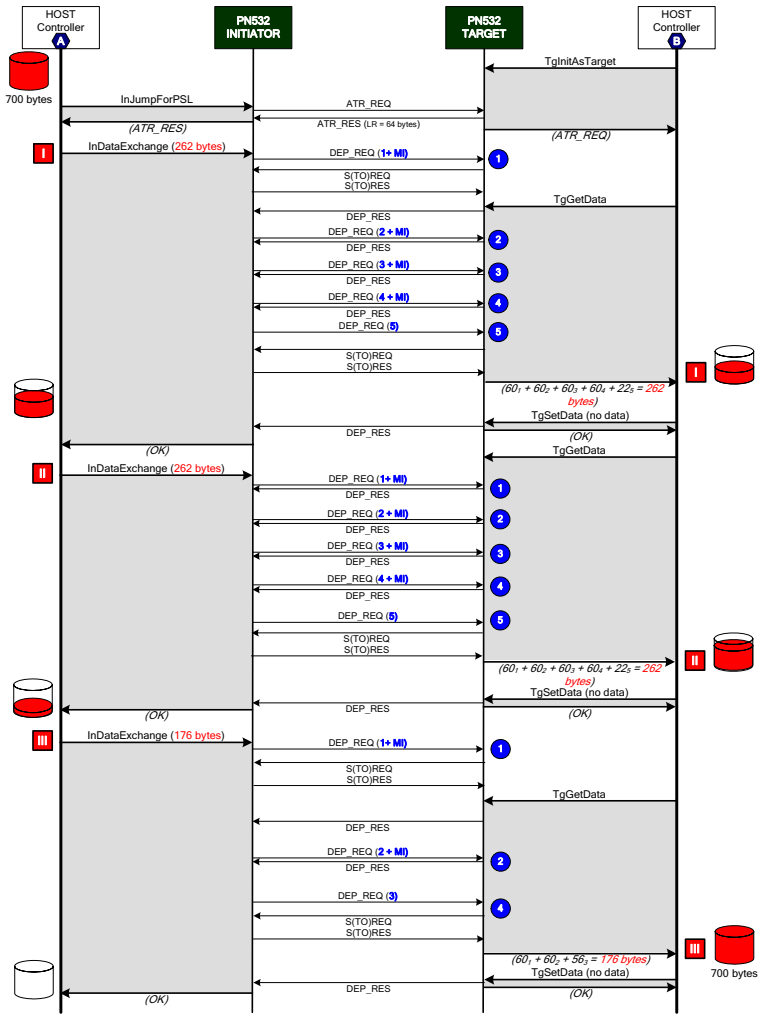


Fig 103. DEP chaining mechanism: packets transfer

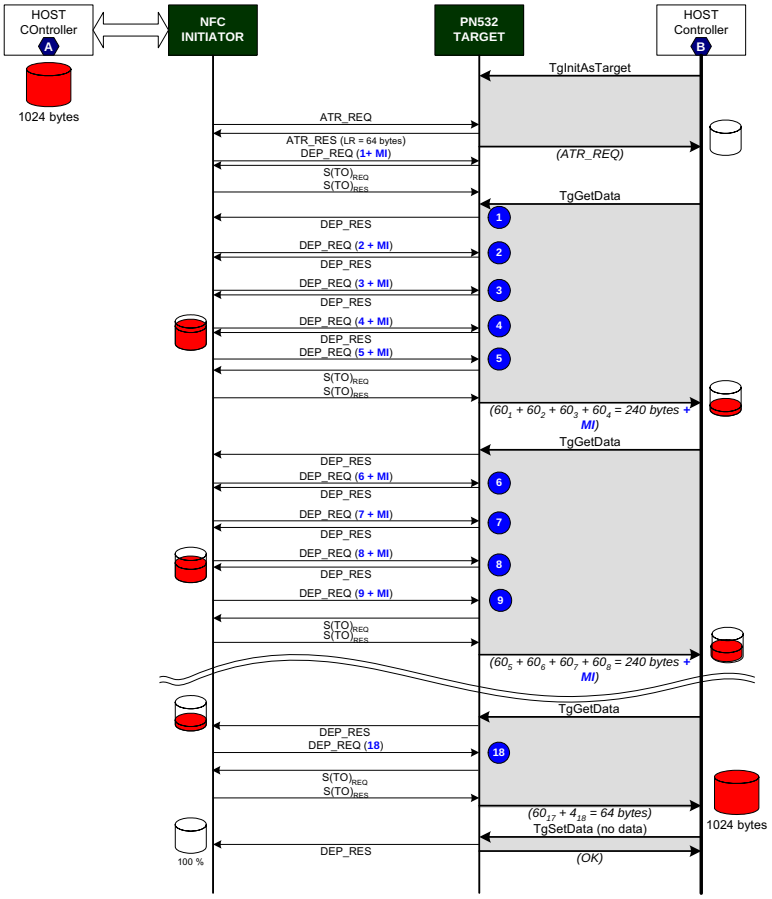


Fig 104. DEP chaining mechanism: “streaming” transfer from initiator to target

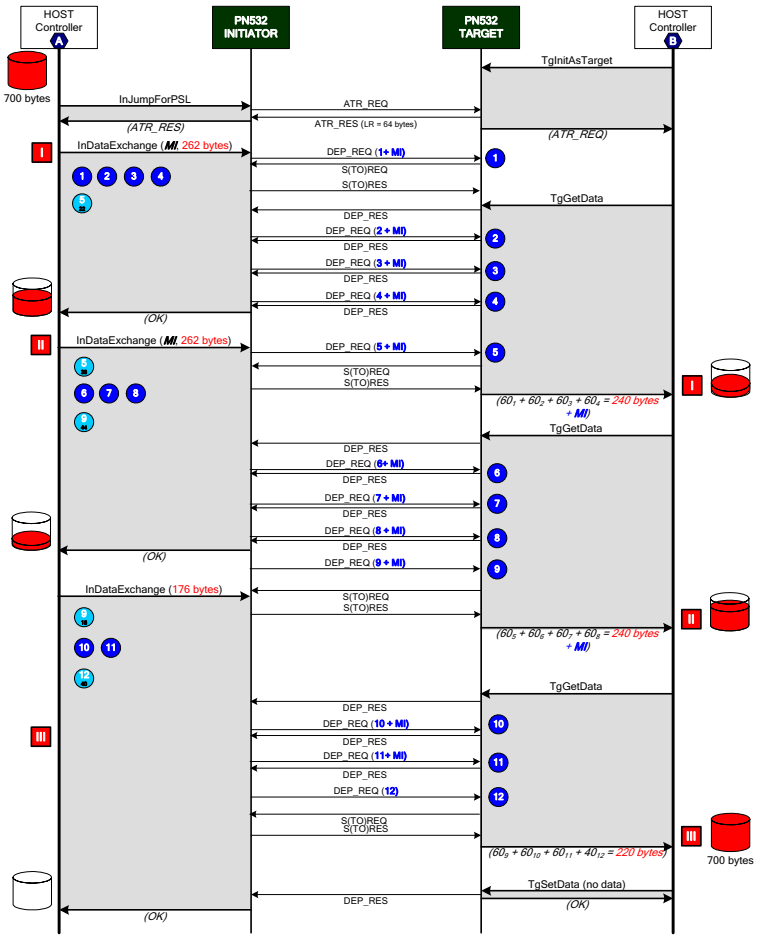


Fig 105. DEP chaining mechanism: transfer with Meta-Chaining – Initiator case

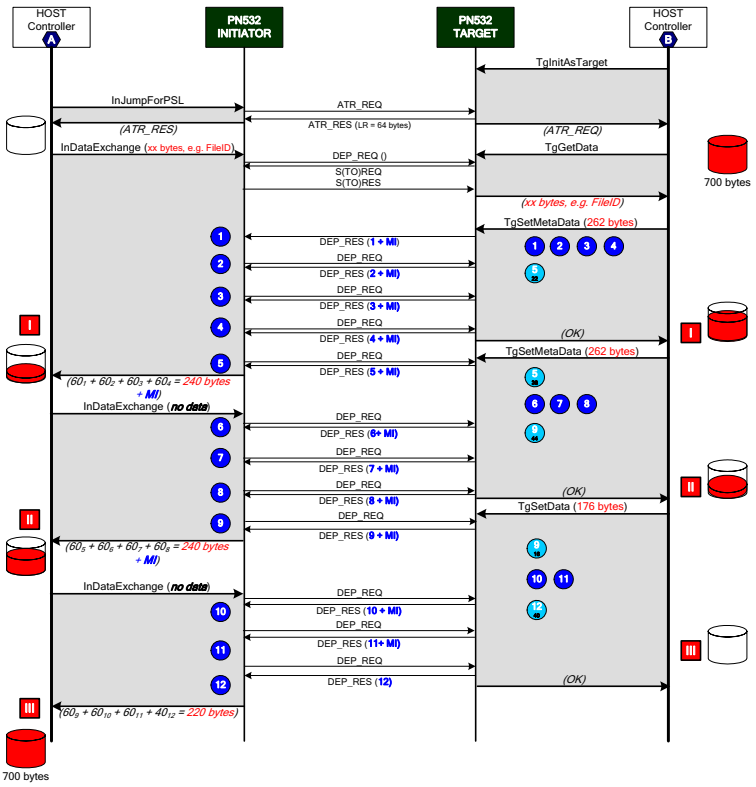


Fig 106. DEP chaining mechanism: transfer with Meta-Chaining – Target case

7.4.6 ISO/IEC14443-4 PICC emulated chaining mechanism

This chapter details how the PN532 configured as ISO/IEC14443-4 emulated target handles the chaining mechanism. Two examples are given.

In the *first example* shown (Fig 107), a ISO/IEC14443-4 PCD is requesting a 256 data bytes read operation (C-APDU Case 2). The expected answer is 256 + 2 = 258 bytes (R-APDU).

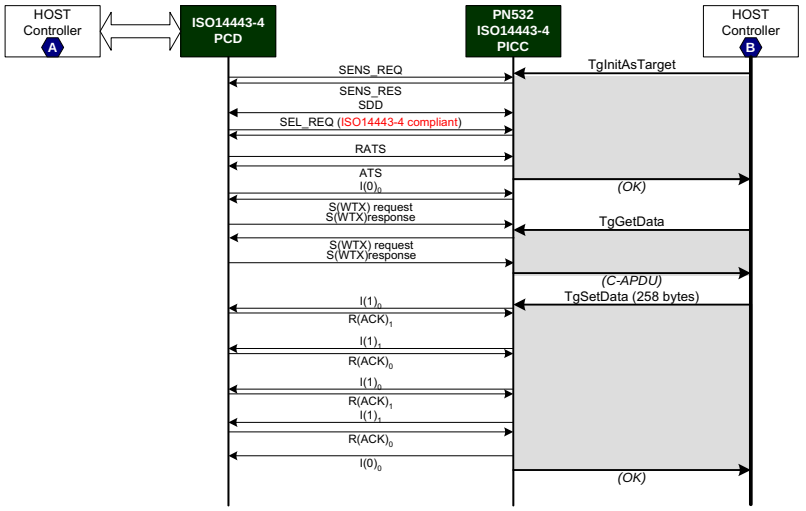


Fig 107. A 256 bytes read operation from PN532 in ISO/IEC14443-4 PICC emulation mode

In the *second example* (Fig 108), a ISO/IEC14443-4 PCD is performing a 255 data bytes C-APDU Case 4 command.
The expected answer (R-APDU) is 257bytes (255 data bytes + SW1, SW2).

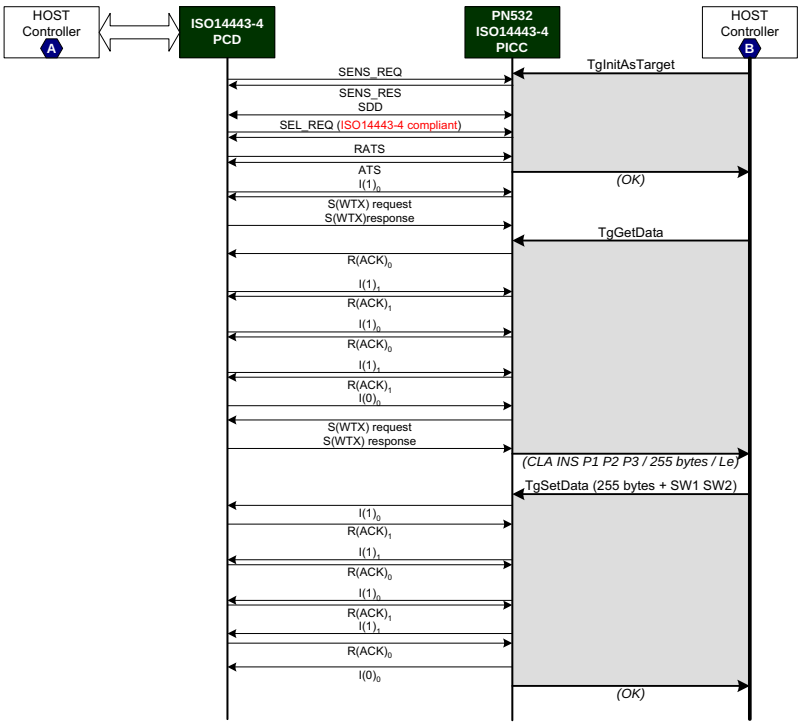


Fig 108. Example of case 4 operation in ISO/IEC14443-4 PICC emulation mode

7.4.7 Comparison of the length of Payload data field

The following figure depicts the available length for the payload field at two different levels:

- In the host controller protocol, for the commands used to get or set data either in initiator or target configuration.
  - **`InDataExchange`**,
  - **`TgGetData`**, **`TgSetData`** and **`TgSetMetaData`**.
- In the NFC Data Exchange Protocol (DEP).

This shows that the capacity of the payload data field at DEP level is lower than the one between the host controller and the PN532, even when DID field is not used (251 bytes vs. 250).

That means that if the host controller uses the total capacity of the **`InDataExchange`** command, the PN532 will have to handle chaining even with a target having a length reduction of 255 bytes (example shown in Fig 103).

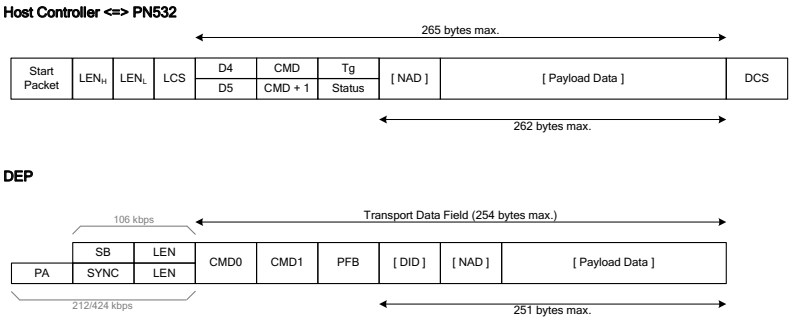


Fig 109. Payload data field maximum capacity

7.5 Examples of use

This paragraph gives some examples of use, detailing the commands used.

7.5.1 PN532 acting as Mifare PCD

The following example describes a short session with a Mifare Standard card.

The first step consists of initializing the Mifare card (the PN532 does not answer to the host controller as long as there is no target detected).
Then, in the second step, the PN532 makes the authentication of the Mifare card to allow reading and writing operations.

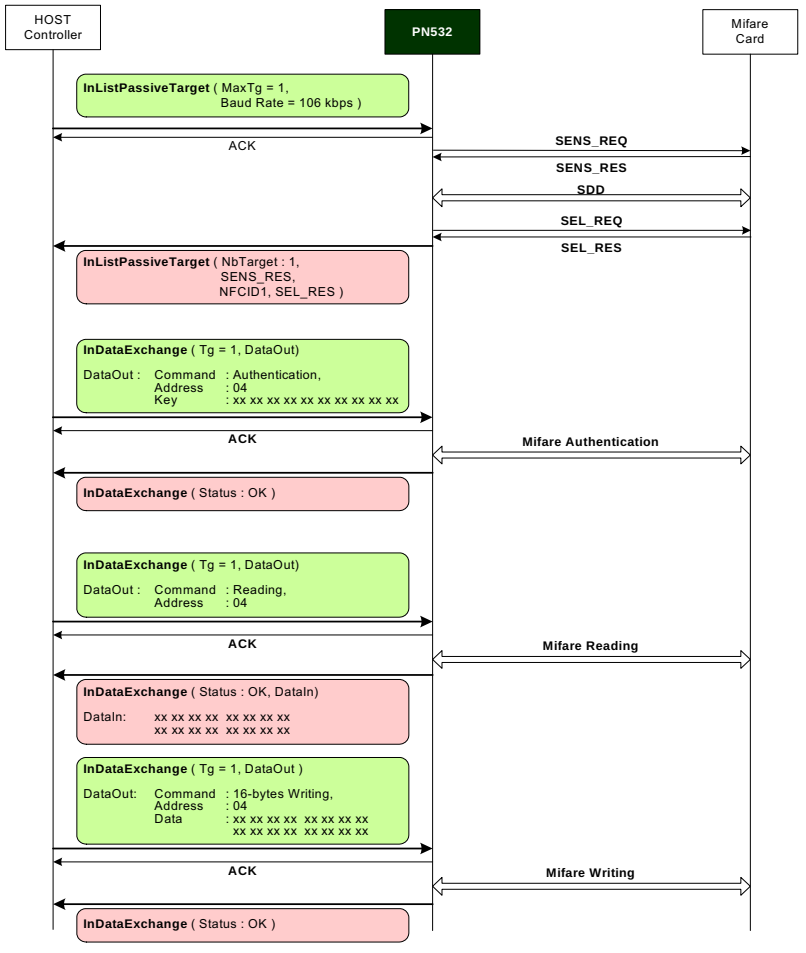


Fig 110. Mifare PCD example

7.5.2 PN532 acting as FeliCa PCD

The following example describes a short session with a FeliCa card.

The first step consists of initializing the FeliCa card (the PN532 does not answer to the host controller as long as there is no target detected). In the second step, the PN532 exchanges data with the FeliCa card using the **`InDataExchange`** command.

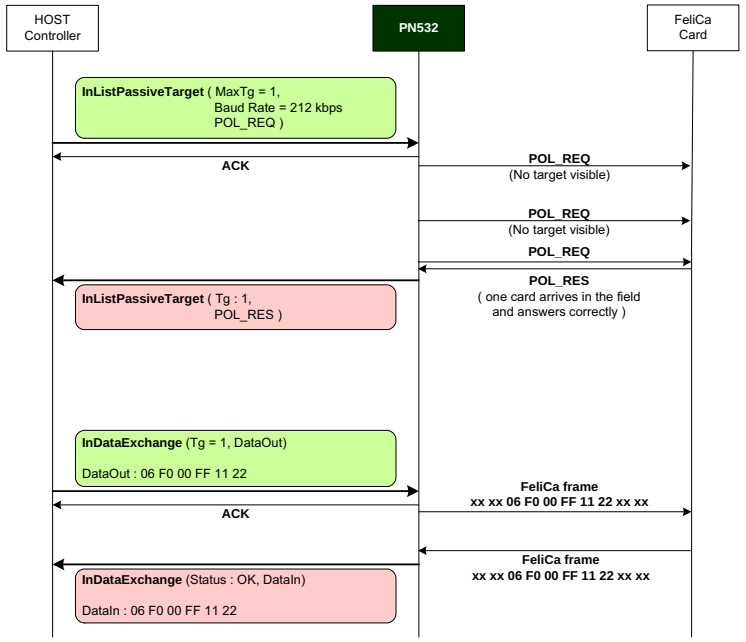


Fig 111. FeliCa PCD example

7.5.3 PN532 acting as 106 kbps target

This example shows how the PN532 behaves in front of a Mifare PCD, when it has been configured as target:

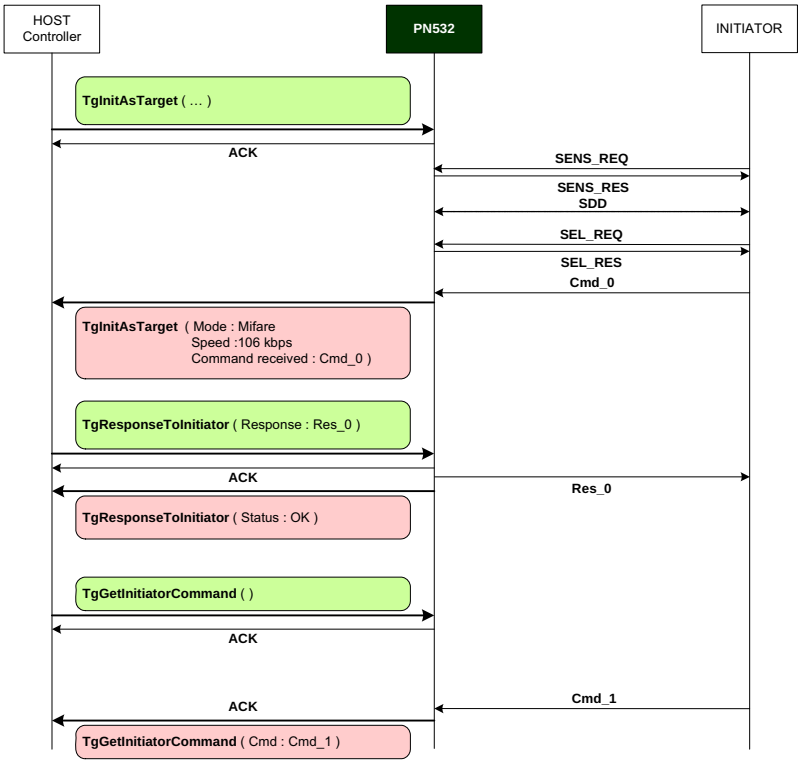


Fig 112. 106 kbps non-DEP target example

7.5.4 PN532 acting as 212 kbps target

This example shows how the PN532 behaves in front of a 212 kbps initiator, when it has been configured as target:

In this example, there are two POL_REQ / POL_RES exchanges during the initialization of the target.

This is just to show that **`TgInitAsTarget`** ends only after having received a command frame different from POL_REQ.

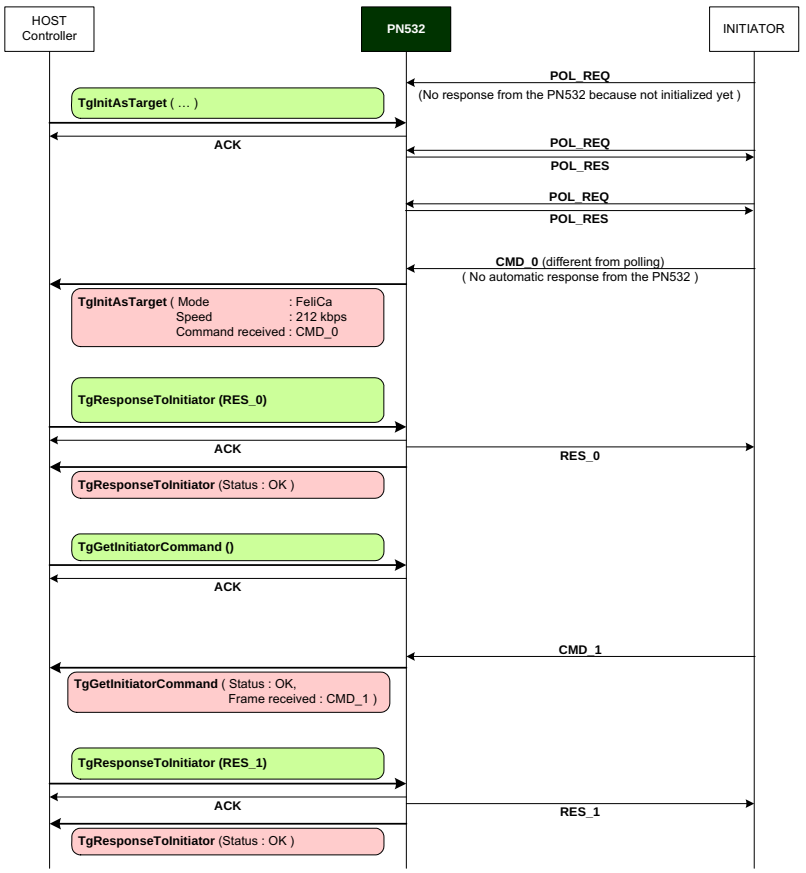


Fig 113. 212 kbps target example

7.5.5 Peer to Peer example with two PN532 (passive mode)

This example shows how to make a peer to peer communication in passive mode using two PN532 ICs.

The three **`InDataExchange`** commands at the initiator side and **`TgGetData`** and **`TgSetData`** at the target side allow building a communication based on the NFC-DEP protocol.

In this example, the communication is established in passive mode at 212 kbps. Other examples are available in §7.4.5, p:178, where the DEP chaining mechanism is considered.

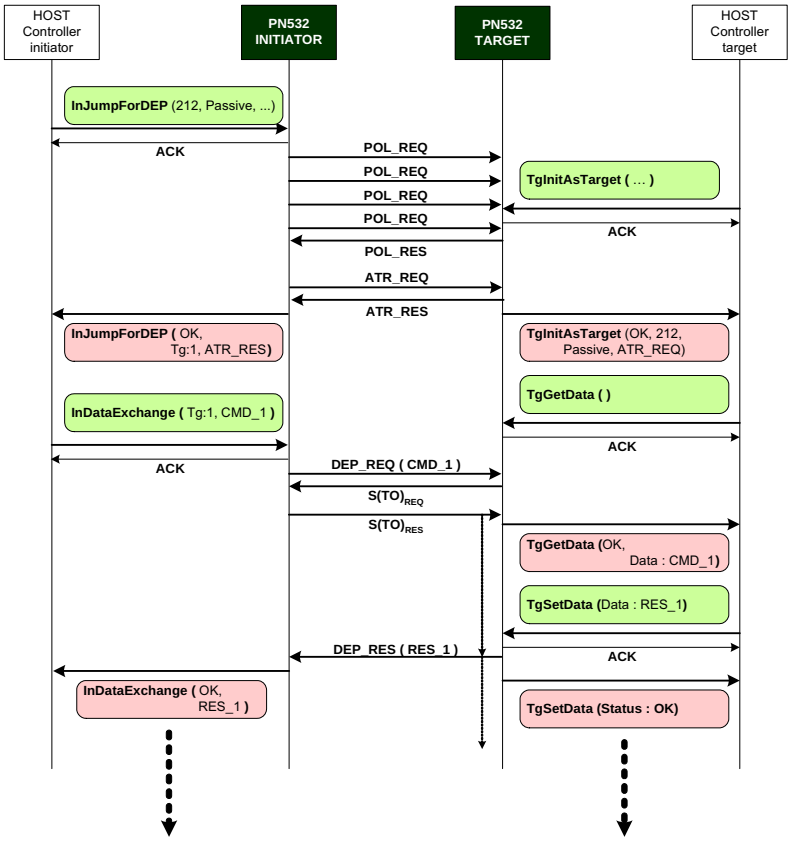


Fig 114. Peer to Peer communication example (passive mode)

7.5.6 Peer to Peer example with two PN532 (active mode)

This example shows how to make a peer-to-peer communication in active mode using two PN532 ICs.

The three commands **`InDataExchange`** at the initiator side and **`TgGetData`** and **`TgSetData`** at the target side allow building a communication based on the NFC-DEP protocol.

In this example, the communication is established in active mode whatever the baud rate is. From the host controller point of view (the one of the PN532 initiator and the one of the PN532 target), the set of commands to use is the same in active and in passive communication modes.

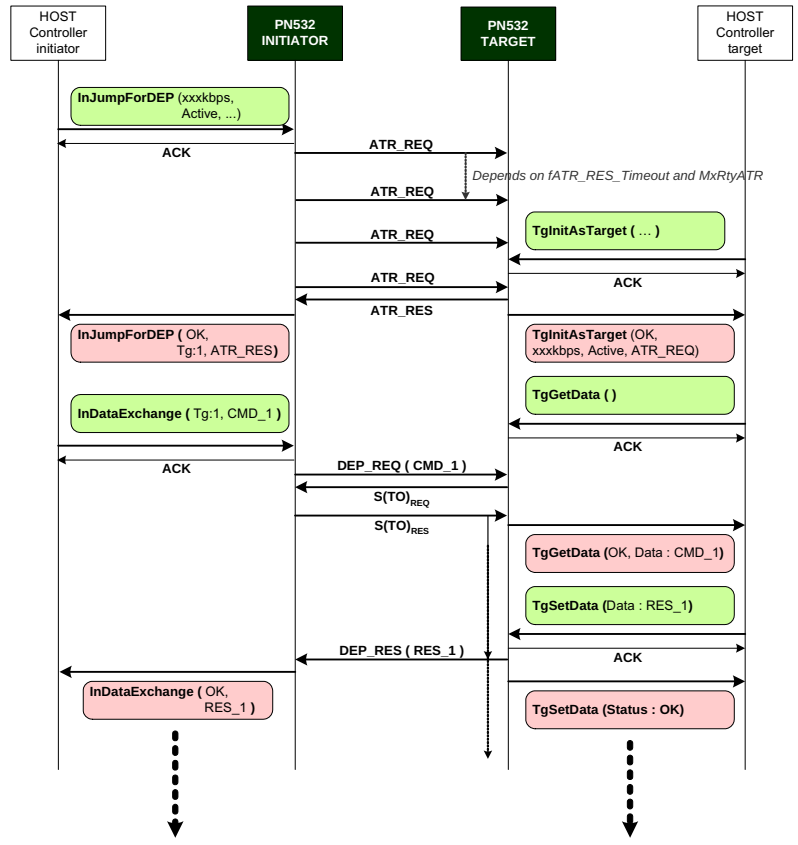


Fig 115. Peer to Peer communication example (active mode)

8. Appendix

8.1 Command set

The available commands are listed below:

Table 27. Command set

Command	Command Code	Page
Miscellaneous		
Diagnose	0x00	69
GetFirmwareVersion	0x02	73
GetGeneralStatus	0x04	74
ReadRegister	0x06	76
WriteRegister	0x08	78
ReadGPIO	0x0C	79
WriteGPIO	0x0E	81
SetSerialBaudRate	0x10	83
SetParameters	0x12	85
SAMConfiguration	0x14	89
PowerDown	0x16	98
RF communication		
RFConfiguration	0x32	101
RFRegulationTest	0x58	107
Initiator		
InJumpForDEP	0x56	108
InJumpForPSL	0x46	113
InListPassiveTarget	0x4A	115
InATR	0x50	122
InPSL	0x4E	125
InDataExchange	0x40	127
InCommunicateThru	0x42	136
InDeselect	0x44	139
InRelease	0x52	140

Command	Command Code	Page
InSelect	0x54	141
InAutoPoll	0x60	144
Target		
TgInitAsTarget	0x8C	151
TgSetGeneralBytes	0x92	158
TgGetData	0x86	160
TgSetData	0x8E	164
TgSetMetaData	0x94	166
TgGetInitiatorCommand	0x88	168
TgResponseToInitiator	0x90	170
TgGetTargetStatus	0x8A	172

9. Legal information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

9.3

Licenses

Purchase of NXP components

Not applicable

9.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

See footnote **Error! Bookmark not defined.**, p:**Error! Bookmark not defined.**

9.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

Mifare — is a trademark of NXP B.V.

10. Tables

Table 1. Configuration modes 6
Table 2. TX framing and TX speed in RFfieldON configuration 7
Table 3. CPU frequency used 9
Table 4. Power modes for CPU 10
Table 5. Power modes for CL interface 10
Table 6. CPU PowerMode used 12
Table 7. Host controller interface selection 24
Table 8. Pin used for SPI interface 25
Table 9. Pin used for HSU interface 26
Table 10. Pin used for I2C interface 26
Table 11. HSU timeout values 34
Table 12. Command set 65
Table 13. Error code list 67
Table 14. List of SFR registers 76
Table 15. Default values of internal flags 88
Table 16. Various timings 101
Table 17. Timings definition for RFConfiguration command 102
Table 18. Maximum retries 103
Table 19. Analog settings for the baudrate 106 kbps type A 104
Table 20. Analog settings for the baudrate 212/424 kbps 105
Table 21. Analog settings for the type B 105
Table 22. Analog settings for the baudrate 212/424 and 848 kbps with ISO/IEC14443-4 protocol 106
Table 23. InDeselect RF actions 139
Table 24. InRelease RF actions 140
Table 25. InSelect RF actions 141
Table 26. Target configuration – Automatic response 157
Table 27. Command set 193

continued >>

11. Figures

Fig 1. States of the PN532 regarding CPU frequency....9
Fig 2. Mode Dispatcher....11
Fig 3. Standby mode after FW decision....12
Fig 4. LowVbat mode....13
Fig 5. Initiator / PCD mode....14
Fig 6. Target / PICC mode....15
Fig 7. Virtual card mode....17
Fig 8. Wired Card mode....18
Fig 9. Initialization sequence....19
Fig 10. ISO/IEC14443-4 PICC emulation....22
Fig 11. Over-current detection before a RF communication command....23
Fig 12. Over-current detection during a RF communication command....23
Fig 13. Normal information frame....28
Fig 14. Extended Information frame....29
Fig 15. ACK frame....30
Fig 16. NACK frame....30
Fig 17. Error frame....31
Fig 18. Preamble....31
Fig 19. Postamble....32
Fig 20. Data link level: normal exchange....33
Fig 21. Data link level: error from the host controller to the PN532....34
Fig 22. Data link level: error from the PN532 to the host controller....35
Fig 23. Data link level: error from the PN532 to the host controller....35
Fig 24. Application level: Successive exchanges....36
Fig 25. Data link level: Abort....37
Fig 26. Application level: Abort a command and process a new one....38
Fig 27. Application level: Error detected....39
Fig 28. HSU link: frames....40
Fig 29. HSU link: general principle of communication....40
Fig 30. I2C link: frames....42
Fig 31. I2C link: general principle of communication....43
Fig 32. I2C link: using P70_IRQ pin....44
Fig 33. SPI: frames....45
Fig 34. SPI: general principle of communication....46
Fig 35. SPI link: using P70_IRQ pin....47
Fig 36. Handshake in case of HSU link – case 1....49
Fig 37. Handshake in case of HSU link – case 2....50
Fig 38. Handshake in case of HSU link – case 3....51
Fig 39. Handshake in case of HSU link – case 3 without H_REQ....52
Fig 40. Handshake in case of HSU link – case 4....53

continued >>

Fig 41. Handshake in case of I2C link – case 1 55
Fig 42. Handshake in case of I2C link – case 1bis 55
Fig 43. Handshake in case of I2C link – case 2 56
Fig 44. Handshake in case of I2C link – case 2 without H_REQ 57
Fig 45. Handshake in case of I2C link – case 3 58
Fig 46. Handshake in case of I2C link – case 4 59
Fig 47. Handshake in case of SPI link – case 1 60
Fig 48. Handshake in case of SPI link – case 2 with H_REQ 61
Fig 49. Handshake in case of SPI link – case 2 without H_REQ 62
Fig 50. Handshake in case of SPI link – case 3 63
Fig 51. Handshake in case of SPI link – case 4 64
Fig 52. SAM status byte definition 75
Fig 53. SetSerialBaudRate 84
Fig 54. fNADUsed 86
Fig 55. Status Byte definition 86
Fig 56. SAM electrical connection 90
Fig 57. SAM: Normal mode 90
Fig 58. SAM: Wired Card mode 91
Fig 59. SAM: Virtual Card mode 92
Fig 60. SAM: Detection of the start of a transaction 92
Fig 61. SAM: P70_IRQ triggered by the CLAD line 93
Fig 62. SAM: P70_IRQ triggered by RF field cut 94
Fig 63. SAM: P70_IRQ triggered by Timeout 95
Fig 64. SAM: Dual Card mode 96
Fig 65. HSU Wake up 99
Fig 66. I2C Wake up 100
Fig 67. SPI wake up 100
Fig 68. InJumpForDEP – Active communication mode – DID used 110
Fig 69. InJumpForDEP – Passive Communication Mode – DID not used 112
Fig 70. InDataExchange – General context 128
Fig 71. InDataExchange – Different target types 129
Fig 72. InDataExchange – Example of a ISO/IEC14443-4 exchange 131
Fig 73. InDataExchange – Example of a FeliCa exchange 133
Fig 74. InDataExchange – Example of a DEP exchange 134
Fig 75. InCommunicateThru (1) 137
Fig 76. InCommunicateThru (2) 137
Fig 77. InCommunicateThru (3) 138
Fig 78. InSelect 143
Fig 79. Auto-polling process 147
Fig 80. Some target activation times during an auto-polling process 148
Fig 81. TgInitAsTarget – Passive DEP 106 kbps 153

continued >>

Fig 82. TgInitAsTarget – ISO/IEC14443-4 PICC emulated 154
Fig 83. TgInitAsTarget – Proprietary command 155
Fig 84. TgInitAsTarget – Passive DEP 212/424 kbps 156
Fig 85. TgInitAsTarget – Passive 212/424 kbps – Proprietary command 156
Fig 86. TgInitAsTarget – Active mode 157
Fig 87. TgSetGeneralBytes 159
Fig 88. TgGetData (1) 161
Fig 89. TgGetData (2) 161
Fig 90. TgGetData for ISO/IEC14443-4 PICC emulated (1) 162
Fig 91. TgGetData for ISO/IEC14443-4 PICC emulated (2) 163
Fig 92. TgSetData 164
Fig 93. TgSetData for ISO/IEC14443-4 PICC emulation 165
Fig 94. TgSetMetaData 167
Fig 95. TgGetInitiatorCommand (1) 168
Fig 96. TgGetInitiatorCommand (2) 169
Fig 97. TgResponseToInitiator (1) 170
Fig 98. TgResponseToInitiator (2) 171
Fig 99. Initiator commands 175
Fig 100. Target commands 176
Fig 101. Target states 177
Fig 102. Legend used for the figures describing the chaining of data 178
Fig 103. DEP chaining mechanism: packets transfer 180
Fig 104. DEP chaining mechanism: “streaming” transfer from initiator to target 181
Fig 105. DEP chaining mechanism: transfer with Meta-Chaining – Initiator case 182
Fig 106. DEP chaining mechanism: transfer with Meta-Chaining – Target case 183
Fig 107. A 256 bytes read operation from PN532 in ISO/IEC14443-4 PICC emulation mode 184
Fig 108. Example of case 4 operation in ISO/IEC14443-4 PICC emulation mode 185
Fig 109. Payload data field maximum capacity 186
Fig 110. Mifare PCD example 187
Fig 111. FeliCa PCD example 188
Fig 112. 106 kbps non-DEP target example 189
Fig 113. 212 kbps target example 190
Fig 114. Peer to Peer communication example (passive mode) 191
Fig 115. Peer to Peer communication example (active mode) 192

continued >>

12. Contents

1. Introduction 3
1.1 Purpose and Scope 3
1.2 Intended audience 3
1.3 Glossary 3
1.4 References 4
1.5 General presentation of the PN532 5
2. Configuration Modes 6
2.1 Standard Mode 6
2.2 PN512 emulation mode 6
2.3 RFfieldON Mode 7
3. Power management 8
3.1.1 CPU frequency 9
3.1.2 Power modes of the PN532 10
3.1.3 Operating modes of the PN532 11
4. ISO/IEC14443-4 PICC emulation concept 21
5. Over-current detection 23
6. Host controller Interfaces 24
6.1 General points 24
6.1.1 Possible links 24
6.1.2 P70_IRQ pin 27
6.2 Host controller communication protocol 28
6.2.1 Frames structure 28
6.2.2 Dialog structure 33
6.2.3 HSU communication details 40
6.2.4 I2C communication details 42
6.2.5 SPI communication details 45
6.3 Handshake mechanism 48
6.3.1 General presentation 48
6.3.2 Handshake mechanism in case of HSU link 49
6.3.3 Handshake mechanism in case of I2C link 54
6.3.4 Handshake mechanism in case of SPI link 60
7. Commands supported 65
7.1 Error handling 67
7.2 Miscellaneous commands 69
7.2.1 Diagnose 69
7.2.2 GetFirmwareVersion 73
7.2.3 GetGeneralStatus 74
7.2.4 ReadRegister 76
7.2.5 WriteRegister 78
7.2.6 ReadGPIO 79
7.2.7 WriteGPIO 81
7.2.8 SetSerialBaudRate 83
7.2.9 SetParameters 85
7.2.10 SAMConfiguration 89
7.2.11 PowerDown 98
7.3 RF Communication command 101
7.3.1 RFConfiguration 101
7.3.2 RFRegulationTest 107
7.3.3 InJumpForDEP 108
7.3.4 InJumpForPSL 113
7.3.5 InListPassiveTarget 115
7.3.6 InATR 122
7.3.7 InPSL 125
7.3.8 InDataExchange 127
7.3.9 InCommunicateThru 136
7.3.10 InDeselect 139
7.3.11 InRelease 140
7.3.12 InSelect 141
7.3.13 InAutoPoll 144
7.3.14 TgInitAsTarget 151
7.3.15 TgSetGeneralBytes 158
7.3.16 TgGetData 160
7.3.17 TgSetData 164
7.3.18 TgSetMetaData 166
7.3.19 TgGetInitiatorCommand 168
7.3.20 TgResponseToInitiator 170
7.3.21 TgGetTargetStatus 172
7.4 Commands summary 173
7.4.1 Commands for Initiator mode 173
7.4.2 Commands for Target mode 174
7.4.3 Commands for ISO/IEC14443-4 PICC mode 174
7.4.4 Target states summary 174
7.4.5 DEP chaining mechanism 178
7.4.6 ISO/IEC14443-4 PICC emulated chaining mechanism 184
7.4.7 Comparison of the length of Payload data field 186
7.5 Examples of use 187
7.5.1 PN532 acting as Mifare PCD 187
7.5.2 PN532 acting as FeliCa PCD 188
7.5.3 PN532 acting as 106 kbps target 189
7.5.4 PN532 acting as 212 kbps target 190
7.5.5 Peer to Peer example with two PN532 (passive mode) 191
7.5.6 Peer to Peer example with two PN532 (active mode) 192
8. Appendix 193
8.1 Command set 193
9. Legal information 195
9.1 Definitions 195
9.2 Disclaimers 195
9.3 Licenses 195
9.4 Patents 195
9.5 Trademarks 195
10. Tables 196
11. Figures 197
12. Contents 200

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: sales.addresses@www.nxp.com

Date of release: 5th November 2007
Document identifier: UM0701-022

founded by Philips