Semiconductors

UM0501-02

PN531/C2 User Manual

Rev. 02 — 25 November 2005 **User Manual**

Document information

Info	Content
Keywords	NFC, PN531/C2, TAMA, V4.2
Abstract	This document describes the firmware V4.2 embedded in the PN531/C2.

PHILIPS

Revision history

Rev	Date	Description
02	2005-11-25	Update for firmware version **V4.2**
01	2005-07-11	Initial version for firmware version **V4.1**

1. Introduction

1.1 Purpose and Scope

The PN531 is a highly integrated transmission module for contactless communication at 13.56 MHz including microcontroller functionality based on a 80C51 core with 32 kbytes of ROM and 1 kbyte of RAM.

The PN531 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 interfaces.

This document describes the firmware embedded in the PN531 chip, in particular the global behaviour in the system depending if the PN531 device is used as Initiator or Target.

1.2 Intended audience

This document has been written to allow the use of the PN531 from the host point of view.

All the RF protocols used by the PN531 are not described in this document. The reader is supposed to have knowledge on NFCIP-1 (Reference [**1**]) and ISO14443 (Reference [**2**]).

1.3 Glossary

APDU	Application Protocol Data Unit
C-APDU	Command APDU
CLAD	ContactLess Active Detection
DEP	Data Exchange Protocol
HSU	High Speed UART
I2C	Inter Integrated Circuit
IC	Integrated Circuit
N/A	Not Applicable
N/I	Not Implemented
N/U	Not Used
PCB	Protocol Control Byte (ISO14443-4)
PFB	Control Information for Transaction (NFCIP-1)
RATS	Request for Answer To Select
RFU	Reserved for Future Use
R/W	Reader / Writer for contactless smartcards
SAM	Security Access Module
SPI	Serial Peripheral Interface
TAMA	Nickname of PN531
TPE	Transport Protocol Equipped
USB	Universal Serial Bus

1.4 References

[1]	ISO/IEC 18092	Near Field Communication - Interface and Protocol (NFCIP-1)
[2]	ISO/IEC 14443-4	Identification cards – Contactless integrated circuit(s) cards - Proximity card(s) Part 4: Transmission protocol
[3]	PN531/C2 Data sheet	PN531/C2 NFC Controller data sheet

1.5 General presentation of the PN531

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

- USB 2.0 full speed
- I2C
- SPI and
- Serial High Speed UART (HSU)

The host protocol is defined in chapter 3.

The firmware supports 3 different operating modes:

- Reader/writer mode for FeliCa™, ISO14443A/MIFARE® and ISO14443A-4 cards
- Supports Card interface mode for FeliCa™ and ISO14443A/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 PN531 handles the complete NFC framing and error detection.

All the different operating modes of the PN531 are configured by using a set of high-level commands described in chapter 4.

2. Configuration Modes

The PN531 has 3 possible modes that can be chosen by using two GPIOs during the reset phase of the IC:

Table 1: Configuration modes

	Selection Pin	
Mode	**IRQ** (pin #28)	**P35** (pin #21)
Normal	1	1
Emulation of the PN511	1	0
RF field ON	0	1

2.1 Normal Mode

This is the default mode of the PN531.
The description of this mode is detailed in this document starting from chapter 3.

2.2 Emulation of the PN511

In this test mode, the PN531 is configured to act as a real PN511 (Joiner) IC using serial interface.
Then, the PN531 can be easily interfaced with the PN511 dedicated host software, as e.g. **Joiner PC Serial**.

The link used is RS232 at 9600 bauds[1]. It is not possible to change the value of the baud rate, the *SerialSpeedReg* register is not emulated.

The emulation of the PN511 IRQ line is supported as well, the pin used is IRQ.
The level of the IRQ line is low when an interrupt occurs. The bit *IRQInv* in the register *CommIEnReg* has no effect (See reference [**3**]).

[1] The RS232 link used here is the standard UART, not the High Speed UART. Consequently, in this mode the PN531 must be interconnected with P30 (pin#27) for the RX line and P31 (pin#26) for the TX line.

2.3 RFfieldON Mode

In this mode, the PN531 is configured to switch on the RF field immediately after the reset.

The modulation and the baud rate used depend on the selection pins P33 and P34 and random data bytes are continuously sent:

Table 2: Tx framing and Tx speed in RFfieldOn configuration

	Selection Pins	
Tx framing – Tx speed	**P33** (pin #36)	**P34** (pin #22)
Mifare® - 106 kbps	1	1
	0	0
FeliCa™ - 212 kbps	0	1
FeliCa™ - 424 kbps	1	0

3. Host Interfaces

3.1 General points

3.1.1 Possible links

The system host controller can communicate with the PN531 by using either the USB interface, SPI, I2C or HSU (High Speed UART) serial links.

The protocol between the host and the PN531, on top of these physical links is described in §3.2, p.15.

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 3: Host interface selection

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

3.1.1.1 USB interface

A USB modelling of the PN531 is proposed below:

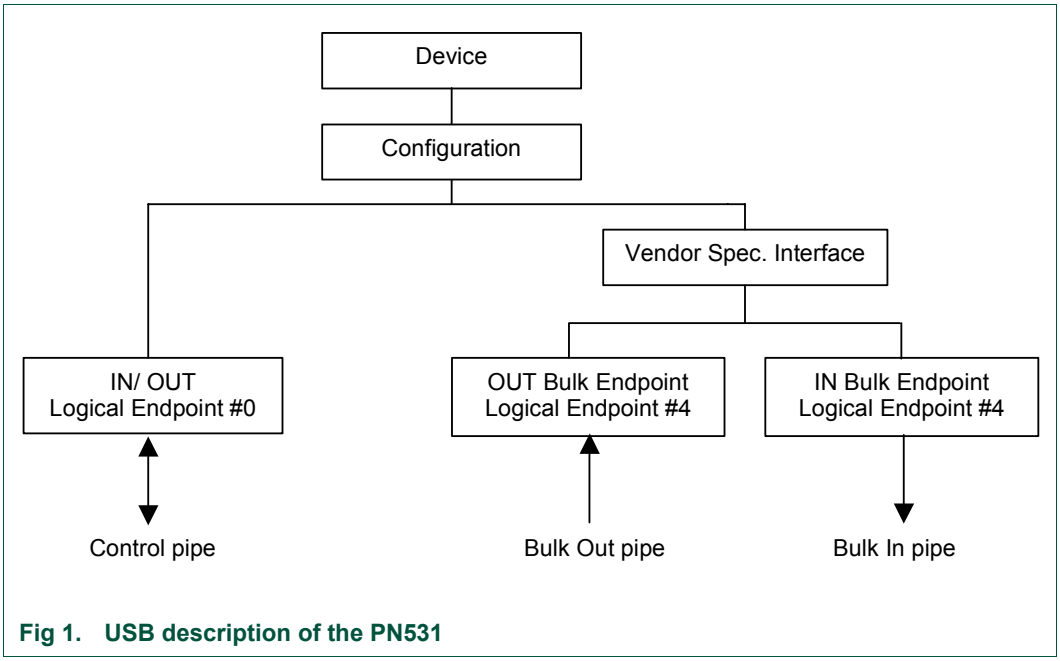


Fig 1. USB description of the PN531

The PN531 uses 2 endpoints that are part of the Vendor Specific Interface in addition to two mandatory default endpoints IN/OUT #0.

- Logical endpoint 0, **control in/out**:
  - it is needed for initializing and configuring the logical device once the device is attached and powered
  - it provides access to the device's information and allows generic USB status and control access
  - supports control transfers
  - buffer size: 8 bytes

- Logical endpoint 4, **bulk out**:
  - this endpoint performs transfers to supply data to the PN531
  - address of the endpoint: 0x04 (logical endpoint 4, type **OUT**)
  - attributes: bulk, no synchronization, data endpoint
  - buffer size: 64 bytes

- Logical endpoint 4, **bulk in**:
  - this endpoint performs transfers to retrieve data from the PN531
  - address of the endpoint: 0x84 (logical endpoint 4, type **IN**)
  - attributes: bulk, no synchronization, data endpoint
  - buffer size: 64 bytes

- Logical endpoints 1, 2 and 3, **interrupt in**:
  - not used

Device descriptor

In the device descriptor during the enumeration phase, the Vendor ID (offset 8 in the Device Descriptor) and the Product ID (offset 10 in the Device Descriptor) are sent by the PN531 to the USB controller.

Two configurations are possible depending on the pin P7.2 (during the reset phase of the IC):

Table 4: USB VID/PID selection

	P 7 . 2	
	0 Sony	**1** Philips
PID	0x0193	0x0531
VID	0x054C	0x04CC

For more information, see the PN531 data sheet (see reference [**3**]).

Remote WakeUp

The PN531 does not support Remote WakeUp feature.

Table 5: Pin used for USB interface

	PN531 Pin number
V_{BUS}	40
D+	31
D-	32

3.1.1.2 SPI interface

Refer to the PN531 data sheet (see reference [**3**]).

The PN531 is a SPI slave with the following pins used:

Table 6: Pin used for SPI interface

	PN531 Pin number
NSS	31
MOSI	32
MISO	33
SCK	34

The mode used for the clock can be chosen for the phase (CPHA bit) and the polarity (CPOL bit). (bits of *SPI_CONTROL_REGISTER*, reference [**3**]).

The pin **P3.0** is used to set the **CPHA** bit (when P3.0 is grounded, CPHA is set to 1, whereas is it set to 0 if P3.0 is level one),
The pin **P3.1** is used to set the **CPOL** bit (when P3.1 is grounded, CPOL is set to 1, whereas is it set to 0 if P3.1 is level one).

Therefore, if no GPIO is grounded, the SPI default mode is **Mode 0**.

Table 7: SPI configuration

		SPI Clock mode selection	
	SPI mode	**P3.0**	**P3.1**
- Data is always sampled on the **second** clock edge of SCK - SCK is active **low**	3	0	0
- Data is always sampled on the **first** clock edge of SCK - SCK is active **low**	2	1	0
- Data is always sampled on the **second** clock edge of SCK - SCK is active **high**	1	0	1
- Data is always sampled on the **first** clock edge of SCK - SCK is active **high**	0	1	1

The data order used is LSB first.

3.1.1.3 HSU interface

Refer to the PN531 data sheet (see reference [**3**]).

HSU interface default configuration is:

Data bit	: 8 bits
Parity bit	: none
Stop bit	: 1 bit
Baud rate	: 9600 bauds
Data order	: LSB first

Table 8: Pin used for HSU interface

	PN531 Pin number
RX	31
TX	32

3.1.1.4 I2C interface

Refer to the PN531 data sheet (see reference [**3**]).

The PN531 is a I2C slave.

The PN531 is configured with I2C address 0x48 and is able to support a clock frequency up to 400kHz.

The data order used is MSB first.

Table 9: Pin used for I2C interface

	PN531 Pin number
SCL	31
SDA	32

3.1.2 IRQ line

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

This IRQ pin may be driven automatically or not by the firmware; by default it is not the case and the system controller must use the PN531 configuration command (**`SetTAMAParameters`**, §4.2.9, p.67) to specify it.
When not used for communication purpose, this IRQ is free and may be used as a GPIO (**`ReadGPIO`** §4.2.6, p.61 and **`WriteGPIO`** §4.2.7, p.63).

This IRQ line may be very useful when using the SPI or I2C interfaces since with these two physical interfaces, the PN531 is a pure slave device and has no possibility to send back to the system controller its response frame if the host does not clock it.

The behavior of this IRQ line is described in the *Dialog structure* chapter (§3.2.2, p.19) and in the specific communications details for each link:

- USB communication details §3.2.3, p.26,
- HSU communication details §3.2.4, p.28,
- I2C communication details §3.2.5, p.30 and
- SPI communication details §3.2.6, p.33.

When configured in handshake mode, the role of this pin is different.
Please refer to the specific chapter dedicated to the handshake mode (§3.3, p.36).

3.1.3 CPU frequency

When configured in Normal mode, the PN531 CPU frequency can take three different values:

- Default CPU frequency:
  This is default value of the CPU frequency when the PN531 is neither working with the host (reception or transmission) nor exchanging data on the RF (reception or transmission).
  This value can be changed by the user (variable bDefaultFreqCPU at address 0x01F8).
- CPU frequency for RF communication:
  This is the CPU frequency used when the PN531 is exchanging data on the RF.
  This value can be changed by the user (variable bFreqCPUForRFOperation at address 0x01F9).
- CPU frequency for host link:
  This is the CPU frequency used when the PN531 is exchanging data with the host.
  This value is fixed to 27.12 MHz, **except for the USB link** where it is the same than the default CPU frequency and thus can be changed.

Table 10: Default CPU frequency used

	CPU frequency		
	Default operation *[0x01F8]*	**During Host operation** (Tx or Rx)	**During RF operation** (Tx or Rx) *[0x01F9]*
HSU	6.78 MHz	27.12 MHz	27.12 MHz
I2C			
SPI			
USB	13.56 MHz		

By writing into the variables bDefaultFreqCPU and bFreqCPUForRFOperation, it is possible to change the default CPU frequency. Use the **`WriteRegister`** command (see §4.2.5, p. 59) with value 0x02 for 6.78, 0x01 for 13.56 and 0x00 for 27.12.

3.2 Host communication protocol

3.2.1 Frames structure

Communication between the host controller and the PN531 is performed through frames, in a half-duplex mode.

Four different types of frames are used in one or the both directions (system controller to the PN531 and PN531 to the system controller).

3.2.1.1 Information frame

Information frames are used to convey:

- commands from the system controller to the PN531,
- and responses to these commands from the PN531 to the system controller.

The structure of this frame is the following:

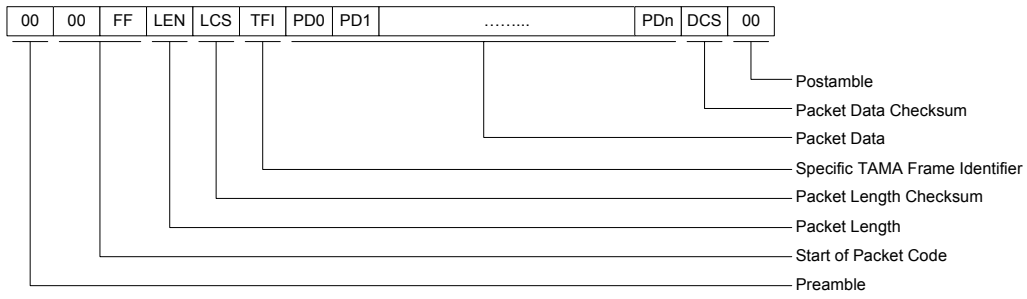


Fig 2. Information frame

- **PREAMBLE** 1 byte[2]
- **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 TAMA Frame Identifier, the value of this byte depends on the way of the message
  - **D4h** in case of a frame from the system controller to the PN531
  - **D5h** in case of a frame from the PN531 to the system 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]

[2] The preamble and postamble fields are represented here as a byte whose value is 0x00. It is always the case in the way from the PN531 to the host controller.
In the way from the host controller to the PN531, refer to each host link communication details paragraph.

3.2.1.2 ACK frame

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

This frame may be used either from the system controller to the PN531 or from the PN531 to the host controller to indicate that the previous frame has been successfully received.

ACK frame:

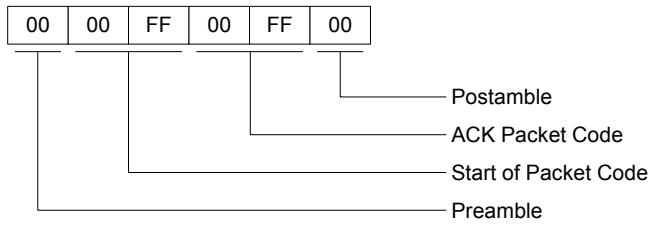


Fig 3. ACK frame

3.2.1.3 NACK frame

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

This frame is used only from the system controller to the PN531 to indicate that the previous frame has not been successfully received, then asking for the retransmission of the last frame from the PN531 to the host controller.

NACK frame:

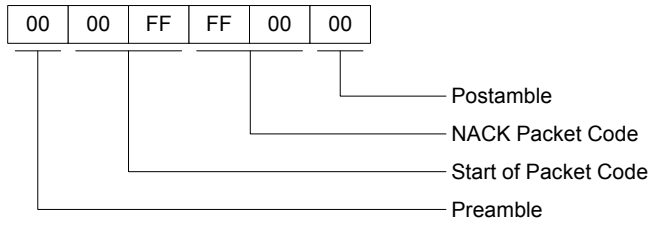


Fig 4. NACK frame

3.2.1.4 Error frame

The syntax error frame is used to inform the system controller that the PN531 has detected an error at the application level.

Error frame:

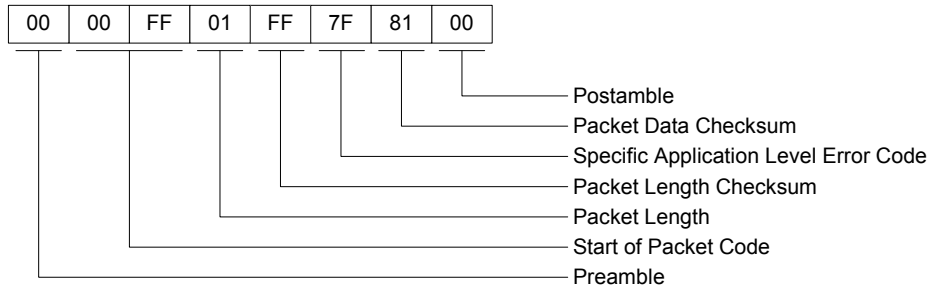


Fig 5. Error frame

3.2.1.5 Preamble and Postamble

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

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

- 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 link communication details paragraph).
  The PN531 uses this synchronization pattern (0x00 0xFF) to detect the beginning of a frame, all the previous data are ignored.

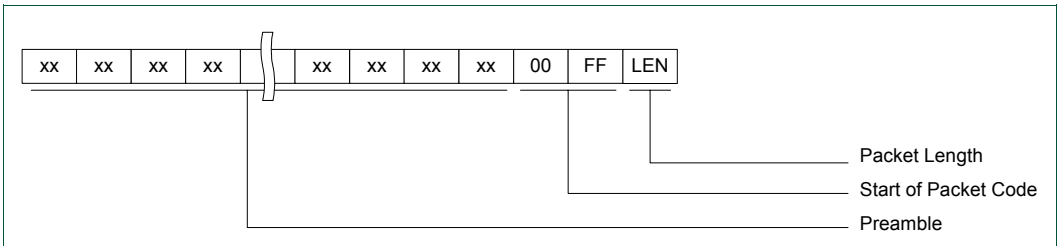


Fig 6. Preamble

- 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 link communication details paragraph).
  The PN531 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) are ignored.

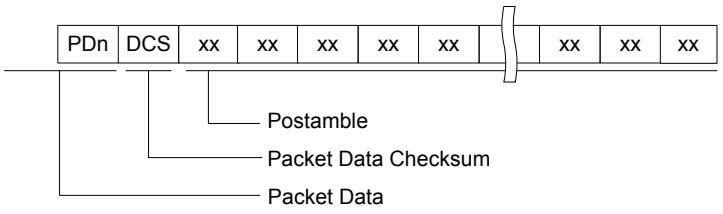


Fig 7. Postamble

- Frames sent by the PN531

  Concerning the frames sent by the PN531 to the host controller, both the preamble and the postamble are constituted of only one 0x00 byte.

- Examples

  All the following frames are the same for the PN531 point of view (**`GetFirmwareVersion`**).

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

3.2.2 Dialog structure

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

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

- it sends a command to the PN531,
- the PN531 sends back an acknowledge to inform the system controller that the command has been successfully received,
- the PN531 executes the command,
- the PN531 sends back the corresponding answer to the system controller,
- optionally, the system controller may send an ACK frame to indicate to the PN531 that the answer has been successfully received.

If the PN531 is configured to handle the IRQ line apart from the handshake communication mode, it drives this line as follows (see Fig 8 and **`SetTAMAParameters`**, §4.2.9, p.67):

- initially, the IRQ line is released (logical level one)
- when a COMMAND frame coming from the system controller is completely received, the PN531 has to send back an ACK frame. As soon as the PN531 is ready to send this ACK frame, the PN531 forces the line to level 0 to inform the system controller.
- when the ACK frame is completely sent back to the system controller, the PN531 releases the IRQ line
- then the PN531 processes the command frame and forces again the IRQ line to the level 0 when the process is completed
- when the RESPONSE frame is completely sent back to the system controller, the PN531 releases the IRQ line again and another exchange can be launched.

3.2.2.1 Data link level

a) Successful exchange at data link level

The figure below describes a normal exchange:

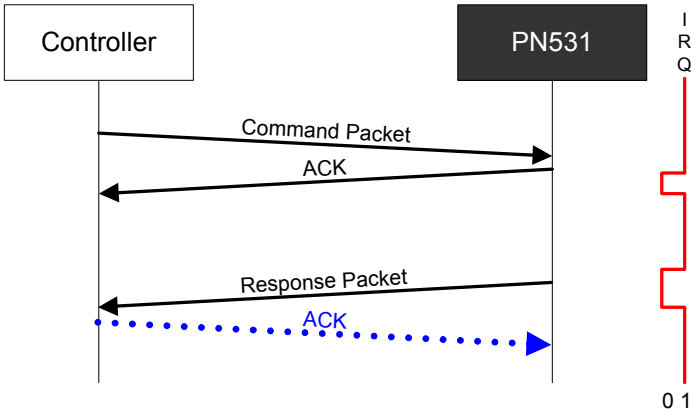


Fig 8. Data link level: normal exchange

b) Error at data link level, from host to PN531

When an error is detected by the PN531 at the data link level, it does not send back an ACK frame to the system controller.

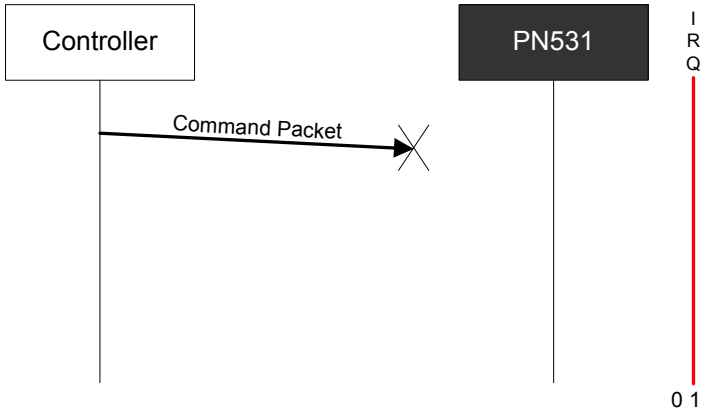


Fig 9. Data link level: error from the host to the PN531

The following errors are considered by the PN531 as data link level errors:

- LCS error
- DCS error
- Timeout error in case of HSU
  The PN531 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

- Framing error in case of HSU (stop bit is 0)

c) Error at data link level, from the PN531 to the host

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 PN531 to send again the last frame.

If used, Time Out value should be determined by the host controller to fit each command. Indeed, some commands may have a very short response time (**`ReadRegister`**, **`SetTAMAParameters`**, …), whereas other commands may response after a very long time (**`InListPassiveTarget`**, **`TgInitTAMATarget`**, …).

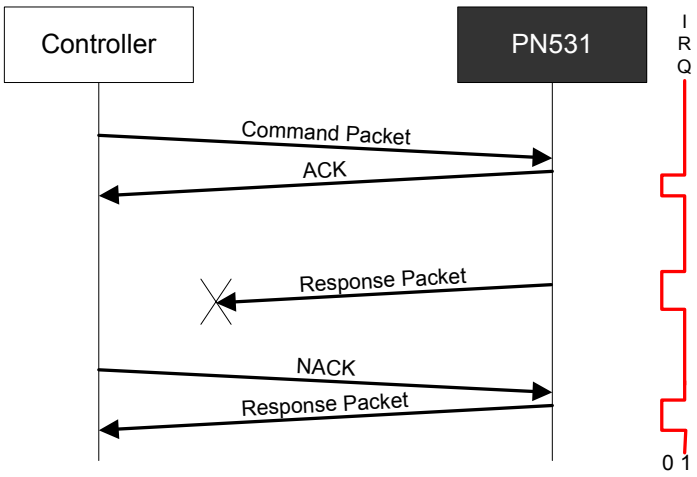


Fig 10. Data link level: error from the PN531 to the host

3.2.2.2 Application level

a) Successive exchanges

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

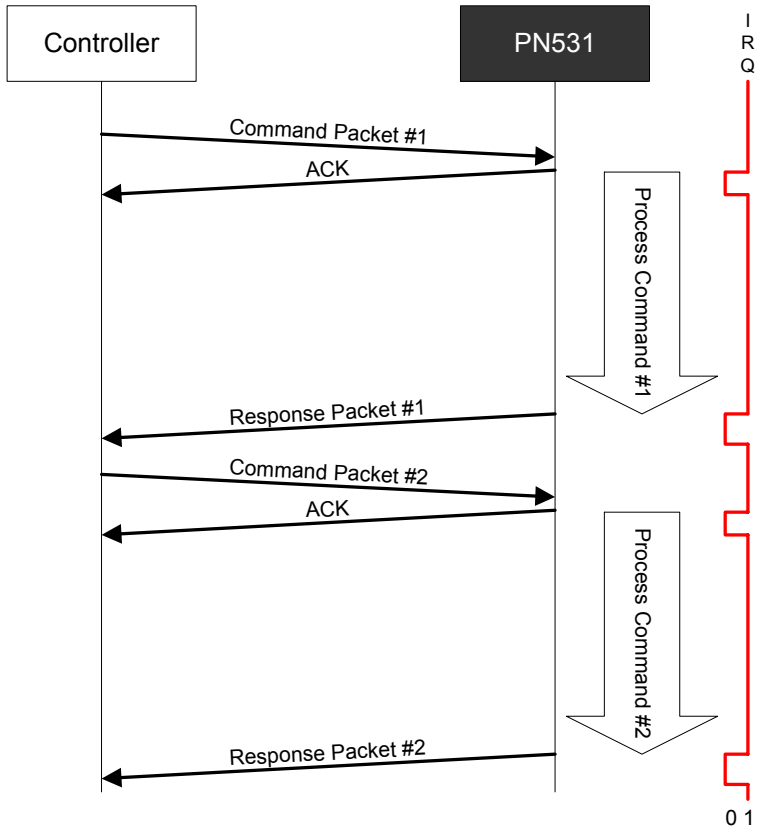


Fig 11. Application level: Successive exchanges

b) Abort

The host can force the PN531 to abort an ingoing process thanks to two different methods.

1. *Abort previous command with a ACK frame*

The host may send a ACK frame to force the PN531 to abort the current process.

In that case, the PN531 discontinues the last processing and does not answer anything to the system controller.

Then, the PN531 starts again waiting for a new command.

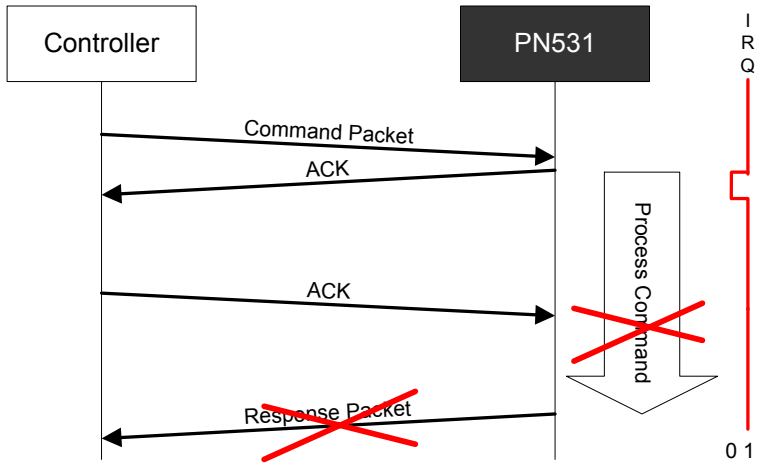


Fig 12. Data link level: Abort

2. *Abort previous command with a new command*

If the PN531 receives a newer 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.

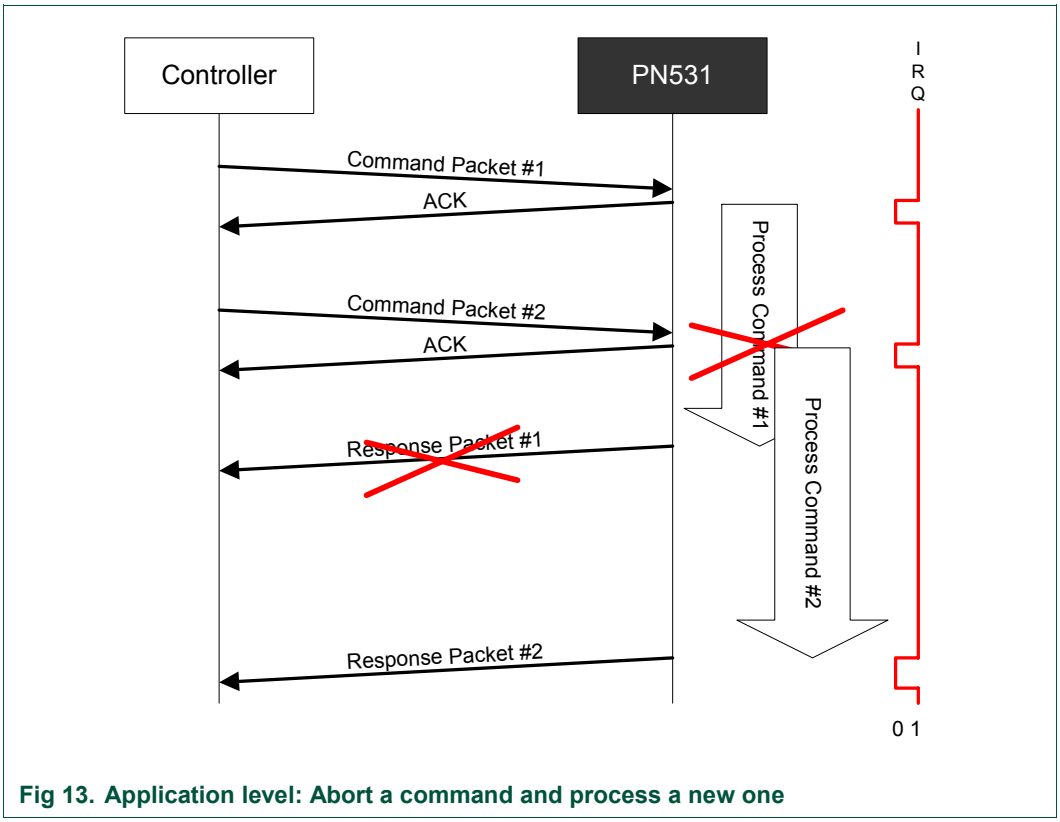


Fig 13. Application level: Abort a command and process a new one

c) Error at application level

When the PN531 detects an error at the application level, it sends back the specific “Syntax Error frame” to the system controller (see §3.2.1.4, p.17).

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

- unknown **Command Code** sent by the system controller in the command frame
- unexpected frame length
- incorrect parameters in the command frame

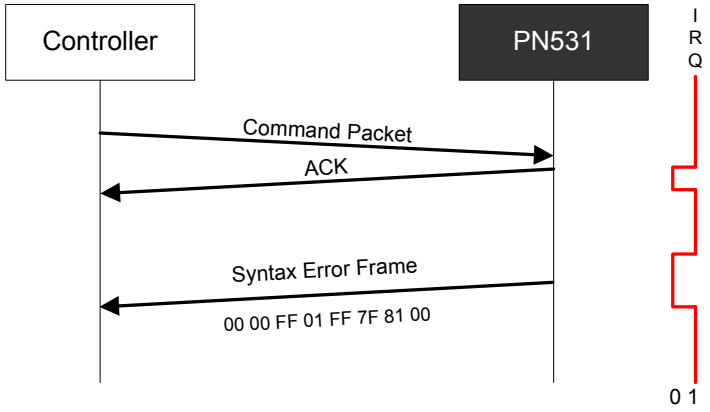


Fig 14. Application level: Error detected

3.2.3 USB communication details

The USB device interface of the PN531 is built around:

- a Control Endpoint 0 (8 bytes IN/ 8 bytes OUT),
- a BULK IN Endpoint (64 bytes),
- a BULK OUT Endpoint (64 bytes).

The command is sent by the system controller over the BULK OUT endpoint and the response is received in the BULK IN endpoint.

The host polls the BULK IN after BULK OUT has been sent.

The frames used when communicating with the USB are exactly the same as defined in the previous paragraphs §3.2.1:

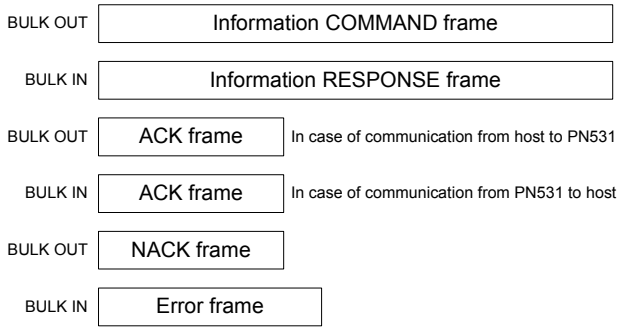


Fig 15. USB link: frames

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

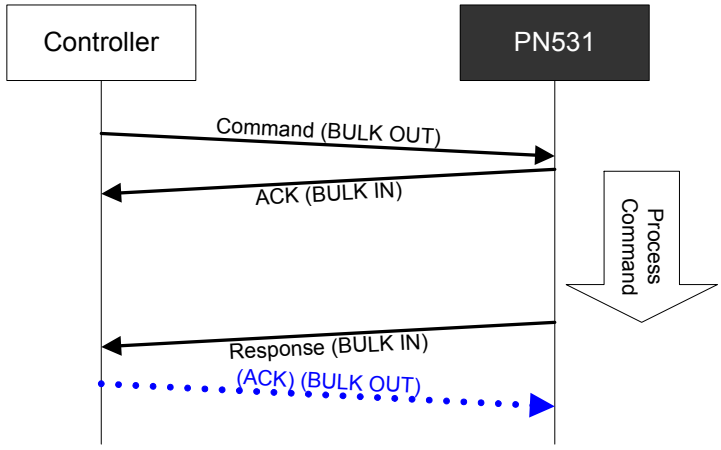


Fig 16. USB link: general principle of communication

Preamble and Postamble:

In the way from the host controller to the PN531, both the preamble and postamble fields have to be composed of a single byte with value of 0x00.

3.2.4 HSU communication details

The HSU interface of the PN531 is a full duplex serial port capable of communicating with a system controller with a baud rate up to 1.288 Mbaud.

The command is sent by the system controller over the PN531 RX line and the response is received on the PN531 TX line.

The frames used when communicating with the HSU are exactly the same as defined in the previous paragraphs §3.2.1:

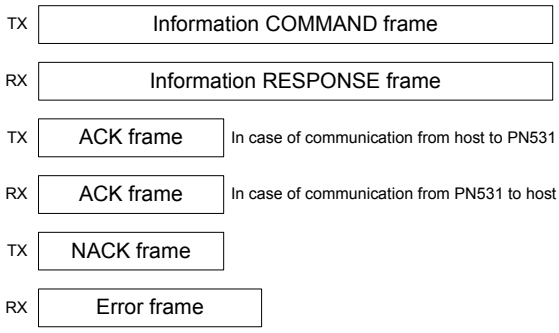


Fig 17. HSU link: frames

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

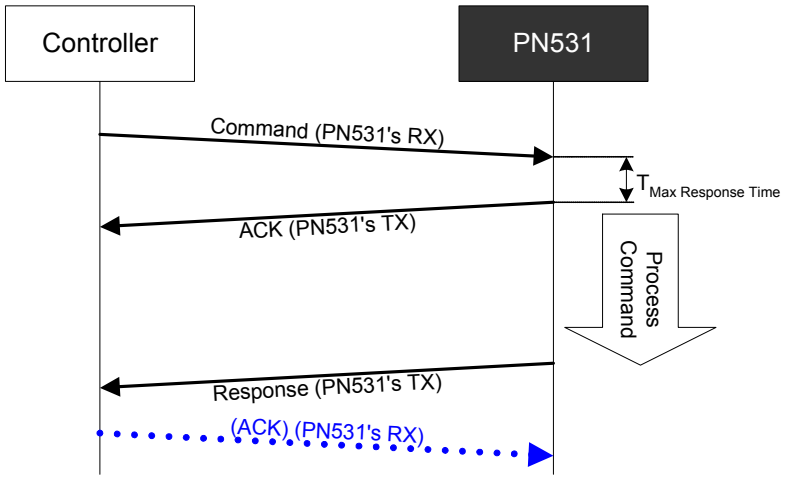


Fig 18. HSU link: general principle of communication

The PN531 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 does not detect an ACK frame within these 15 ms, the host should resend the same command frame.

Preamble and Postamble:

In the way from the host controller to the PN531, both the preamble and postamble fields may have a length different from one byte (0 to n) and the content is unimportant.

3.2.5 I2C communication details

The I2C interface of the PN531 is compliant with the I2C bus specification.
The PN531 is configured as slave (address 0x48) and is able to communicate with a system 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 §3.2.1.

Some modifications are added due to the necessity of synchronization.

The PN531 indicates to its host controller that it is ready to operate or not on the I2C bus by using the I2C acknowledge mechanism.
It then indicates that a response to the host is available or not with the following status byte:

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

- when bit **RDY = 0**, the PN531 has no frame available to be transferred to the system controller
- when bit **RDY = 1**, the PN531 has a frame available to be transferred to the system controller

The different frames are modified as follows:

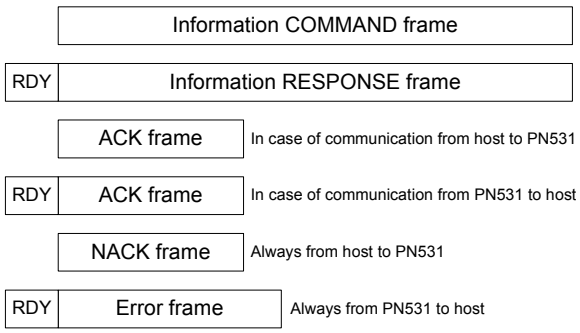


Fig 19. I2C link: frames

When the system controller wants to read data from the PN531, 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 must close the communication by sending a **I2C STOP** condition.

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

If this byte indicates that the PN531 is not available, a **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 PN531 indicates that it is ready, the rest of the frame can be read before sending a **I2C STOP** condition.

The following figure depicts this mechanism:

- COMMAND sent by the host controller
- the system controller polls the Status byte (using its own frequency)
- ACK frame “sent” by the PN531
- polling of the Status byte by the system controller
- RESPONSE frame “sent” by the PN531
- Optional ACK sent by the host controller

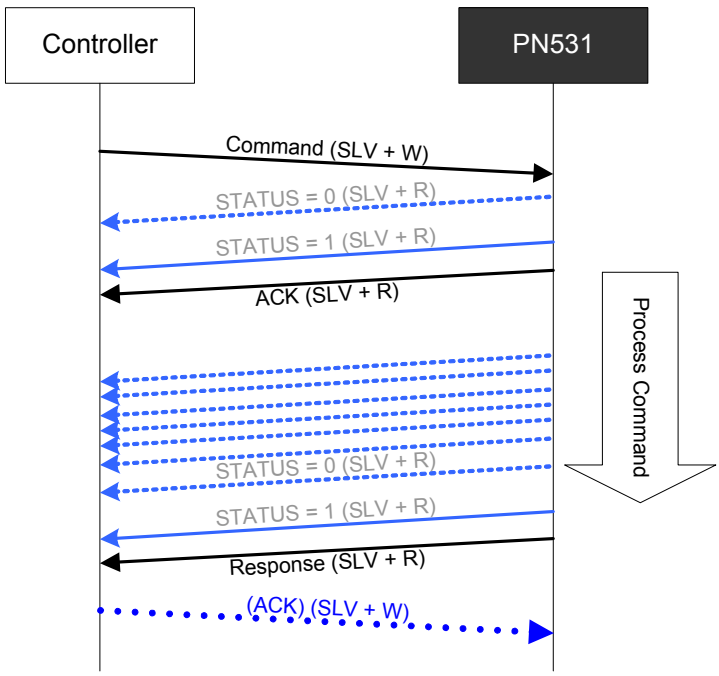


Fig 20. I2C link: general principle of communication

Legend used:

SLV + W: write operation at PN531 address (0x48)

SLV + R: read operation at PN531 address (0x49)

A better way for the system controller is to use the IRQ line that indicates when the PN531 is ready to send its frame.

In that case, the host can wait for this line to be asserted by the PN531 before to read the status byte. As a consequence, the overall traffic on the I2C bus is reduced.

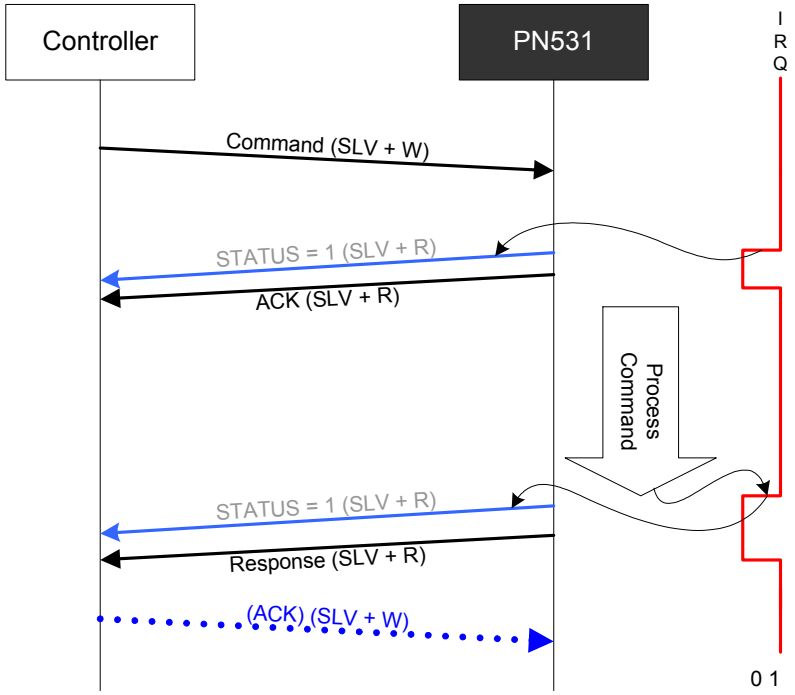


Fig 21. I2C link: using IRQ line

Preamble and Postamble:

In the way from the host controller to the PN531, both the preamble and postamble fields may have a length different from one byte (0 to n) and the content is unimportant.

3.2.6 SPI communication details

The SPI interface of the PN531 is compliant with the SPI bus specification.

The PN531 is configured as a slave and is able to communicate with a system controller with a clock (SCK) up to 5MHz.

The SPI interface includes a specific register allowing the host to know if the PN531 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 PN531 has no frame available to be transferred to the system controller
- when bit **RDY = 1**, the PN531 has a frame available to be transferred to the system controller

As a consequence, the frames used when communicating with the SPI are slightly modified compared to those defined in the paragraph §3.2.1.

Before initiating an exchange (either from the system controller to the PN531 or from the PN531 to the system controller), the system controller must write a byte indicating to the PN531 what is the following operation:

- **first byte = xxxx xx10b**, status reading (PN531 to host)
- **first byte = xxxx xx01b**, data writing (host to PN531)
- **first byte = xxxx xx11b**, data reading (PN531 to host)

The different frames are modified as follows:

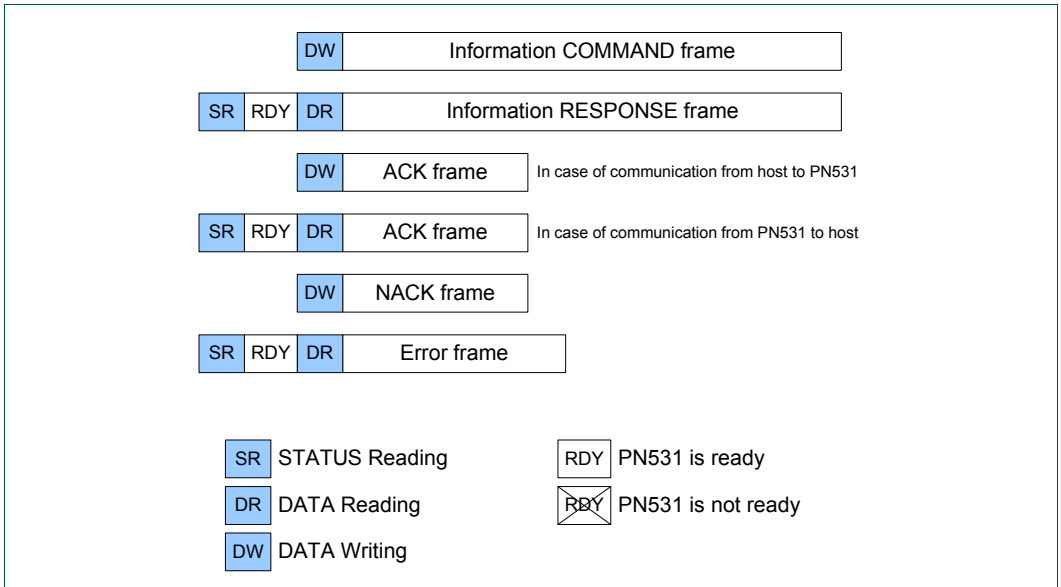


Fig 22. SPI: frames

When the system controller wants to read data from the PN531, it has to read first the status byte and as long as the RDY bit is not equal to 1, it has to retry:

The following figure depicts this mechanism:

- COMMAND sent by the host controller
- the system controller polls the Status byte using its own frequency
- ACK frame “sent” by the PN531
- polling of the Status byte by the system controller
- RESPONSE frame “sent” by the PN531
- Optional ACK sent by the host controller

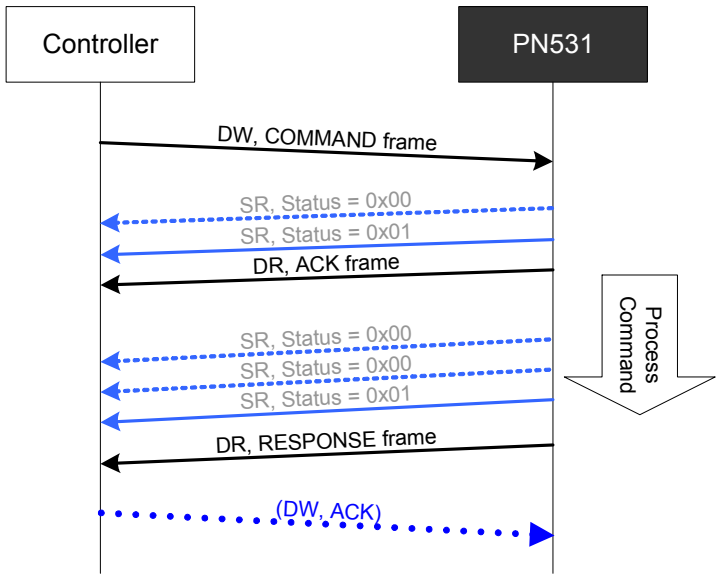


Fig 23. SPI: general principle of communication

A better way for the system controller is to use the IRQ line that indicates when the PN531 is ready to send its frame.

In that case, the host can wait for this line to be asserted by the PN531 and has no more need to read the status byte. As a consequence, the overall traffic on the SPI bus is reduced.

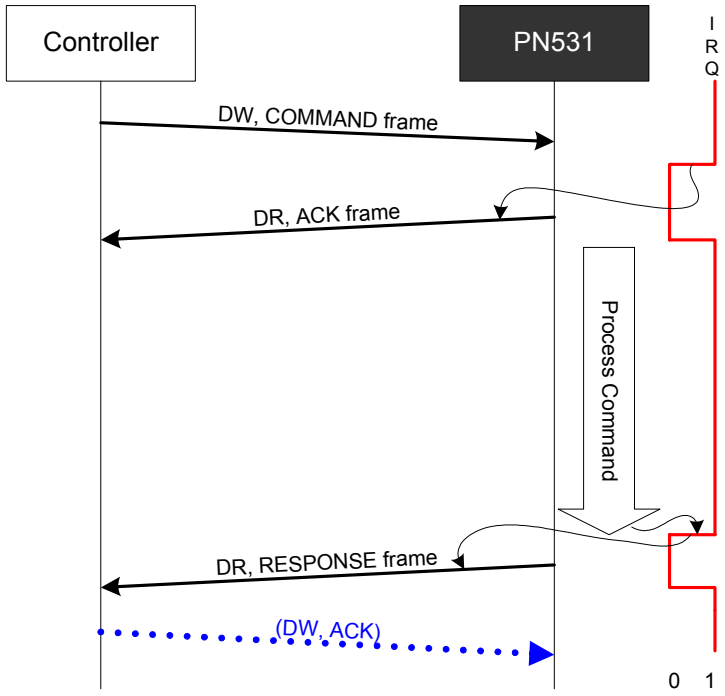


Fig 24. SPI link: using IRQ line

Preamble and Postamble:

In the way from the host controller to the PN531, both the preamble and postamble fields have to be composed of a single byte with value of 0x00.

3.3 Handshake mechanism

3.3.1 General presentation

Optional handshake mechanism is available for both HSU and I2C links. The configuration of this mode is done during boot sequence of the IC with pin P31:

Table 12: Handshake selection

Handshake	Selection Pin P31 (pin #26)
Handshake is selected	0
Normal mode	1

The goals of the handshake mechanism are:

- for the host controller, to wake the PN531 up (if it was asleep),
- for the PN531, to wake the host controller up (if it was asleep),
- for the host controller, to reduce the traffic on I2C 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 PN531 and the host controller are frequently in power down mode.

There are two possibilities for the PN531 to be in power down mode:

- use the **`PowerDown`** command (§4.2.11, p. 77).
- use the **`TgInitTAMATarget`** command (§4.3.13, p.118). After having received this command and sent the ACK frame, the PN531 goes automatically into power down mode if no external RF field is detected.

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

3.3.2 Handshake mechanism in case of HSU link

The handshake/wake-up mechanism is based on a four lines interface between the host processor and the PN531:

- **T_R_X** : Serial reception line of the PN531
- **T_T_X** : Serial transmission line of the PN531
- **H_REQ** : Request or acknowledge line from the host (connected to the P32 line of the PN531)
- **IRQ** : PN531 request line (IRQ for the host)

3.3.2.1 Normal case

When neither the PN531 nor the host controller are 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 25.

In both cases (H_REQ asserted or not), the PN531 reacts in the same way, as described below.

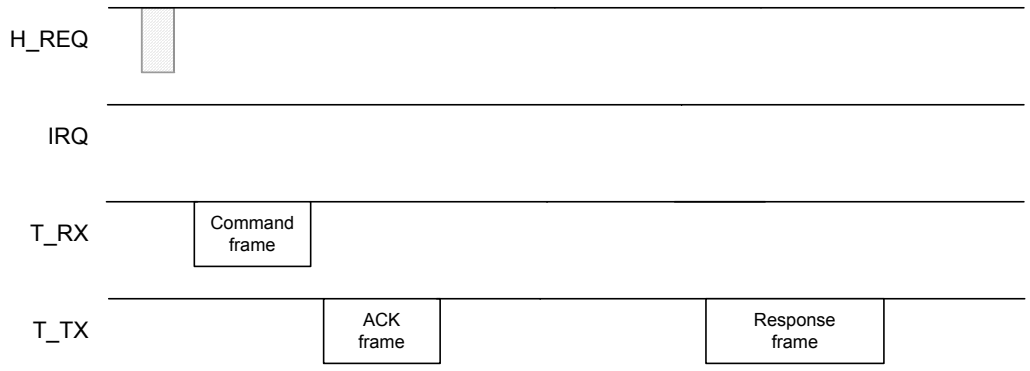


Fig 25. Handshake in case of HSU link – case 1

3.3.2.2 Case of PN531 in power down mode

The PN531 is in power down mode and the host controller sends a new command.

Depending on the enabled wake up sources (see **`PowerDown`** command, §4.2.11, p.77), the host can wake it up either:

- by generating a pulse on the H_REQ line (minimum duration: 1µs) or
- by sending a command on T_Rx line (in that case, this command is lost from the PN531 point of view).

In both cases, the PN531 does not inform that it is awaken, then the host has to wait for a minimum delay of ~200µs before sending a new command that will be properly understood.

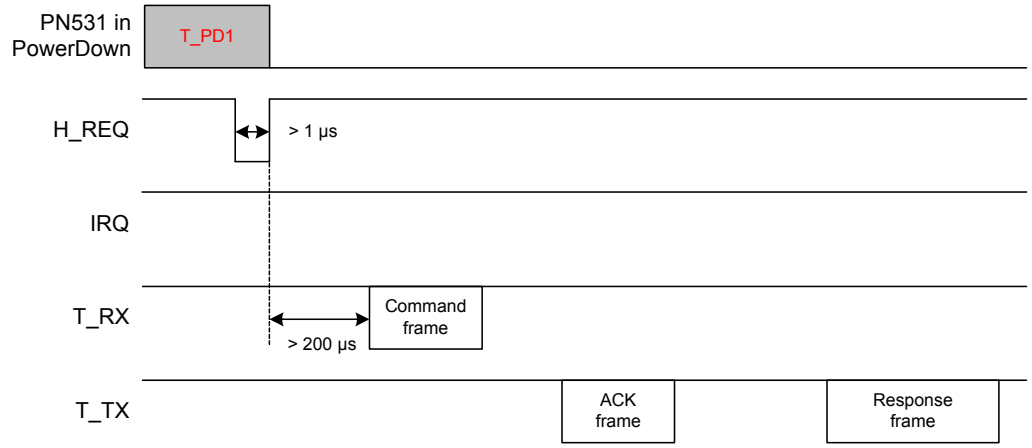


Fig 26. Handshake in case of HSU link – case 2

3.3.2.3 Case of the TgInitTAMATarget command

After having received the **`TgInitTAMATarget`** (§4.3.13, p.118) command from the host controller and if no external RF field is detected, the PN531 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 §3.3.2.2).

Once the PN531 has been activated as a target, it informs the host controller with the IRQ line.

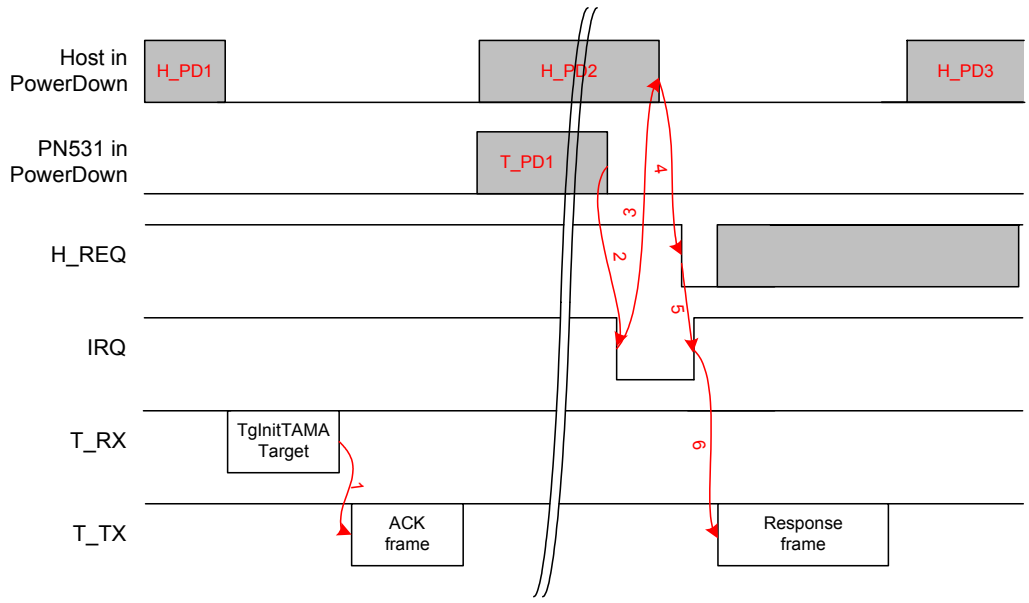


Fig 27. Handshake in case of HSU link – case 3

- The host sends the command and the PN531 acknowledges the command frame (**1**) and goes in power down **(T_PD1)**.
- Then the host controller may go to power down **(H_PD2)**.
  The PN531 is woken up by the detection of an external RF field.
- As soon as the PN531 has been activated, it is ready to send back the response to the host and it pulls IRQ line to low (**2**). If the host was asleep, it will wake it up (**3**). Then the host asserts its H_REQ line to acknowledge (**4**).
- After having detected a falling edge on H_REQ, the PN531 releases the IRQ line to high (**5**) and sends the response frame to the host (**6**).
- At the end of the exchange, the host controller may return into power down mode **(H_PD3)**.

3.3.2.4 Case of SAMConfiguration – Virtual Card

Combined with the Virtual Card mode (configured with the **`SAMConfiguration`** command (see §4.2.10, p. 70)), the **`PowerDown`** command allows reducing power consumption 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 PN531 asserts the IRQ line (**1**), the host is waken up (**2**) (if it was asleep **H_PD1**).
- Due to the fact that this IRQ negative pulse occurs outside a standard exchange initiated by the host controller, the host controller shall send a **`GetGeneralStatus`** command. It shall acknowledge the IRQ pulse with a H_REQ falling edge (**2**). The IRQ line is then set to 1 by the PN531 within #5µs (**3**), and the host controller shall set the H_REQ line to 1 as well, but there is no maximum delay for that (**4**).

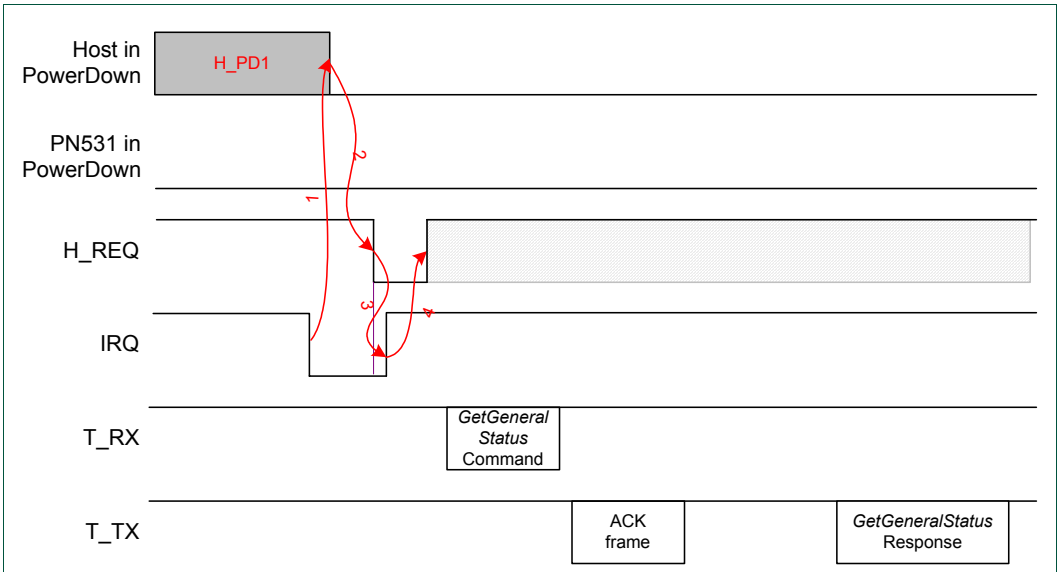


Fig 28. Handshake in case of HSU link – case 4

Example:

SAMConfiguration (Virtual Card, no timeout) ➔ D4 14 02 00
PowerDown (INT0 + RFLevelDetector) ➔ D4 16 09
….

Then, the host controller waits for IRQ being asserted by the PN531 indicating that something happened with the SAM.

GetGeneralStatus () ➔ D4 04

3.3.3 Handshake mechanism in case of I2C link

The handshake/wake-up mechanism is based on a four lines interface between the host processor and the PN531:

- **SDA** : I2C data line
- **SCL** : I2C clock line
- **H_REQ** : Request or acknowledge line from the host (connected to the PN531 P32 line)
- **IRQ** : PN531 request line (IRQ for the host)

3.3.3.1 Normal case

When neither the PN531 nor the host controller are in power down mode, the host controller does not need to assert the H_REQ line.

The PN531 generates a IRQ signal to inform that the answer is ready.

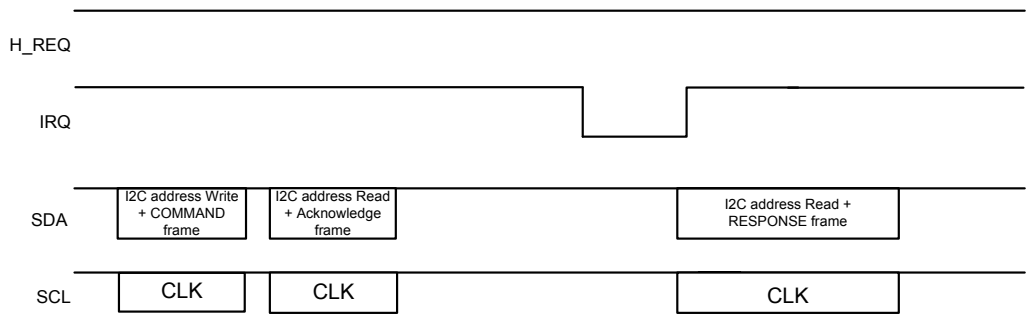


Fig 29. Handshake in case of I2C link – case 1

Remark:

However, if the host controller starts an exchange asserting the H_REQ signal, the PN531 will react as described below.

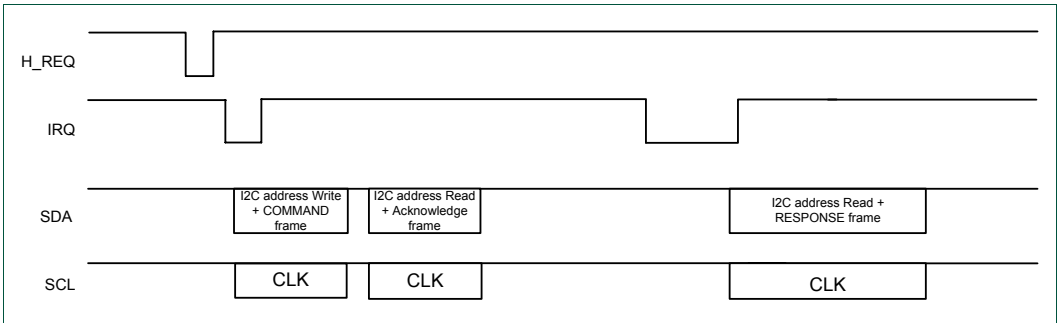


Fig 30. Handshake in case of I2C link – case 1bis

3.3.3.2 Case of PN531 in power down mode

The PN531 is in power down mode and the host controller sends a new command.

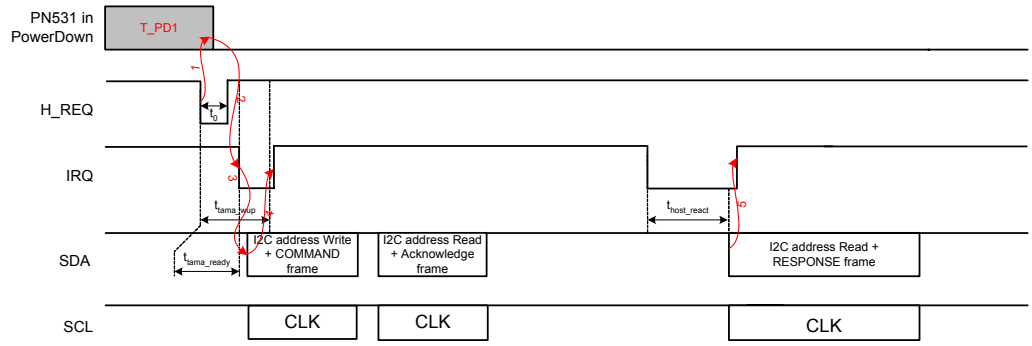


Fig 31. Handshake in case of I2C link – case 2

- The PN531 is in power down mode **(T_PD1)**.
  The host wants to send a frame to the PN531, so it asserts the H_REQ line. That makes the PN531 waking up (**1**) and it acknowledges with IRQ line (**2**).
- Then before sending the command frame (**3**), the host has 2 possibilities:
  - either the host waits for a defined delay (t_{tama_wup}) which guarantees in that case that the PN531 will be waken up when the I2C frame will be sent,
  - or the host monitors the IRQ line to check when the PN531 is woken up (with a maximum timeout of t_{tama_wup}).
    This option can optimize the overall waiting time if the PN531 was not asleep; there is no need to wait the maximum wakeup time ($t_{tama\_ready} < t_{tama\_wup}$).
- The PN531 sets back IRQ line after having recognized its I2C slave address (**4**).
- The PN531 acknowledges the command frame (ACK frame).
- As soon as the PN531 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 PN531. The detection of its own I2C address makes the PN531 releasing the IRQ line (**5**).

Timings:

- t_0 : The minimum duration of this pulse depends on the internal state of the PN531 (awake or asleep):
  - PN531 awake: $t_0 \geq$ **1µs**
  - PN531 asleep: $t_0 \geq$ **175µs**
- t_{tama_ready} :The delay between the falling edge of the H_REQ pulse and the falling edge of the IRQ line represents the minimum delay the PN531 needs to be ready. This delay depends on the internal state of the PN531 and of the CPU frequency. Typical values are:
  - PN531 awake: $t_{tama\_ready} \leq$ **10µs**
  - PN531 asleep: $t_{tama\_ready} \leq$ **170µs**

3.3.3.3 Case of the TgInitTAMATarget command

After having received the **`TgInitTAMATarget`** (§4.3.13, p.118) command from the host controller and if no external RF field is detected, the PN531 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 §3.3.2.2).

Once the PN531 has been activated as a target, it informs the host controller with the IRQ line.

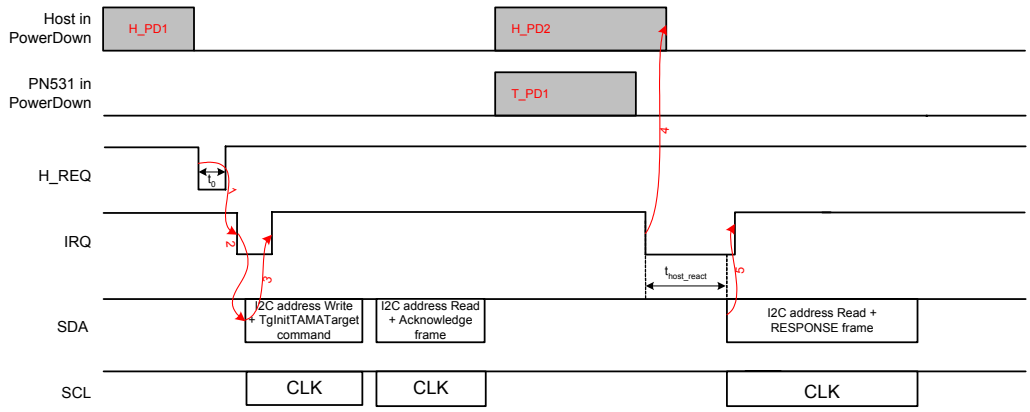


Fig 32. Handshake in case of I2C link – case 3

- The host is possibly in power down mode **(H_PD1)**.
  The host wants to send a frame to the PN531, so it asserts the H_REQ line (**1**).
- The host monitors the IRQ line to check when the PN531 is ready (**2**).
- The PN531 sets back IRQ line after having recognized its I2C slave address (**3**).
- The PN531 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 PN531 has been activated, it is ready to send back the response to the host and it asserts its IRQ line. If the host was asleep, it will wake it up (**4**).
- Once ready (t_{host_react}), the host controller gets back the answer frame from the PN531. The detection of its own I2C address makes the PN531 releasing the IRQ line (**5**).

3.3.3.4 Case of SAMConfiguration – Virtual Card

Combined with the Virtual Card mode (configured with the **`SAMConfiguration`** command (see §4.2.10, p. 70)), the **`PowerDown`** command allows to save power consumption 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 PN531 asserts the IRQ line to inform the host controller that something occurred at the SAM side (**1**), the host is waken up (if it was asleep **H_PD1**).
- Due to the fact that this IRQ negative pulse occurs outside a standard exchange initiated by the host controller, the host controller shall send a **`GetGeneralStatus`** command.
- Then the rest of the sequence is the same as described in the Fig 29.

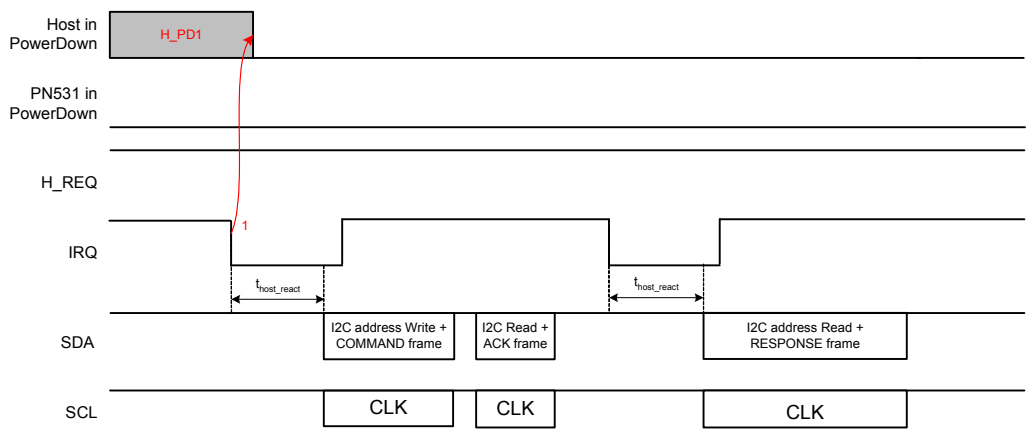


Fig 33. Handshake in case of I2C link – case 4

Example:

SAMConfiguration (Virtual Card, no timeout) ➔ D4 14 02 00
PowerDown (INT0 + RFLevelDetector) ➔ D4 16 09
….

Then, the host controller waits for IRQ being asserted by the PN531 indicating that something happened with the SAM.

GetGeneralStatus () ➔ D4 04

4. Commands supported

The following description of the commands details:

- the frame structure[3] including the type and amount of data:
  - that the host controller has to deliver to the PN531 (***Input***)
  - that the PN531 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 PN531 column indicates if the command may be used with the PN531 configured as an initiator or/and with the PN531 configured as a 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 system controller to the PN531.

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 PN531 as an initiator (respectively Target) a **In** prefix has been added (respectively **Tg** prefix)

Table 13: Command set

Command	PN531 as Initiator	PN531 as Target	Command Code	Page
Miscellaneous				
Diagnose	**X**	**X**	0x00	50
GetFirmwareVersion	**X**	**X**	0x02	53
GetGeneralStatus	**X**	**X**	0x04	54
ReadRegister	**X**	**X**	0x06	57
WriteRegister	**X**	**X**	0x08	59

[3] The frame representation does not include the complete frame, but only the following field:

- TFI
- Command Code
- when needed, the input or output data.

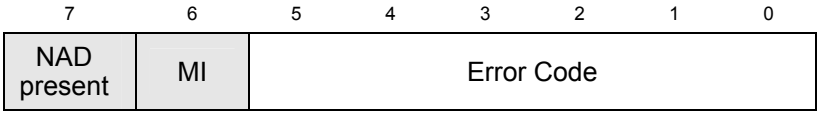
Thus, the Preamble, Start of Packet, Packet Length, LCS, DCS and Postamble are omitted in this description.

Command	PN531 as Initiator	PN531 as Target	Command Code	Page
ReadGPIO	X	X	0x0C	61
WriteGPIO	X	X	0x0E	63
SetSerialBaudRate	X	X	0x10	65
SetTAMAParameters	X	X	0x12	67
SAMConfiguration	X	X	0x14	70
PowerDown	X	X	0x16	77
RF communication				
RFConfiguration	X	X	0x32	80
RFRegulationTest			0x58	85
InJumpForDEP	X		0x56	86
InJumpForPSL	X		0x46	91
InListPassiveTarget	X		0x4A	94
InATR	X		0x50	98
InPSL	X		0x4E	100
InDataExchange	X		0x40	102
InCommunicateThru	X		0x42	111
InDeselect	X		0x44	114
InRelease	X		0x52	115
InSelect	X		0x54	116
TgInitTAMATarget		X	0x8C	118
TgSetGeneralBytes		X	0x92	125
TgGetDEPData		X	0x86	127
TgSetDEPData		X	0x8E	130
TgSetMetaDEPData		X	0x94	131
TgGetInitiatorCommand		X	0x88	133
TgResponseToInitiator		X	0x90	135
TgGetTargetStatus		X	0x8A	137

4.1 Error handling

In some of the commands detailed hereafter, there is a status byte returned by the PN531 reflecting if the RF communication has been successful or not.

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 PN531 answer frame for the **`InDataExchange`** or **`TgGetDEPData`** contain a NAD byte.
  See §4.2.9: **`SetTAMAParameters`**, p.67.

- The **MI** bit informs the host controller that the PN531 configured as a 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 and by the target in examples given in §4.4.5: Chaining mechanism, p.143.

- 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 14: Error code list

Error cause	Error code
Time Out, the target has not answered	0x01
A CRC error has been detected by the contactless UART	0x02
A Parity error has been detected by the contactless UART	0x03
During a MIFARE® anticollision/select operation, an erroneous Bit Count has been detected	0x04
Framing error during MIFARE® operation	0x05
An abnormal bit-collision has been detected during bit wise anticollision at 106 kbps	0x06
Communication buffer size insufficient	0x07
RF Buffer overflow has been detected by the contactless UART (bit BufferOvfl of the register *CL_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. reference [**3**], description of the *CL_ERROR* register)	0x0B

Error cause	Error code
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
DEP Protocol: The PN531 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, **ref. [1]**).	0x12
DEP Protocol / Mifare® / ISO14443-4: 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
ISO14443-3: 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 interface)	0x26
This command is not acceptable due to the current context of the PN531 (Initiator vs. Target, unknown target number, Target not in the good state, …)	0x27
The PN531 configured as a target has been released by its initiator	0x29

4.2 Miscellaneous commands

4.2.1 Diagnose

Input:

D4	00	NumTst	[InParam]

- **NumTst** Test number to be executed by the PN531 (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:

This command is designed for self-diagnosis.

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

The PN531 returns result (**OutParam**) with 1 to 252 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 and the PN531. “Parameter Length” and “Parameters” in response packet are as same as “Parameter Length” and “Parameter” in command packet.
  - Parameter Length : m (0 <= m <= 252)
  - Parameter : Data
  - Result Length : Same value of m.

  OutParam consists of NumTst concatened 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 PN531 sends a FeliCa™ polling command packet 128 times to target. The PN531 counts the number of fails and returns the failed number to host. 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`** (§4.3.1, p.80) with the item n°7.
  - 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 PN531 is configured in target mode. The analog settings used are those defined by using the command **`RfConfiguration`** (§4.3.1, p.80) with the item n°6. 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 PN531 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 *CL_TXMODE* register defining the baud rate and the modulation type in transmission
  - Parameter 3 : Content of the *CL_RXMODE* 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.

For example:

– the PN531 is configured to receive frame with passive 106 kbps modulation type. The frames are sent back immediately.

D4	00	05	00	80	80

– the PN531 is configured to receive frame with passive 212 kbps modulation type. The frames are sent back with a delay of 64 ms.

D4	00	05	80	92	92

– the PN531 is configured to receive frame with passive 424 kbps modulation type. The frames are sent back immediately.

D4	00	05	00	A2	A2

- **NumTst = 0x06** : Attention Request Test
  This test can be used by an initiator to ensure that a DEP target is still in the field. 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 no or incorrect response, the Result informs about the status of the transaction (refer. to §4.1, p.48)

  – Parameter Length : 0
  – Result Length : 1
  – Result : 0x00 ➔ OK,
    different from 0x00 ➔ not OK, Status byte

4.2.2 GetFirmwareVersion

Input:

D4	02

Output:

D5	03	Ver	Rev

- **Ver** Version of the firmware
- **Rev** Revision of the firmware

Description:

The PN531 sends back the version of the embedded firmware.

In the case of the PN531/C2, the version is 4.2.

That leads to **Ver : 0x04**
Rev : 0x02

4.2.3 GetGeneralStatus

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 PN531.
- **Field** indicates if an external RF field is present and detected by the PN531 (**Field = 0x01**) or not (**Field = 0x00**).
- **NbTg** is the number of targets currently controlled by the PN531 acting as initiator.
- For each target controlled by the PN531 (maximum 2 targets):
  - $\mathbf{Tg_i}$: logical number
  - $\mathbf{BrRx_i}$: bit rate in reception
    - 0x00 : 106 kbps
    - 0x01 : 212 kbps
    - 0x02 : 424 kbps
  - $\mathbf{BrTx_i}$: bit rate in transmission
    - 0x00 : 106 kbps
    - 0x01 : 212 kbps
    - 0x02 : 424 kbps
  - $\mathbf{Type_i}$: modulation type
    - 0x00 : Mifare®
    - 0x10 : FeliCa™
    - 0x01 : Active mode
- **SAM status** informs on the status of the SAM connection.

Description:

This command allows the host controller to know at a given moment the complete situation of the PN531.

Err:

Err contains error code as defined in the error code paragraph (§4.1, p.48). After the execution of the **`GetGeneralStatus`** command, the content of the latest error is cleared.

Field:

The PN531 scans the RF field to inform the host if an external field is detected or not.

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

When the PN531 is configured as an initiator, for all the targets handled by the PN531, the indication of the baud rate used and the modulation is given. The **Tg_i** information corresponds to the logical target number attributed by the PN531 when a previous **`InListPassiveTarget`**, **`InJumpForDEP`** or **`InJumpForPSL`** command has been used[4].

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 PN531.

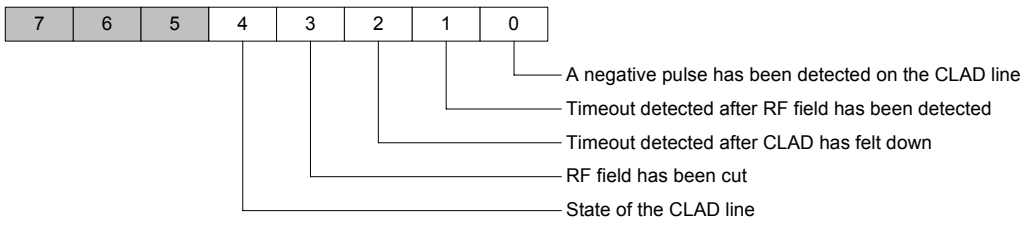


Fig 34. SAM status byte definition

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

When **bit 1** is set to 1, a timeout has been detected after the detection of an external RF field.

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

When **bit 3** is set to 1, an external RF field has been cut. This bit is significant only when the PN531 is configured in virtual card mode with a timeout value different from 0.

[4] 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.

When **bit 4** 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, this bit is not significant. In other words, for example when the PN531 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.

For more information, refer to §4.2.10, p.70.

4.2.4 ReadRegister

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 detected

Description:

This command is used to read the content of one or several internal registers of the PN531 (located either in the SFR area or in the XRAM memory space).

For each address **ADR**, the PN531 performs a reading operation in the register located at address **ADR**. Then the value is returned in the VAL parameter.

- SFR registers
  The host 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 15: List of SFR registers

Address	Register	Address	Register
0x87	PCON	0xAE	HSU_COUNTER
0x9A	RWL	0xB0	P3
0x9B	TWL	0xB8	IP
0x9C	FIFOFS	0xD1	CL_COMMAND
0x9D	FIFOFF	0xD7	P5
0x9E	SFF	0xE8	IEN1
0x9F	FIT	0xF4	P7CFGA
0xA1	FITEN	0xF5	P7CFGB
0xA2	FDATA	0xF7	P7
0xA3	FSIZE	0xF8	IP1
0xA8	IE	0xFC	P3CFGA
0xA9	SPI_CONTROL	0xFD	P3CFGB
0xAA	SPI_STATUS		
0xAB	HSU_STATUS		
0xAC	HSU_CONTROL		
0xAD	HSU_PRESCALER		

- XRAM memory mapped registers
  The complete address is given by the high and low bytes of address.
  See reference [**3**].

Example:

The host reads the value 0xF0 in the register *RX_THRESHOLD* located at address 0x6308 and the value 0x55 in the P3CFGA register (SFR) located at address 0xFC.

The frame from the system controller to the PN531 is:

D4	06	63	08	FF	FC

and the one returned by the PN531 is:

D5	07	F0	55

4.2.5 WriteRegister

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 detected

Description:

This command is used to overwrite the content of one or several internal registers of the PN531 (located either in the SFR area or in the XRAM memory space).

For each address **ADR**, the PN531 performs a writing operation of the value **VAL** in the register located at address **ADR**.

- SFR registers
  The host 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 (§0, p.57).
- XRAM memory mapped registers
  The complete address is given by the high and low bytes of address.
  See reference [**3**].

Example:

The host writes the value 0xF0 in the register *RX_THRESHOLD* located at address 0x6308 and the value 0xAA in the P3CFGA register (SFR) located at address 0xFC.

The frame from the system controller to the PN531 is:

D4	08	63	08	F0	FF	FC	AA

and the frame returned by the PN531 is:

D5	09

Warning:

The behavior of the PN531 may be altered by this command.

This command is only recommended for debug purposes.

4.2.6 ReadGPIO

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	P70
					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 PN531 reads the value for each port and returns the information to the host controller.

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

- **P32** corresponds to the pin P32_INT0.
  P32 can be used as a standard GPIO and is therefore not used as an external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§4.2.11, p.77), this pin can be used for the waking up.
  Moreover, when configured to use the handshake mechanism (§3.3, p.36), this pin is used for the H_REQ line.
- **P33** corresponds to the pin P33_INT1.
  P33 can be used as a standard GPIO and is therefore not used as an external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§4.2.11, p.77), this pin can be used for the waking up.
- When configured to use the SAM companion chip (**`SAMConfiguration`**, §4.2.10, p.70), **P34** is used for the CLAD line.
- **P70** corresponds to the pin named IRQ.
  The value read for this bit is not relevant when:
  - the IRQ line is used for the host exchange protocol (**`SetTAMAParameters`**, §4.2.9, p.67),
  - the IRQ line is used with the handshake mechanism (§3.3, p.36).
- **P71** and **P72** are the GPIOs corresponding to the pins MISO and SCK of the SPI bus. They can be used as GPIO when the PN531 is not configured to use the SPI interface to communicate with the host controller.

- **P72** is the GPIO used to choose the PID/VID (Philips or Sony) at the power up of the IC when USB interface is used.
- **I0** and **I1** are the GPIOs used also to configure the host interface to be used. Once the selection has been done by the firmware, these two pins can be used as GPIOs.

4.2.7 WriteGPIO

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	P70
7					2	1	0

Output:

D5	0F

Description:

The PN531 applies the value for each port that is validated by the host controller (bit **Val** of each port).

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`** (§4.2.6, p.61), the GPIO may be used with the following limitations of usage:

- **P32** corresponds to the pin P32_INT0. It can be used as a standard GPIO and is therefore not used as an external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§4.2.11, p.77), this pin can be used for the waking up.
  Moreover, when configured to use the handshake mechanism (§3.3, p.36), this pin is used for the H_REQ line.
- **P33** corresponds to the pin P33_INT1. It can be used as a standard GPIO and is therefore not used as an external interrupt trigger.
  Nevertheless, for the **`PowerDown`** command (§4.2.11, p.77), this pin can be used for the waking up.
- **P34** can be used as a standard GPIO
  Moreover, when configured to use the SAM companion chip (**`SAMConfiguration`**, §4.2.10, p.70), this pin is used for the CLAD line.
- **P70** corresponds to the pin named IRQ.
  The host shall not force this bit when:
  - the IRQ line is used for the host exchange protocol (**`SetTAMAParameters`**, §4.2.9, p.67),
  - the IRQ line is used with the handshake mechanism (§3.3, p.36).

- **P71** and **P72** are the GPIOs corresponding to the pins MISO and SCK 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.
- **P72** is the GPIO used to choose the PID/VID (Philips or Sony) at the power up of the IC when USB interface is used.

Example:

The host controller wants to:

- set P3.0 and P3.1
- reset P3.2, P3.3, P3.4 and P3.5
- leave P7.0, P7.1 and P7.2 unchanged

The frame from the system controller to the PN531 is:

D4	0E	83	00

and the frame returned by the PN531 is:

D5	0F

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

The frame from the system controller to the PN531 is:

D4	0E	BF	87

and the frame returned by the PN531 is:

D5	0F

4.2.8 SetSerialBaudRate

Input:

D4	10	BR

- **BR** is a byte indicating the baud rate asked by the system 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 baud rate requested is not possible (BR > 8)
- the link used is not the HSU (High Speed UART).

Description:

This command is used to select the baud rate on the serial link between the host and the PN531 (HSU).

When another link between the system controller and the PN531 is used (USB, I2C or SPI), this command is not allowed. In that case, the PN531 will inform the host of an application level error.

The PN531 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 system controller.

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

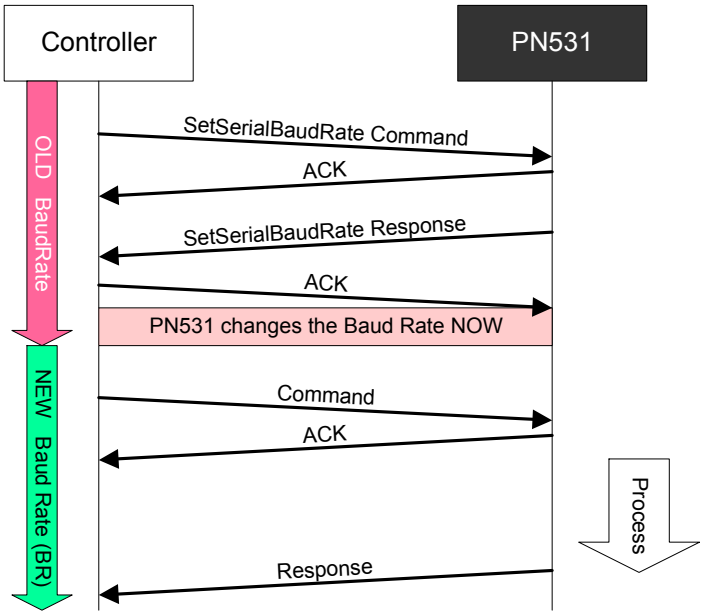


Fig 35. SetSerialBaudRate

4.2.9 SetTAMAParameters

Input:

D4	12	Flags

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

rfu	nu	nu	4	3	2	1	0

  - bit 0 : fNADUsed ➔ use of the NAD information in case of initiator configuration (DEP and ISO14443-4 RW).
  - bit 1 : fDIDUsed ➔ use of the DID information in case of initiator configuration (DEP only).
  - bit 2 : fAutomaticATR_RES ➔ automatic generation of the ATR_RES in case of target configuration.
  - bit 3 : fIRQUsed ➔ use of the IRQ line for the host protocol.
  - bit 4 : fAutomaticRATS ➔ automatic generation of the RATS in case of ISO14443-4 card R/W mode.
  - bit 7 : RFU ➔ **must** be set to 0.

Output:

D5	13

Syntax Error Conditions:

- Bit fIRQUsed set to 1 when the PN531 is in handshake communication mode.

Description:

This command is used to set internal parameters of the PN531, and then to configure its behavior regarding different cases.

fNADUsed (DEP and ISO14443-4 RW):

In DEP mode:

By default, the PN531 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 §4.3.8, p. 102).

On the opposite side, when the PN531 configured as a target is in front of an initiator using NAD byte, the NAD value received by the PN531 will be transmitted to the PN531 host controller for further analysis, and the NAD value to be returned to the initiator will be elaborated by the host controller of the PN531 (so, the NAD values are transported respectively within the **`TgGetDEPData`** and **`TgSetDEPData`** commands).

In both cases (PN531 initiator or PN531 target in Fig 36), 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 §4.1: Error handling, p. 48 and Fig 37).

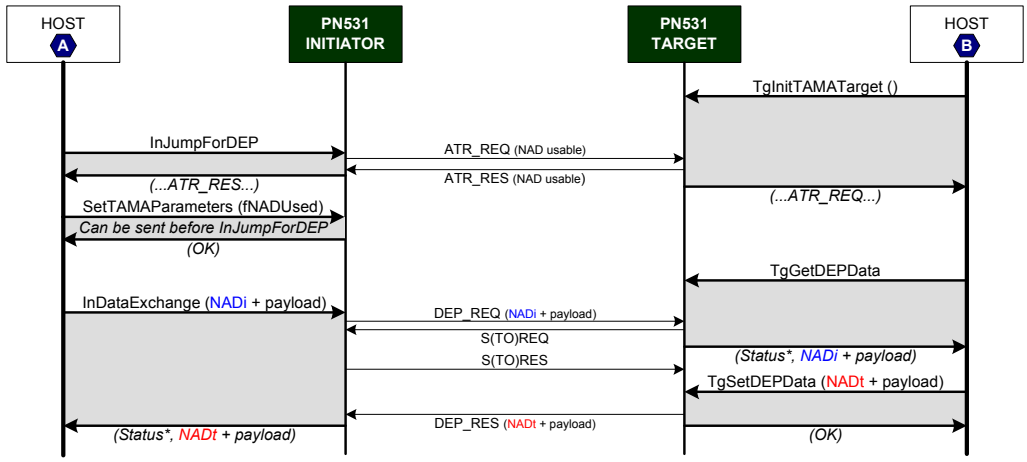


Fig 36. fNADUsed

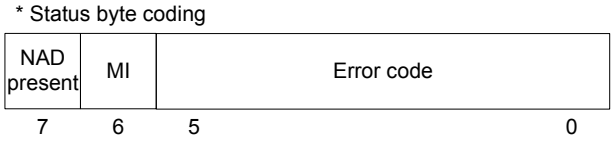


Fig 37. Status Byte definition

In ISO14443-4 RW mode:

By default, the PN531 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 §4.3.8, p. 102).

In reception mode, the NAD value received by the PN531 will be transmitted to the PN531 host controller.

fDIDUsed (DEP only):

By default, the PN531 initiator does not use the DID byte in the Transport Protocol (not multi-target configuration). So the host controller must set this flag in order to use DID. In that case, the DID value itself is completely handled internally by the firmware.
The DID has a fixed value of 0x01 when fDIDUsed is set to 1.

fAutomaticATR_RES:

By default, the PN531 target sends back to the initiator the ATR_RES after having received an ATR_REQ.
For various reasons, the host controller of the PN531 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.

Refer to **`TgSetGeneralBytes`**, §4.3.14 p.125 and **`TgInitTAMATarget`**, §4.3.13, p.118 to have a detailed description of these two possibilities.

fIRQUsed:

By default, the PN531 does not use the IRQ line (see §3.2.2, p.19).
If the host wants to use this line to be informed when an answer message is ready, it must set this flag.
This bit is not usable when handshake mode is chosen (§3.3, p.36). A Syntax error frame is returned in that case.

fAutomaticRATS:

When executing the command **`InListPassiveTarget`** at 106 kbps, the PN531 may initialise cards supporting ISO14443-4 protocol (the PN531 knows that the target is ISO14443-4 compliant by analysing the SENS-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 PN531.

If the user does not want to use the ISO14443-4 protocol with a card that is ISO14443-4 compliant, this flag must be set to 0.

Summary of the default settings:

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

Table 16: Default values of internal flags

Property	Default value
fNADUsed	Not used
fDIDUsed	Not used
fAutomaticATR_RES	Yes, automatic
fIRQUsed	Not used
fAutomaticRATS	Yes, automatic

4.2.10 SAMConfiguration

Input:

D4	14	Mode	[Timeout]

- **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 PN531+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 R/W commands (**`InListPassiveTarget`**, **`InDataExchange`**, …).
  - 0x04: **Dual Card**, both the PN531 and the SAM are visible from the external world as two separate targets.
- **Timeout** defines the time-out only in Virtual card configuration (**Mode** = 0x02). In Virtual Card mode, this field is mandatory.

Table 17: Timeout definition for the SAM Virtual Card mode

Byte Value	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

Output:

D5	15

Syntax Error Conditions:

- Incorrect values for the parameters Mode and Timeout.
- Timeout parameter missing in Virtual Card Mode.

Description:

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

A SAM companion chip can be used to bring security. It is connected to the PN531 by using a proprietary S2C interface (SigIn (pin #24), SigOut (pin #23) and CLAD (pin #22)).

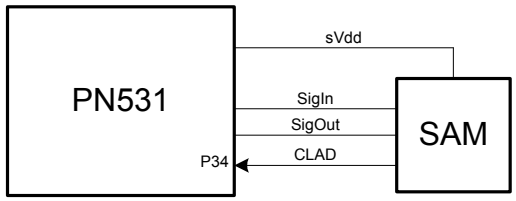


Fig 38. SAM electrical connection

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 Reader/Writer. In this mode, the PN531 can act either as an initiator or as a target.

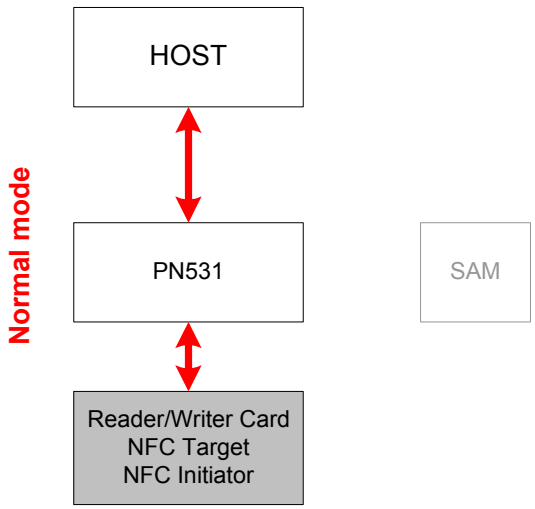


Fig 39. SAM: Normal mode

- The **Wired Card** mode is used to communicate with the SAM internally.
  No RF field is emitted. The PN531 does not react to an external Reader/Writer.

  In this mode, the PN531 acts as a Reader/Writer 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 ISO14443-4).
  - **`InDeselect`** to deselect the SAM properly.

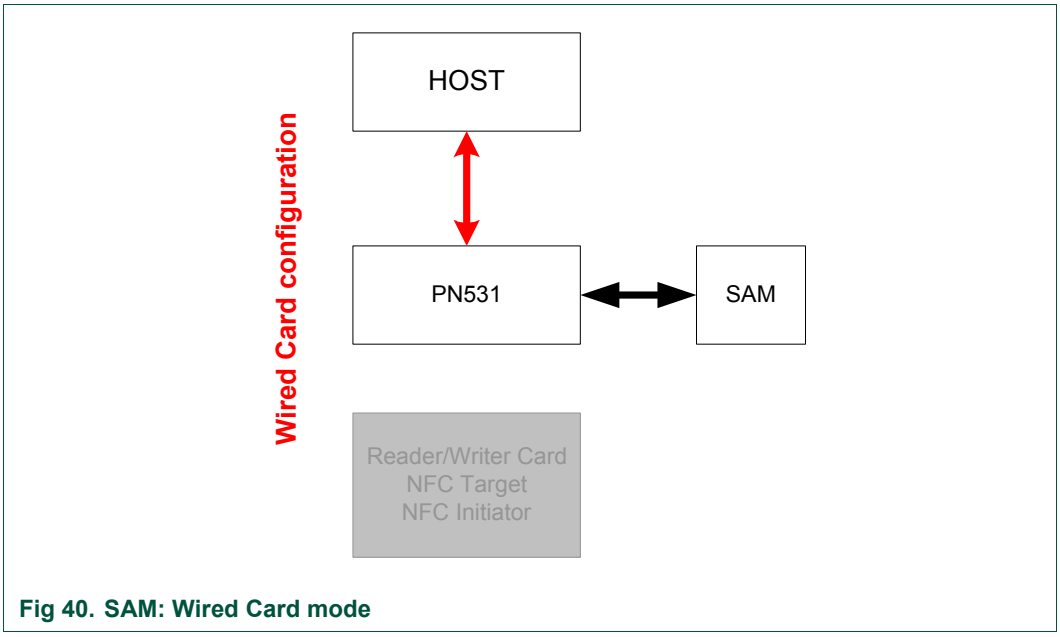


Fig 40. SAM: Wired Card mode

- In the **Virtual Card** mode, the PN531 acts as a real contactless card, and all the responses to external requests are handled by the SAM itself; neither the PN531 nor the host controller have to take care of the exchanges.
  The PN531 acts just as a bridge (analog front-end + antenna) between the Reader/Writer and the SAM.

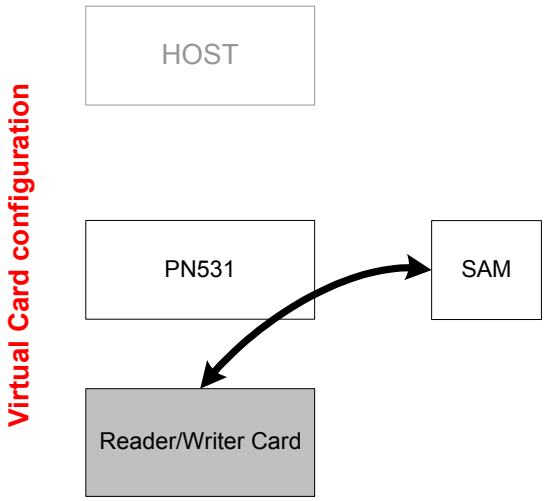


Fig 41. SAM: Virtual Card mode

The SAM CLAD line enables the PN531 to know when the contactless transaction between the external Reader/Writer and the SAM is completed.

This line is active low. The SAM asserts this line after having received a REQUEST from the external Reader/Writer. It stays active until one of the following conditions occurs:

– a valid HALT, or DESELECT command is received from the external Reader/Writer

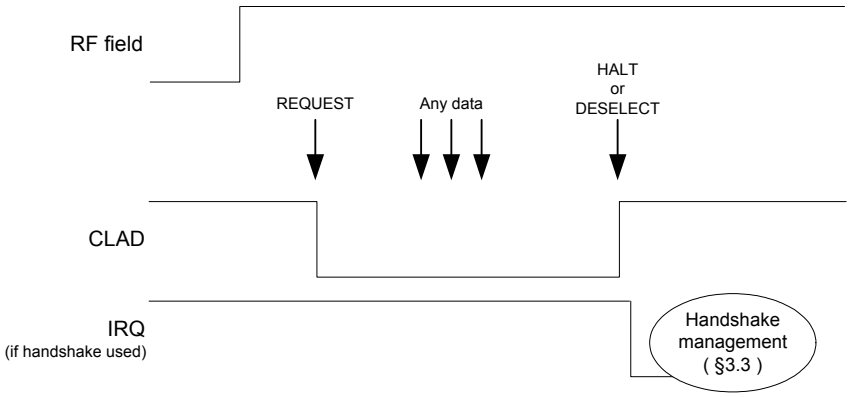


Fig 42. SAM: CLAD line (1)

– external RF field is cut:

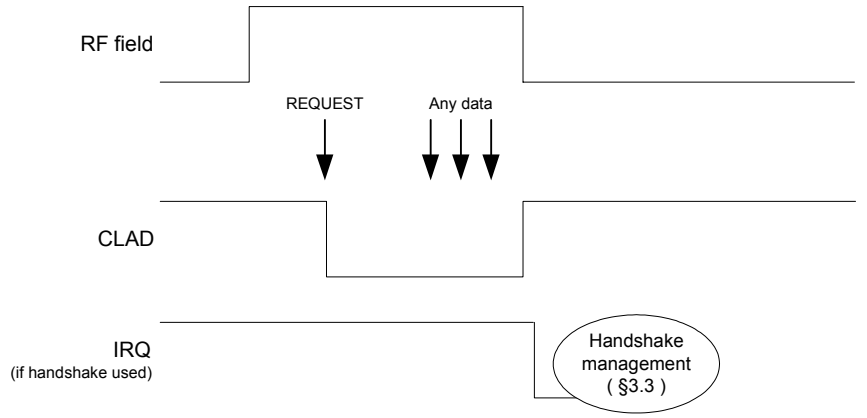


Fig 43. SAM: CLAD line (2)

The PN531 monitors this CLAD line to inform the host controller (**in handshake mode**) when a transaction has been finished (successfully or not) as described in the §3.3.2.4, p.40 and §3.3.3.4, p.45.

The conditions for the PN531 to inform the host controller are that a complete negative pulse has been detected on the CLAD line or that the **timeout** value has been reached.

The timeout starts as soon as external RF field is detected (Fig 44), and re-starts when a falling edge is detected on the CLAD line (Fig 45).

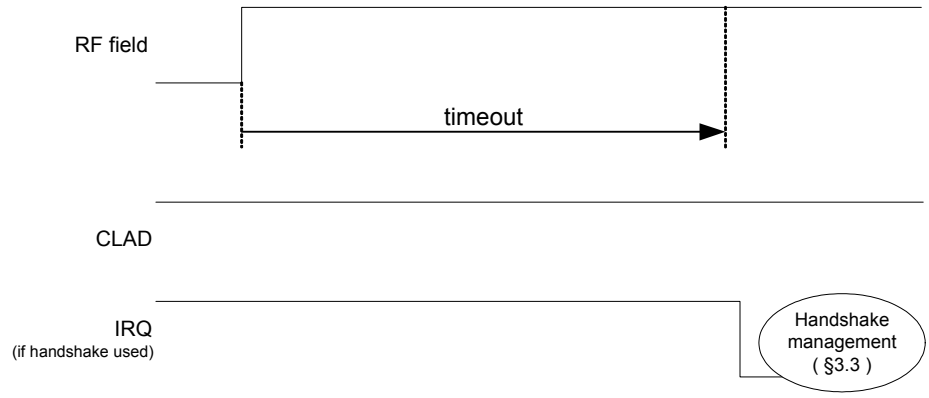


Fig 44. SAM: timeout (1)

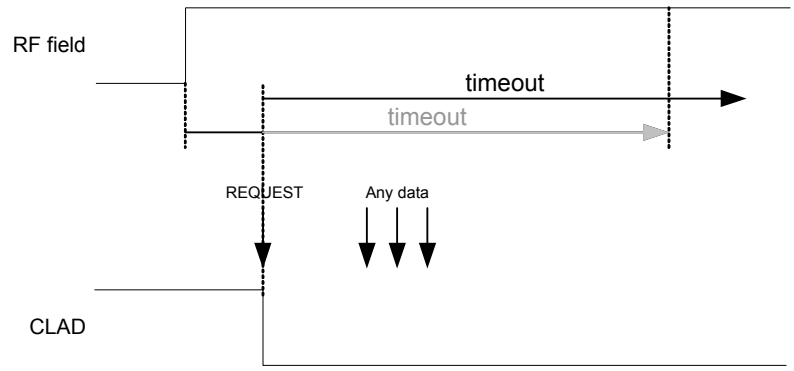


Fig 45. SAM: timeout (2)

Of course, if a complete pulse is detected on the CLAD line, the timeout is stopped and the host controller is informed as well (Fig 46).

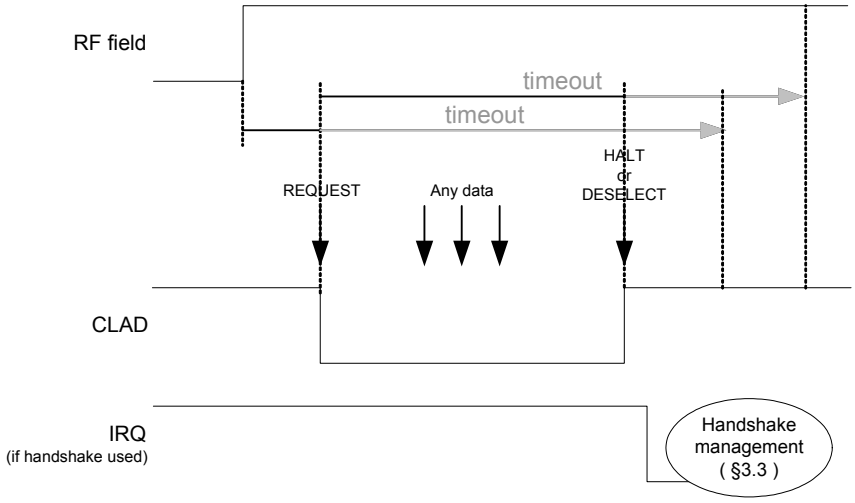


Fig 46. SAM: timeout (3)

Once informed by the IRQ line that something happened with the SAM, the host is invited to use the **`GetGeneralStatus`** command (§4.2.3, p.54) 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 at external world; the SAM and the target functionality of the PN531 itself.

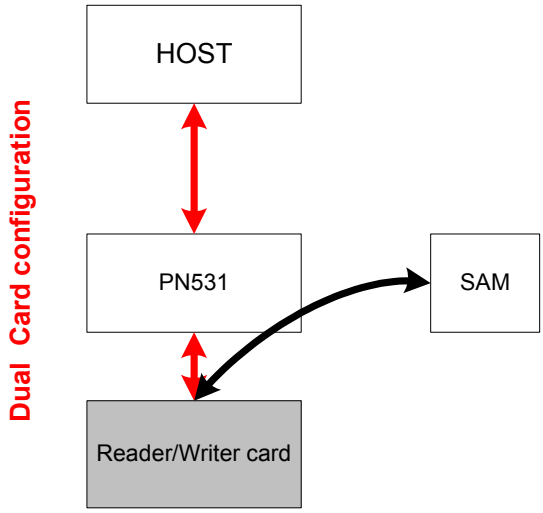


Fig 47. SAM: Dual Card mode

Example:

One possible scenario to use the SAM is the following:

- Configure the PN531 in **Virtual Card** mode to allow transaction from external Reader/Writer to the SAM.
  If the handshake mechanism is used, the PN531 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 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 return 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*

4.2.11 PowerDown

Input:

D4	16	WakeUpEnable

- **WakeUpEnable** defines the authorized sources to wake up the PN531 from power down mode.

Output:

D5	17	Status

- **Status** indicates if the command is accepted or not (see §4.1, p.48).

Syntax Error Conditions:

- **WakeUpEnable** parameter missing

Description:

This command can be used to put the PN531 (including the contactless analog front end) into power down mode in order to save power consumption.

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

rfu	rfu	SPI	HSU	RF Level Detector	USB	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” bits (bits 6 and 7) **must** be 0.

If the PN531 is currently activated as a target, the session will be lost with this command and the internal state returned by a **`TgGetTargetStatus`** command will be *RELEASED*.

When The PN531 is configured in initiator mode, the user has to manually switch the RF field off before using this command to go into power down mode.
The following **`RFConfiguration`** command will be used:

32 01 00

In the case of USB link used, the suspend mechanism is completely internally managed by the firmware.

Thus, the **`PowerDown`** command cannot be used in that case (Status informs that the command is not allowed: error 0x26).

Remark: If the handshake mode is used (with I2C or HSU link), the PN531 goes automatically into power down mode after **`TgInitTAMATarget`** command is launched. It wakes up only when an external RF field is detected or when the host controller sends a new command (§3.3, p.36).

HSU wake up condition:

When the host controller sends a command to the PN531 on the HSU link in order to exit from power down mode, the PN531 needs some delay to be fully operational (the real waking up condition is the 5th rising edge on the serial line, see ref. [**3**]).

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 ant wait the waking up delay before sending the command frame

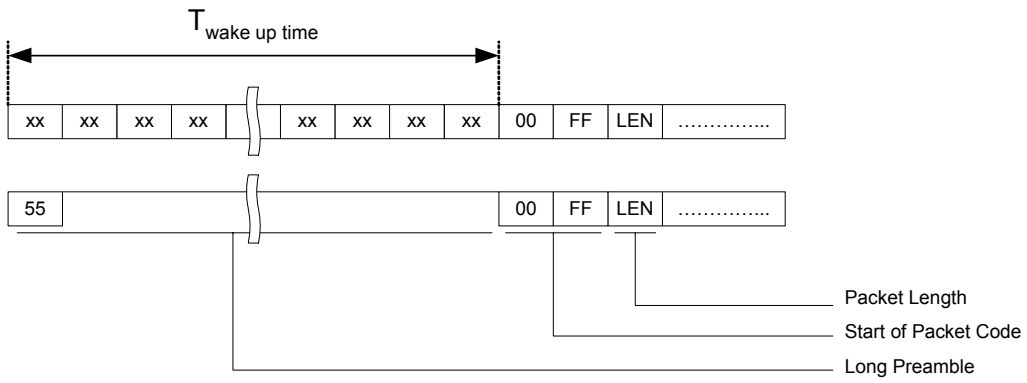


Fig 48. HSU Wake up

SPI wake up condition:

When the host controller sends a command to the PN531 on the SPI link in order to exit from power down mode, the PN531 needs some delay to be fully operational (this delay has a typical value of 300µs[5]).

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

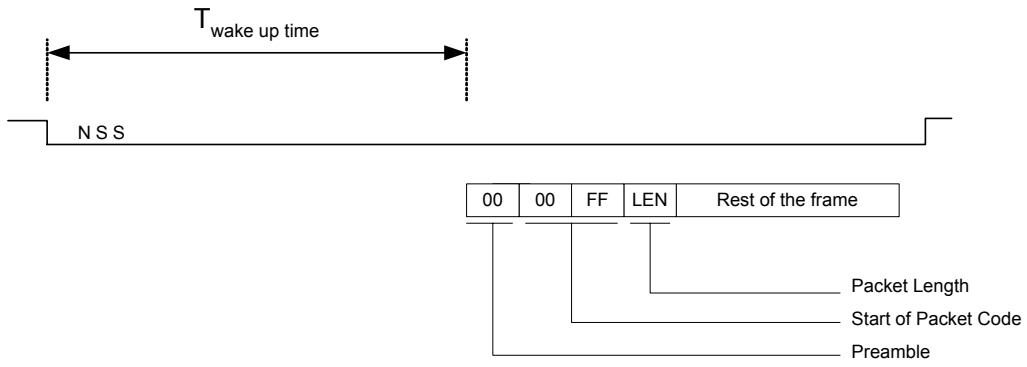


Fig 49. SPI wake up

[5] See Reference [**3**] for a more precise value (depending on power supply value, temperature, …)

4.3 RF Communication commands

4.3.1 RFConfiguration

Input:

D4	32	CfgItem	ConfigurationData []

The **ConfigurationData** field length and content depends on the **CfgItem** as follows:

- **CfgItem = 0x01: RF Configuration** (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 **off**, the bit **AutoRFCA** indicates that the PN531 does not need to take care of external field before switching on its own field.
  In other words, if **AutoRFCA** is off and **RFon/off** is on, the PN531 will generate RF field whatever external field is (present or not).

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

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 PN531 is initiator. A target is considered as mute if no valid ATR_RES is received within this timeout value. In this way, the PN531 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 PN531 uses in the **`InCommunicateThru`** (§4.3.9, p.111) 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`** (§4.3.8, p.102) when the target is a FeliCa™ or a Mifare® card (Ultralight, Standard, …).

For the timings of item 2, the coding is the following:

In case n = 0 No timeout / no delay

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

Table 18: Timings definition for RFConfiguration command

Byte Value	Timeout Value
0x00	no timeout / no delay
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 PN531 will use in the **`InCommunicateThru`** (§4.3.9, p.111) command in case of mute target or error detected. This information is also used with **`InDataExchange`** (§4.3.8, p.102) when the target is a FeliCa™ or a Mifare® card.
  The specific value 0xFF means that the PN531 retries eternally.
  The default value of this parameter is 0x00 (no retry).

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

Byte 1	**MxRtyATR**
Byte 2	**MxRtyPSL**
Byte 3	**MxRtyPassiveActivation**

The parameters **MxRtyATR**, **MxRtyPSL** and **MxRtyPassiveActivation** define the number of retries that the PN531 will use in case of the following processes:

- **MxRtyATR** is a byte containing the number of times that the PN531 will retry to send the ATR_REQ in case of incorrect reception of the ATR_RES (or no reception at all - timeout).
  Value 0xFF means to try eternally, 0x00 means only once.
  The default value of this parameter is 0xFF (infinitely).
- **MxRtyPSL** is a byte containing the number of times that:
  - the PN531 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 PN531 will retry to send the PPS request in case of incorrect reception of the PPS response (or no reception at all) for the ISO14443-4 protocol.

  Value 0xFF means to try eternally, 0x00 means only once.
  The default value of this parameter is 0x00 (the PSL_REQ/PPS request is sent once).
- **MxRtyPassiveActivation** is a byte containing the number of times that the PN531 will retry to activate a target in **`InListPassiveTarget`** command (§4.3.5, p.94).
  Value 0xFF means to try eternally, 0x00 means only once.
  The default value of this parameter is 0xFF (infinitely).

- **CfgItem = 0x06: Analog settings when configured as a target** (11 bytes)

  This CfgItem is used to choose the analog settings that the PN531 will use when configured as a target whatever the baud rate or the type of modulation are.

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

Byte 1	**RFCfgReg**
Byte 2	**GsNReg**
Byte 3	**CWGsPReg**
Byte 4	**ModGsPReg**
Byte 5	**ModWidthReg**
Byte 6	**TxBitPhaseReg**
Byte 7	**RxSelReg**
Byte 8	**DemodReg** for passive communication mode
Byte 9	**RxThresholdReg**
Byte 10	**DemodReg** for active communication mode
Byte 11	**GsNLoadModReg**

- **CfgItem = 0x07: Analog settings when configured as an initiator** (11 bytes)

  This CfgItem is used to choose the analog settings that the PN531 will use when configured as an initiator whatever the baud rate or the type of modulation are.

  The eleven internal registers that are set with this command are the same than for the CfgItem 6

  For all the registers, which can be modified with CfgItem 6 or 7, the default values are the following:

Table 19: Default values for registers in RFConfiguration command

Register	Default value for Target Mode (CfgItem 6)	Default value for Initiator Mode (CfgItem 7)
RFCfgReg	0x59	0x59
GsNReg	0xF4	0xF4
CWGsPReg	0x3F	0x3F
ModGsPReg	0x30	0x30
ModWidthReg	0x26	0x26
TxBitPhaseReg	0x87	0x87
RxSelReg	0x84	0x84
DemodReg (passive)	0x61	0x4D
RxThresholdReg	0x85	0x85
DemodReg (active)	0x61	0x61
GsNLoadModReg	0x6F	0x6F

Output:

D5	33

Syntax Error Conditions:

- Various timings values greater than 16 (item 2).
- Incorrect command length.

Description:

This command is used to configure the different settings of the PN531 as described in the input section of this command.

4.3.2 RFRegulationTest

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					00 : Mifare® 10 : FeliCa™	

Output:

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

Description:

This command is used for radio regulation test.

The PN531 makes RF transmission with pseudo random numbers by using the PRBS15 bit of the *TEST_SEL2* register (see reference [**3**]). 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 or 424 kbps).
The **TxMode.TxFraming** parameter defines the type of modulation (Mifare® or FeliCa™ modulation).

The PN531 transmits data until a new command comes from the host controller.

4.3.3 InJumpForDEP

Input:

D4	56	ActPass	BR	Next	[PassiveInitiatorData] (4 or 5 bytes)
					[NFCID3i] (10 bytes)
					[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 initialisation 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 wants to initialize a target with a known ID (according to reference [**1**], 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 ref. [**1**], §11.2.2.5.
- **NFCID3i** is the NFCID3 of the initiator that is used by the PN531 within the ATR_REQ.
  If the NFCID3i is not delivered by the host controller, the PN531 will use a random value to build the ATR_REQ.
- **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 §4.1, p.48).
- **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)

Description:

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.

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 the **`SetTAMAParameters`** (§4.2.9, p.67)), the PN531 constructs the ATR_REQ with or without a DID parameter. If used, the DID value is fixed by the PN531 to 0x01.
    - receive ATR_RES.
      The PN531 waits for this answer from the target a maximum time (timeout defined with the **`RFConfiguration`** command (§4.3.1, p.80)).
      In case of incorrect ATR_RES received, the PN531 sends again ATR_REQ (MxRtyATR times)
  - if a correct ATR_RES is received then the PN531 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.

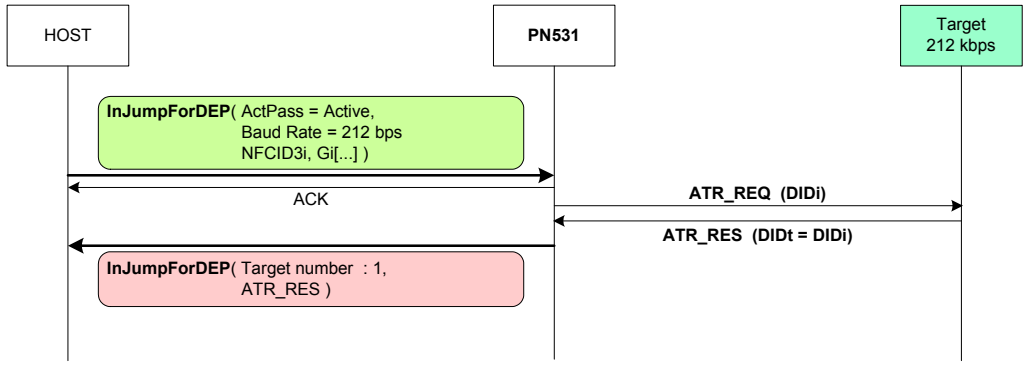


Fig 50. InJumpForDEP – Active communication mode – DID used

- PASSIVE MODE
  - do Initial RFCA
  - if BR = 106 kbps
    - select one target (SENS_REQ, SDD_REQ, SEL_REQ).
    - if the target is Transport Protocol Equipped (SEL_RES), then store the NFCID1t for further use.
    - if no target or no TPE target is detected, try again the process (SENS_REQ, …). The absence of target is detected with a ~5 ms timeout value.
    - 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 PN531 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`** : §0, p.93).
    - if a correct POL_RES answer is received, then store the NFCID2t for further use.
    - if no valid POL_REQ is received in due time, try again the process (POL_REQ).
    - send an 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 PN531 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 PN531 and the ATR_RES is sent back to the host controller (CMD0 CMD1 = 0xD5 0x01 excepted).
    The PN531 waits for this ATR_RES coming from the target a maximum time (timeout defined with the **`SetTAMAParameters`** (§4.2.9, p.67) command).
    In case of incorrect ATR_RES received, the PN531 sends again ATR_REQ (MxRtyATR times).
  - If correct ATR_RES is received then the PN531 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.

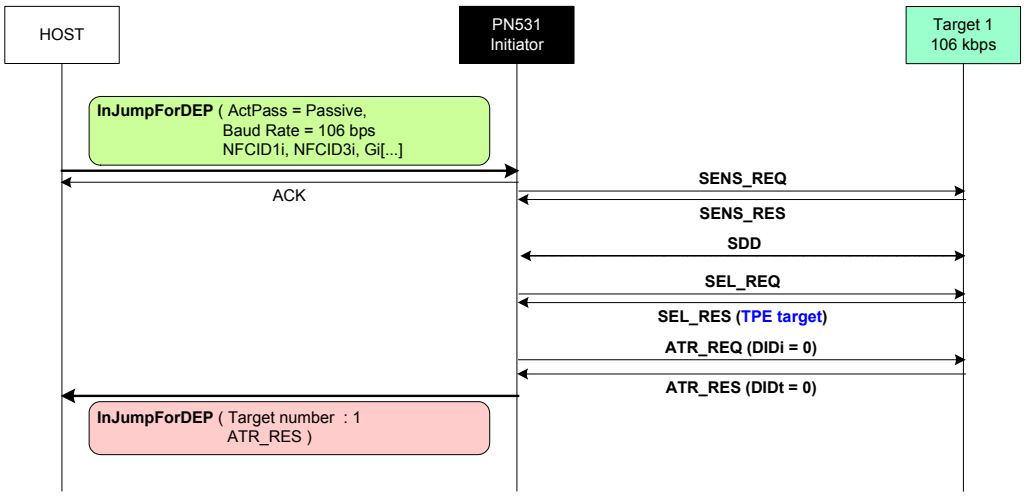


Fig 51. InJumpForDEP – Passive Communication Mode – DID not used

Rq: 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.

4.3.4 InJumpForPSL

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 initialisation 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 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 ref. [**1**], §11.2.2.5.
- **NFCID3i** is the NFCID3 of the initiator that is used by the PN531 within the ATR_REQ.
  If the NFCID3i is not delivered by the host controller, the PN531 will use a random value to build the ATR_REQ.
- **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 §4.1, p.48)
- **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)

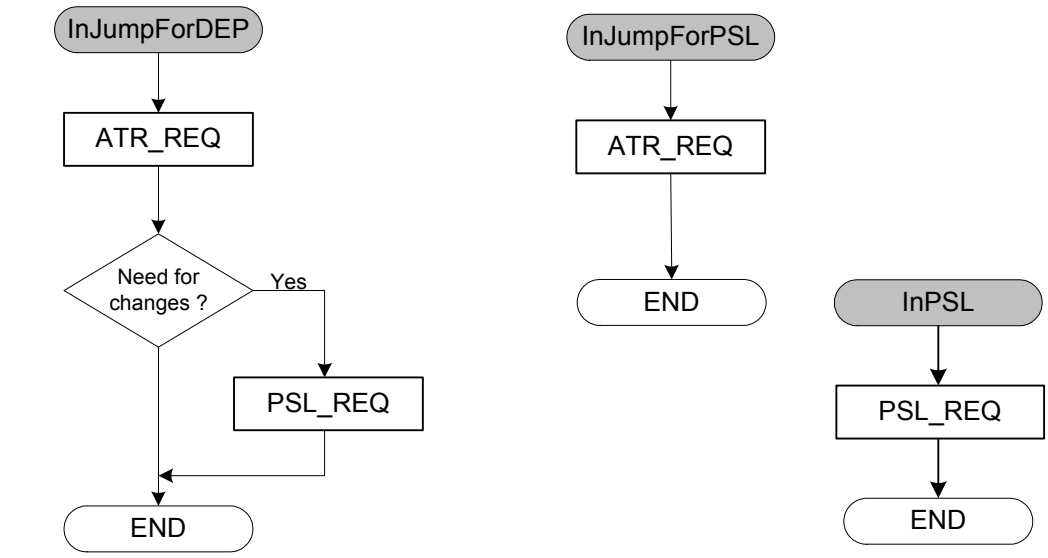
Description:

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.

Consequently, the process of this command is exactly the same than the one of **`InJumpForDEP`** (§4.3.3, p. 86).

The following figure shows the differences between these two commands.

In the **`InJumpForPSL`** representation, one can see that a PSL can be done by using additional **`InPSL`** command (§4.3.7, p.100).



a. InJumpForDEP

b. InJumpForPSL

Fig 52. Comparison between InJumpForDEP and InJumpForPSL

Rq: 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.

4.3.5 InListPassiveTarget

Input:

D4	4A	MaxTg	BR	[InitiatorData[]]

- **MaxTg** is the maximum number of targets to be initialized by the PN531. The PN531 is capable of handling 2 targets maximum at once, so this field should not exceed 0x02.
- **BR** is the Baud Rate to be used during the initialisation
  - 0x00 : 106 kbps
  - 0x01 : 212 kbps
  - 0x02 : 424 kbps
- **InitiatorData[]** is an array of data to be used during the initialisation of the target(s). 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 wants to initialize a target with a known ID. In that case, InitiatorData[] contains the ID of the card (or part of it).
  - **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 ref. [**1**], §11.2.2.5.

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:**

Tg	SENS_RES (2 bytes)	SEL_RES (1 byte)	NFCIDLength (1 byte)	NFCID1[] (NFCIDLength bytes)	[ATS[]] (ATSLength bytes[6])

  - **212/424 kbps:**

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

[6] 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. Reference [**2**]). Thus here, the ATS structure is [ATSLength xx_1 xx_2 xx_3 xx_4 … $xx_{ATSLength-2}$ $xx_{ATSLength-1}$]

Syntax Error Conditions:

- **MaxTg** value is incorrect (0 or higher than 2)
- **BR** value is incorrect
- TSN number in the **InitiatorData** is incorrect in passive 212/424 kbps mode (different from 0x00, 0x01, 0x03, 0x07 or 0x0F)

Description:

The goal of this command is to detect as many targets (maximum **MaxTg**) as possible in passive mode. Depending on the baud rate requested, the PN531 will use a "Mifare®" initialisation or a "FeliCa™" one.

The default analog settings or those that have been modified by the host controller with the **`RFConfiguration`** command (CfgItm7) are used during the activation.

- if **BR** = 106 kbps
  - As long as there is no target detected (maximum = **MaxTg**),
    (*This process is done MxRtyPassiveActivation times (§4.3.1, p.80, 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 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 ID of a card.
    - if one of several target(s) have been detected with the previous command, handle the anti-collision (SDD_REQ) and then select one target (SEL_REQ command).
    - if the selection is successful, the PN531 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 PN531) is lost.
    - 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)
      Remark: 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 the target indicates that it is ISO14443-4 compliant, then the PN531 carries out the ISO14443-4 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 a ISO14443-4 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 ($0 \leq$ **NbTg** $\leq$ 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.

- else if **BR** = 212 or 424 kbps
  - As long as there is no target detected, 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 (§4.3.1, p.80, 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 trials allowed).*
  - When one or several targets have answered to the polling request command, the PN531 checks the coherence of the answer(s):
    - the command byte in the polling response has to be equal to 1
    - the PN531 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 PN531 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 PN531) is lost.

    Fill the **TargetData$_i$[]** output buffer with the answers of the valid targets:
    - 1 byte containing the logical number attributed (Tg)
    - response code byte, fixed value 0x01
    - 1 byte indicating the length of the POL_RES (IDm + Pad + [SystCode])
    - 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).

Rq: If this command is aborted by the host controller without any target activated, the RF field is automatically switched off to decrease power consumption.

Examples:

- In this first example, the host requires the initialisation of one target at 106 kbps.

```
➔ 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 this second example, the host requires the initialisation 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`

4.3.6 InATR

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 (optional)
  If the NFCID3i is not delivered by the host controller, the PN531 will use a random value to build the ATR_REQ
- **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 §4.1, p.48)

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.

Description:

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

It is assumed that the target **Tg** has been initialized before (see command **`InListPassiveTarget`** §0, p.93).
If the **Tg** number is unknown, the PN531 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:

- do Initial RFCA

- 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**)
  - send ATR_REQ constructed by means of **NFCID3i** and **Gi** at **BR**
    Depending on the value of the internal parameter fDIDUsed (set by the host controller with the **`SetTAMAParameters`** (§4.2.9, p.67)), the PN531 constructs the ATR_REQ with or without a DID parameter. If used, the DID value is fixed by the PN531 to 0x01.
    If NFCID3i is not present in the command from the host controller, the PN531 uses a random value. If **Gi** is not present in the command, the PN531 constructs the ATR_REQ without **Gi** bytes.
  - receive ATR_RES
    The complete ATR_RES received from the target is returned back to the system controller in the **ATR_RES** field for further decision (CMD0 CMD1 = 0xD5 0x01 excepted).
    The NFCID3t is stored internally in the PN531, as a 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 command **`SetTAMAParameters`** (§4.2.9, p.67).
    If no valid ATR_RES is received (MaxRtyATR attempts), the PN531 returns in the **Status** byte an error code.

    The PN531 waits for this ATR_RES coming from the target a maximum time (timeout defined with the **`SetTAMAParameters`** (§4.2.9, p.67) command).
    In case of incorrect ATR_RES received, the PN531 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)
```

4.3.7 InPSL

Input:

D4	4E	Tg	BRit	BRti	[ModWidth]

- **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
- **ModWidth** is an optional parameter that will be applied –in case of ISO14443-4 card- in the *CL_MODWIDTH* internal contactless register (See Reference [**3**]).

Output:

D5	4F	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48).

Syntax Error Conditions:

- **BRit** value is incorrect
- **BRti** value is incorrect
- Incorrect command length

Description:

This command is used by a host controller to change the defined bit rates either with a TPE target or with a ISO14443-4 target.

It is assumed that the target **Tg** has been activated before (see commands **`InListPassiveTarget`** (§0, p.93), **`InATR`** (§4.3.6, p.98) and **`InJumpForPSL`** (§4.3.4, p.91)). If this is not the case, or if the **Tg** number is unknown, the PN531 informs the host controller with the 0x27 error code (**Status**).

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)
  - the PN531 takes internally into account the parameters changes
  - The **Status** byte is set to 0x00 and sent back to the host controller
- else the PN531 gives up and answers to the system controller with a **status** byte <> 0x00.

In the case of a **ISO14443-4 card**, a Protocol and Parameter Selection 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)
  - the PN531 takes internally into account the parameters changes
  - The **Status** byte is set to 0x00 and sent back to the host controller
  - If **ModWidth** is present in the command, this parameter is written in the *CL_MODWIDTH* register. This allows improving communication performance depending on the new bit rate chosen by the host controller.
- else the PN531 gives up and answers to the system controller with a **status** byte <> 0x00.

Possible errors returned:

- Negotiation already performed with the relevant target ➔ error 0x26
- The target is neither a TPE nor ISO14443-4 ➔ error 0x26
- Unknown target number ➔ error 0x27

4.3.8 InDataExchange

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 wants to send more data that all the data contained in the DataOut[] array (see Chaining mechanism §4.4.5, p.143). This bit is only valid for a TPE target, not pure Mifare®, FeliCa™ or ISO14443-4 types.
- **DataOut** is an array of raw data to be sent to the target by the PN531.
  The total length of this array is determined by the host ⇔ PN531 protocol layer (see §4.4.6, p.149).

Output:

D5	41	Status	[DataIn[]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48).
  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 §4.4.5, p.143).
- **DataIn** is an array of raw data received by the PN531 (coming from the target)

Syntax Error Conditions:

- In case of ISO14443-3 target:
  - **Cmd** value is incorrect
  - Bad number of data for Authentication command (Data0..9)
  - Bad number of data for 16 bytes writing command (Data0..15)
  - Bad number of data for 4 bytes writing command (Data0..3)

Description:

This command is used to support protocol data exchanges between the PN531 as an initiator and a target whatever type it is.

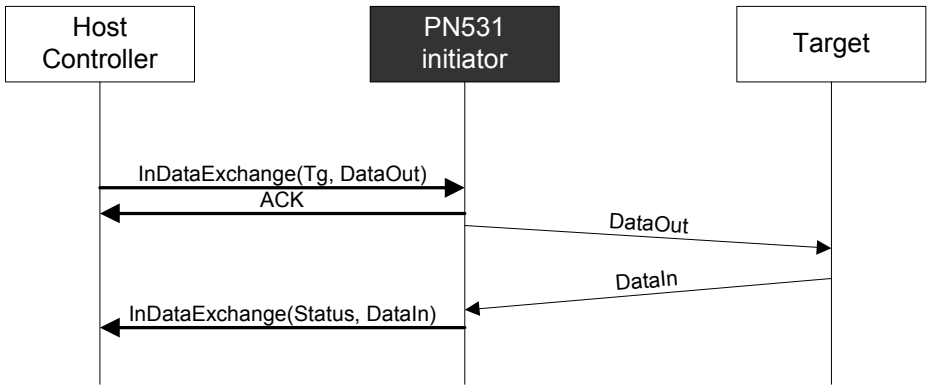


Fig 53. InDataExchange – General context

This command can be used for exchanging data with a target that has been previously enumerated by the PN531, by using one of the 3 possible commands: **`InListPassiveTarget`**, **`InJumpForDEP`** or **`InJumpForPSL`**.

If the target number is unknown, the PN531 returns an error in the **status** byte (Code 0x27).

The PN531 stores internally all the necessary information needed about all the targets:

- type (DEP, Mifare® card, ISO14443-4 card, FeliCa™ card),
- baud rate, communication mode (active or passive),
- internal state (selected, deselected, released, …)

So, first of all, the PN531 applies all the correct configuration settings corresponding to the target **Tg** (Baud Rate, modulation type, …).

Then, if the target n° **Tg** is not currently selected (current state memorized inside the PN531), the PN531 initiates the selection.
The detailed process of the selection depends on the type of the target; it is given in the **`InSelect`** (§4.3.12, p.116) command description.

After this selection stage, the PN531 takes in charge the data exchange.

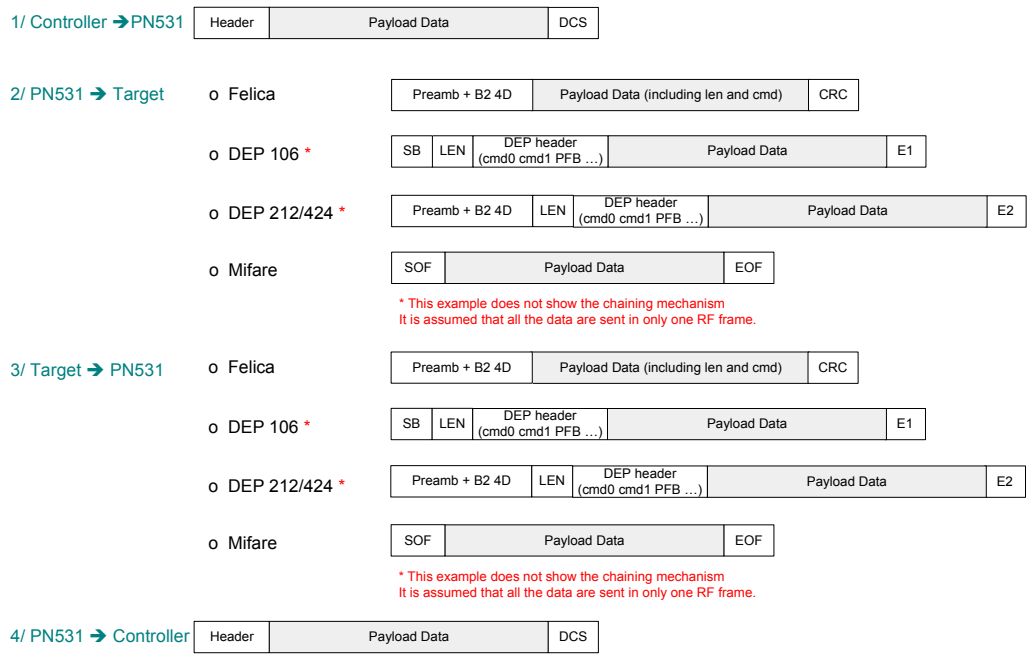


Fig 54. InDataExchange – Different target types

The way of exchanging data is different depending on the real nature of the target (DEP, Mifare® card, FeliCa™ card, ISO14443-4 card):

- **Mifare® card**

When the target **Tg** is Mifare® compliant, the input parameters are interpreted by the PN531 to execute a Mifare® exchange. The PN531 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`** §4.3.1, p.80.

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

Cmd	Addr	[Data0..15]

- **Cmd** is the Mifare® specific command byte
- **Addr** is the address associated with the Mifare® command
- **Data0..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:
    - Data0..5 contain the 6 bytes key,
    - Data6..9 contain the 4 bytes serial number of the card.

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

[Data0..15]

- **Data0..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® card documentation 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 PN531 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 ID 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..0x10 from the address 0x02.

- **ISO14443-4 card**

When the target **Tg** is ISO14443-4 compliant, the input parameters are interpreted by the PN531 to execute a ISO14443-4 exchange.

The PN531 uses the data contained in the **DataOut[]** buffer to build the frames.

The main ISO14443-4 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[]**.

If parameter P3 (from C-APDU in **DataOut[]**) is greater than 0xFA (250 bytes), the C-APDU command is not sent and the PN531 returns a status byte with error 0x07.

Example:

It is assumed in this example that the logical number attributed by the PN531 to the card is #01. 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.

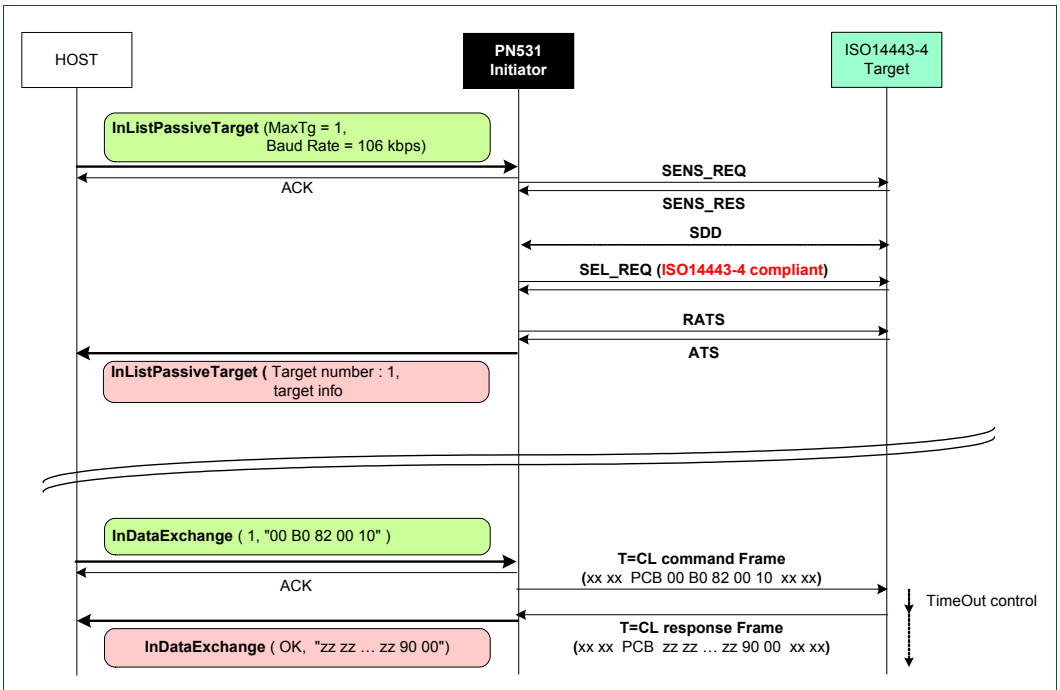


Fig 55. InDataExchange – Example of a ISO14443-4 exchange

During the exchanges, the following timeout control is used between the ISO14443-4 command and the response from the card:

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

Concerning the error handling, the PN531 tolerates up to 3 errors detected in the communication flow before returning an error code to the host controller.

- **FeliCa™ card**

When the target **Tg** is a FeliCa™ card, the PN531 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 PN531 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. The configuration of this timeout is done with the **`RfConfiguration`** command (§4.3.1, p.80), item 2 (fRetryTimeout) and 4 (MaxRtyCOM).

Examples:

It is assumed in this example that the logical number attributed by the PN531 to the card is #01. 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.

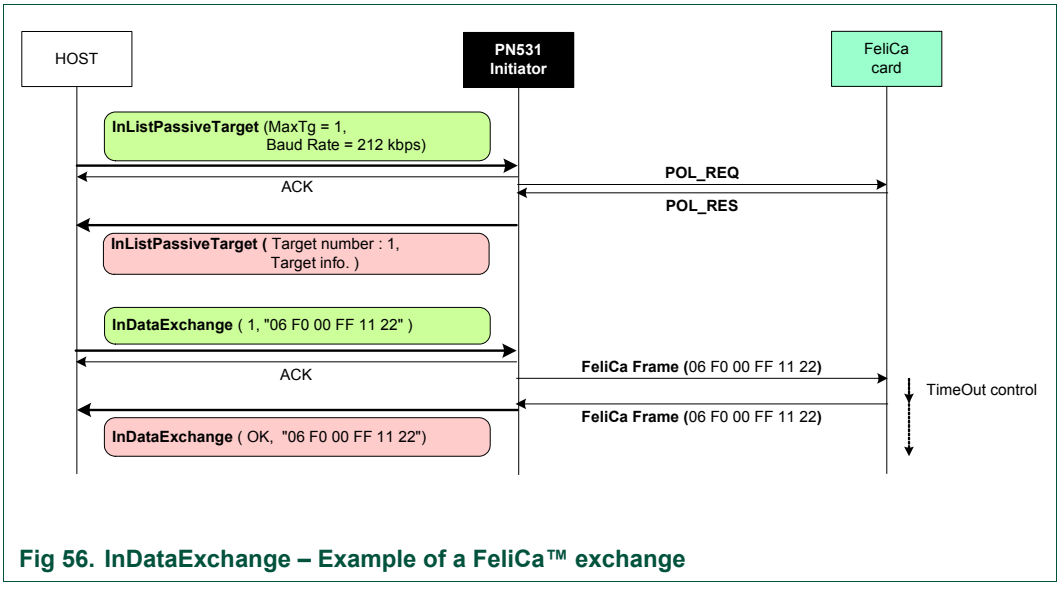


Fig 56. InDataExchange – Example of a FeliCa™ exchange

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

Here, the status byte informs of a timeout detected by the PN531.

- **DEP target**

When the target **Tg** is a NFC-DEP, the PN531 takes care of the protocol internally.

The PN531 sends the data contained in the **DataOut[]** array either in one or several stages (chaining mechanism as described in reference [**1**] and §4.4.5, p.143) depending on the total length of the frame to send.

The PN531 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 PN531.

If the host command does not contain *More Information* bit in the **Tg** field, the PN531 switches to reception. After having received a complete answer from the target, the PN531 sends back the data received in the **DataIn[]** array.

If the *More Information* bit is set, the PN531 returns no data to its own host controller but takes care of the target with the timeout extensions request and response.

Remark: Both **DataIn[]** and **DataOut[]** can contain NAD information, when used in the DEP. See **`SetTAMAParameters`**, §4.2.9, p.67 to have a complete description.

Example:

It is assumed in this example that the logical number attributed by the PN531 to the card is #01. The card 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.

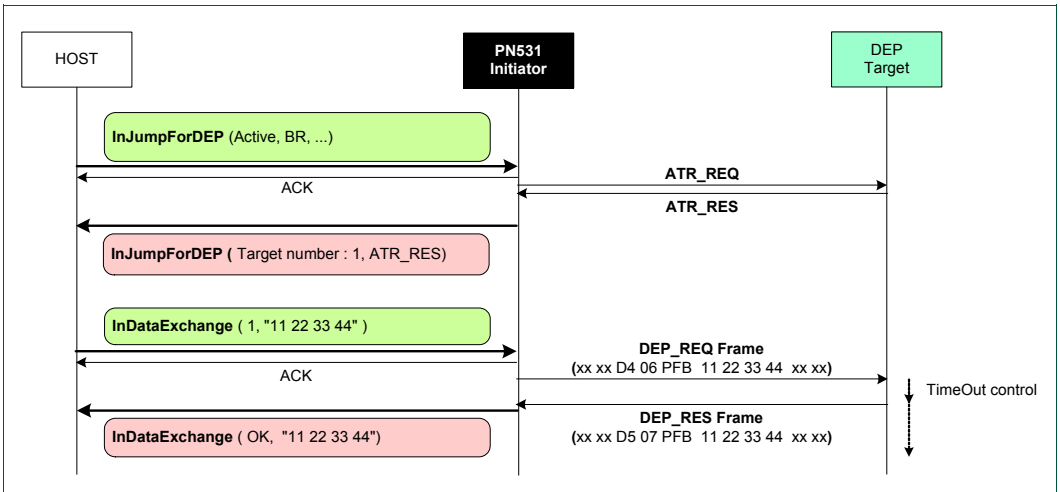


Fig 57. InDataExchange – Example of a DEP exchange

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

This second example shows the use of MI bit in the command.

⇒ **D4 40 41 11 22 33 44** **F2 F3**

⇐ **D5 41 00**

The value of the status byte is 0x00, indicating that the exchange is correct. No data are returned, the host can send the remaining data:

⇒ **D4 40 01 11 22 33 44 55 66 77**

⇐ **D5 41 00 55 AA 55 AA 55 AA**

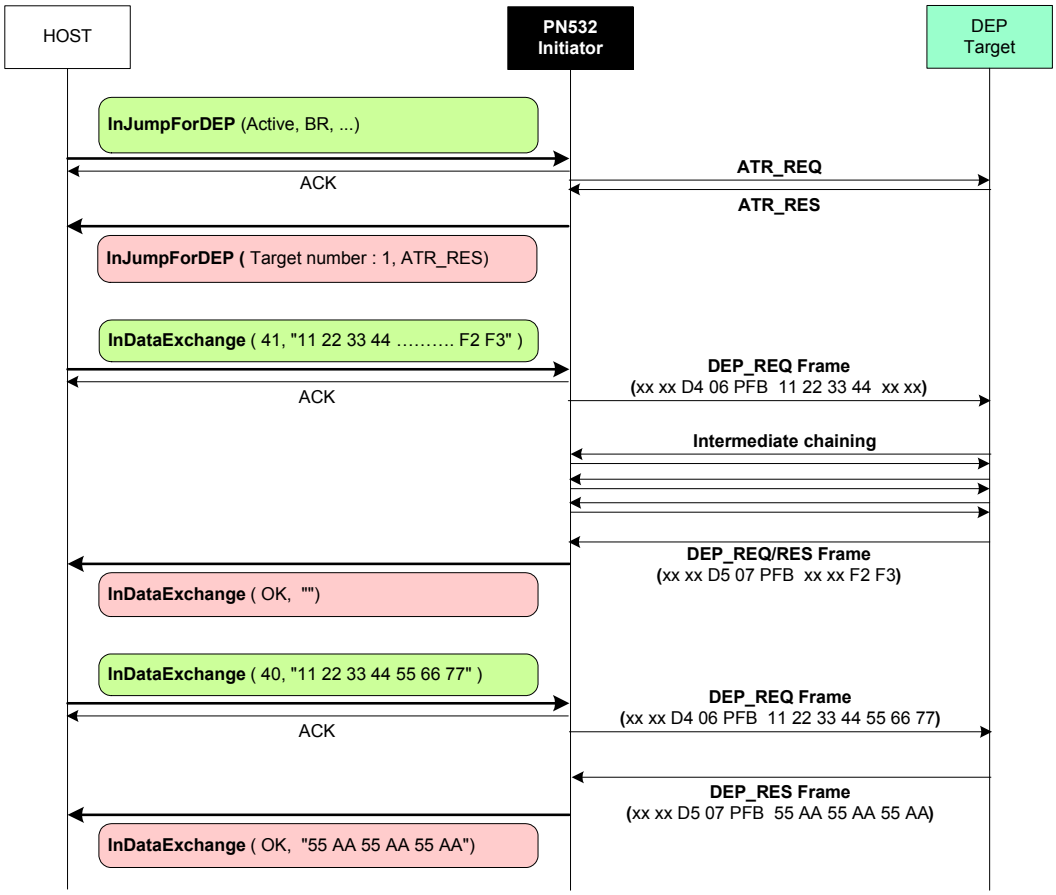


Figure 1 : InDataExchange – Example of a DEP exchange with MetaChaining

4.3.9 InCommunicateThru

Input:

D4	42	[DataOut[]]

- **DataOut** is an array of raw data to be sent to the target by the PN531. The total length of this array is determined by the host ⇔ PN531 protocol layer (max. 252 bytes, cf. §4.4.6, p.149).

Output:

D5	43	Status	[DataIn[]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48)
- **DataIn** is an array of raw data received by the PN531 (coming from the target).

Description:

This command is used to support basic data exchanges between the PN531 and a target.

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 (§0, p.93) are used for transmitting the **DataOut[]** and for receiving the **DataIn[]** bytes.

This command is complementary of the **`InDataExchange`** command (§4.3.8, p.102). The main difference compared to **`InDataExchange`** is that here the PN531 does not handle at 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 done 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 PN531:

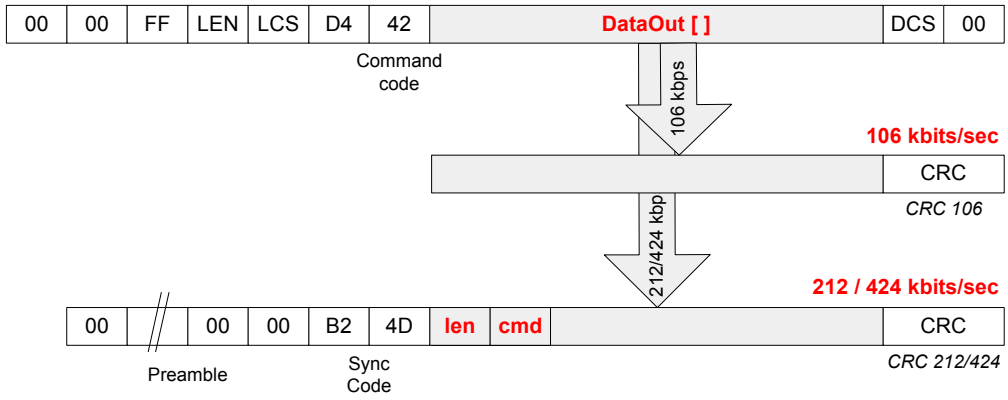


Fig 58. InCommunicateThru (1)

- 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[]** :

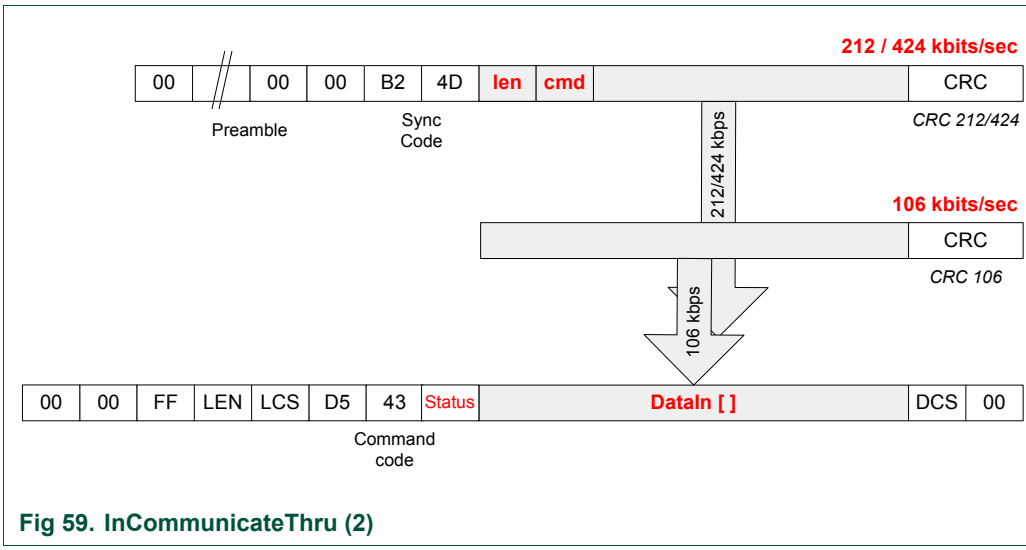


Fig 59. InCommunicateThru (2)

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

In that case, the PN531 acts only as a RF-transceiver between the host controller and the selected target.

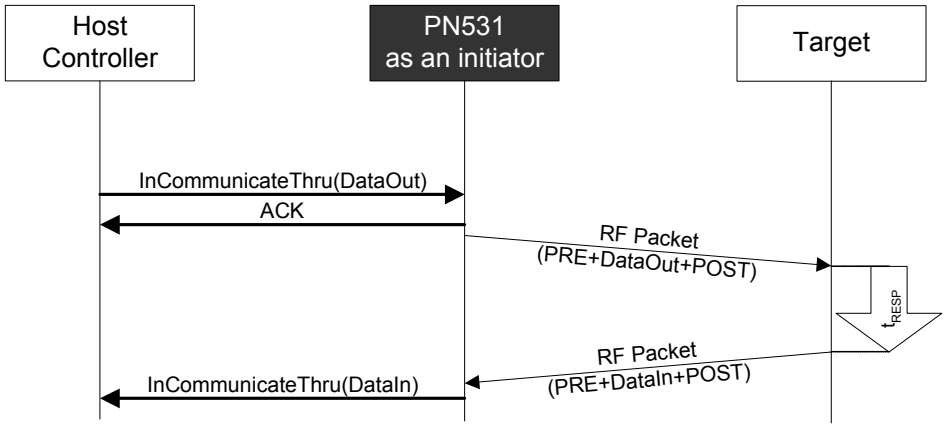


Fig 60. InCommunicateThru (3)

If the parameter **fRetryTimeout** of the command **`RFConfiguration`** (§4.3.1, p.80) is not set, no control is done on the delay (t_{RESP}) used by the target to send back its answer. The host has to manage timeout by itself.

Otherwise (**fRetryTimeout** activated), the PN531 checks the response delay (t_{RESP}) to detect mute target (delay greater than parameter **fRetryTimeout**).

In case of error (either mute target or communication error), the PN531 sends again the RF packet to the target as many times as defined in the **MaxRtyCOM** parameter (cf. **`RFConfiguration`** command in §4.3.1, p.80).

4.3.10 InDeselect

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 §4.1, p.48)

Description:

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

If this target is unknown (**Tg** number not attributed by the PN531) an error code is returned in the **Status** byte (code 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 initialised.
In case of **Tg** equals to 0x00, this process is done for the current selected target.

Table 20: InDeselect actions

Target Type	Action
DEP compliant (whatever the baud rate and the communication mode are)	Send **DSL_REQ**
Non-DEP target 106 kbps	Send **SLP_REQ**
Non-DEP target ISO14443-4 compliant	Send **S(deselect)REQ**
Non-DEP target 212/424 kbps	No action

4.3.11 InRelease

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 §4.1, p.48)

Description:

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

Releasing a target means that the host controller wants to forget it, so the PN531 can erase all the information relative to this target.

This command is used whatever the targets type (DEP or not) and its 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 the current selected target.

Table 21: InRelease actions

Target Type	Action
DEP compliant (whatever the baud rate and the communication mode are)	Send **RLS_REQ**
Non-DEP target 106 kbps	Send **SLP_REQ**
Non-DEP target ISO14443-4 compliant	Send **S(deselect)REQ**
Non-DEP target 212/424 kbps	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).

4.3.12 InSelect

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 §4.1, p.48)

Description:

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

If this target is unknown (**Tg** number not attributed by the PN531) an error code is returned in the **Status** byte (code 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 initialised.

Table 22: InSelect actions

Target Type	Action
DEP compliant target Active communication mode (whatever the baud rate is)	• Wake up of the deselected target (**WUP_REQ**).
DEP target Passive communication mode 106 kbps	• Initialisation and Single Device Detection (**ALL_REQ**, **SEL_REQ**) using the *NFCID1t* of the target that has been stored during the first 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 first activation of this target.
DEP target Passive communication mode 212/424 kbps	• Initialisation 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 first activation of this target (using padding as described in **`InJumpForDEP`**, §4.3.3, p.86).
Non-DEP target 106 kbps	• Initialisation and Single Device Detection (**ALL_REQ**, **SEL_REQ**) using the *NFCID1t* of the target that has been stored during the first activation of this target.
Non-DEP target ISO14443-4 compliant	• Initialisation and Single Device Detection (**ALL_REQ**, **SEL_REQ**) using the *NFCID1t* of the target that has been stored during the first activation of this target. • Send a **RATS**.
Non-DEP target 212/424 kbps	• Initialisation and Single Device Detection (**POL_REQ**).

Note : This command can be used in combination with `InDeselect` command (§4.3.10, p.114), as described in the following figure.

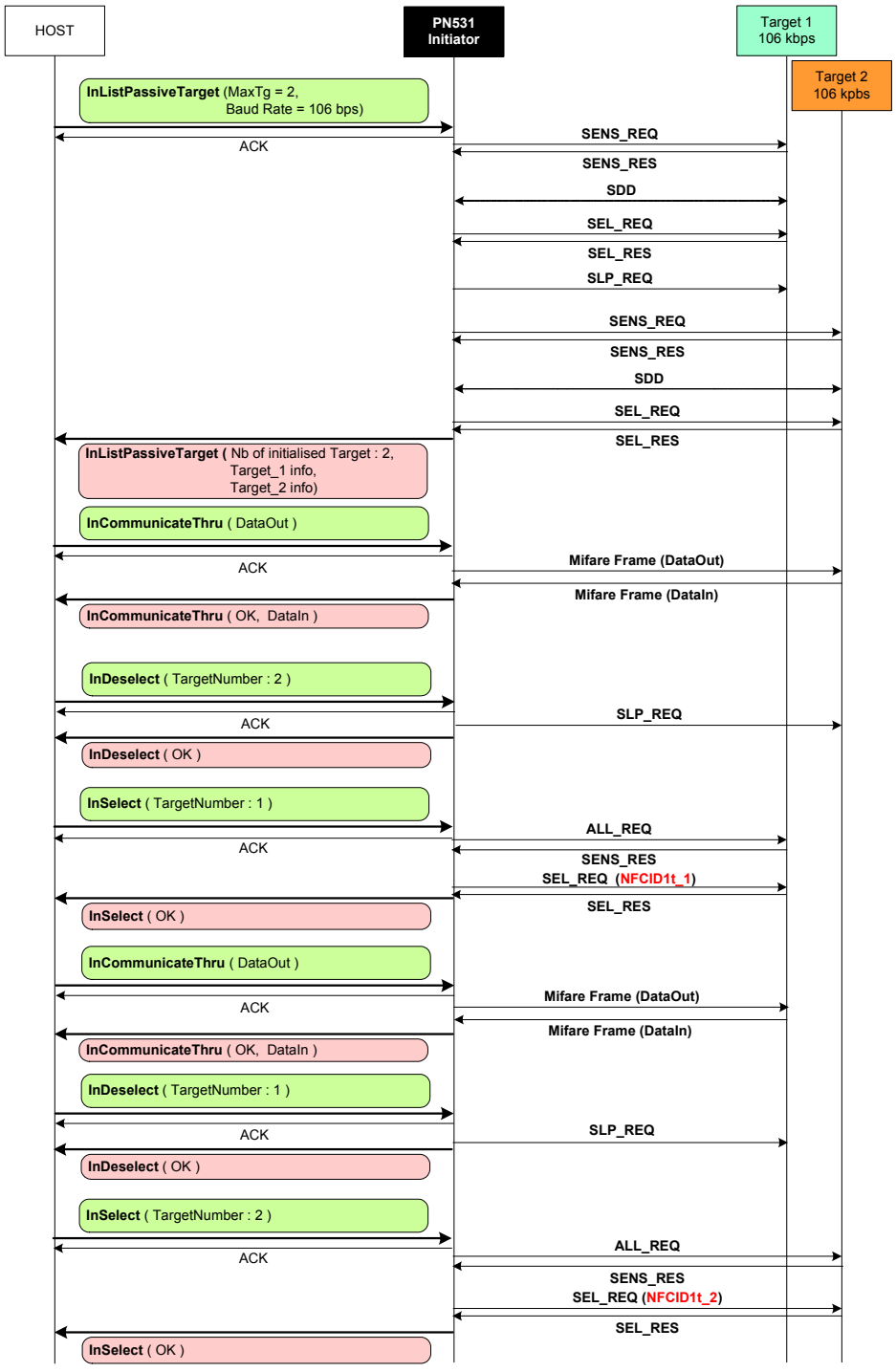


Fig 61. InSelect

4.3.13 TgInitTAMATarget

Input:

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

- **Mode** is a byte indicating which mode the PN531 should respect

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

  - *PassiveOnly* flag is used to configure the PN531 to accept to be initialized only in passive mode, i.e. to refuse active communication mode.
  - *DEPOnly* flag is used to configure the PN531 to accept to be initialized only as a DEP target, i.e. receiving an ATR_REQ frame. The PN531 can be activated either in passive or active mode, but if the PN531 receives a proprietary command frame as a first command following autocoll process, it will be rejected and the PN531 returns automatically in the autocoll state.
- **MifareParams** is the information needed to be able to be activated at 106 kbps in passive mode. MifareParams[] table is composed of:
  - *SENS_RES (2 bytes).*
  - *NFCID1t has a fixed length of 3 bytes containing the nfcid11 to nfcid13 bytes. Indeed, the PN531 can handle only NFCID1t in single size.*
  - *SEL_RES (1 byte), typical value = 0x40.*
- **FeliCaParams[]** contain the information to be able to respond to a polling request at 212/424 kbps in passive mode. FeliCaParams[] table is composed of:
  - *NFCID2t (8 bytes)*
  - *PAD (8 bytes)*
  - *SystemCode (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.
- **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).

Output:

D5	8D	Mode	InitiatorCommand[]

- **Mode** is a byte indicating in which mode the PN531 has been activated:

7	6	5	4	3	2	1	0
nu	BaudRate			nu	nu	Framing Type	

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

00 : Mifare®
01 : Active mode
10 : Felica™

- **InitiatorCommand** is an array containing the first valid frame received by the PN531 once the PN531 has been initialized.
  This frame is different depending on the mode in which the PN531 has been initialized (DEP, passive 106 kbps or 212/424 kbps).

Description:

The host uses this command to configure the PN531 as a target.

When this command is used by the host controller, the PN531 first stores the input parameters in the dedicated area of the internal contactless UART and then activates the AutoColl command.
This AutoColl command handles FeliCa™ polling and Mifare® anticollision automatically.

Thus, **`TgInitTAMATarget`** is ended when a complete command frame has been received from the external initiator. Depending on the initialisation type, the 3 following scenarios are possible:

- 106 kbps passive:
  - when a SENS_REQ command is detected, the PN531 sends back the SENS_RES contained in **MifareParams**
  - then the PN531 uses the NFCID1t part of **MifareParams** during the anticollision process
  - at the end of the selection, the PN531 sends the SEL_RES to the initiator
  - then the PN531 waits for a command coming from the initiator that closes the AutoColl internal command.
  - This command may be an ATR_REQ, a SLP_REQ or a proprietary command.
    - *ATR_REQ*: if the flag *fAutomaticATR_RES* is set (§4.2.9, p.67), the PN531 sends back automatically the ATR_RES frame to the initiator (example in Fig 62).
      Otherwise, the ATR_RES will be sent back to the initiator only after having received a **`TgSetGeneralBytes`** (§4.3.14, p.125) from the host controller.
      The complete ATR_REQ is returned to the host controller.

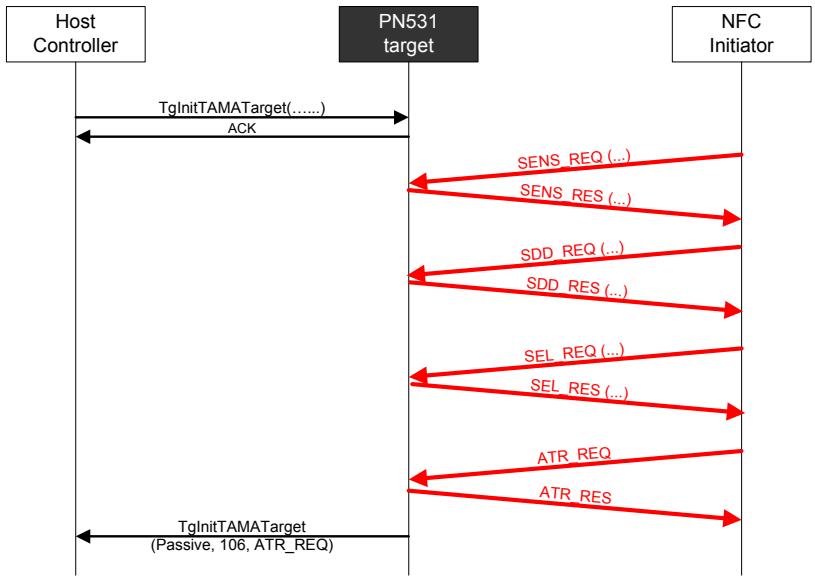


Fig 62. TgInitTAMATarget – Passive DEP 106 kbps

- *SLP_REQ:* the PN531 starts again an AutoColl sequence.
- *proprietary command:* the PN531 does nothing with this command (example in Fig 63).
  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 PN531 starts a new AutoColl sequence.

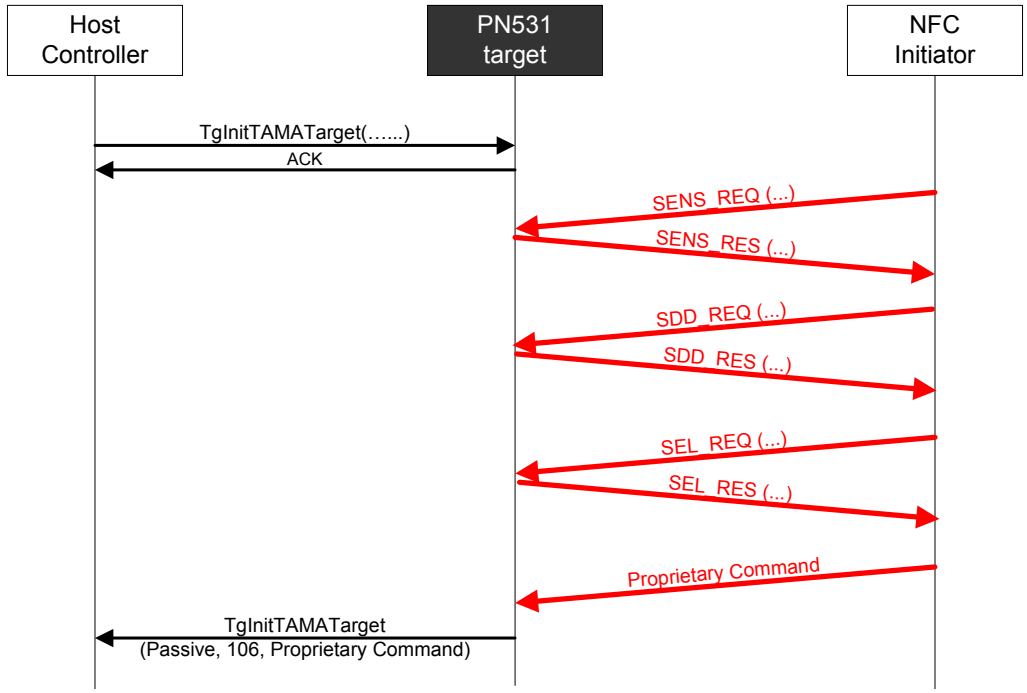


Fig 63. TgInitTAMATarget – Proprietary command

- 212/424 kbps passive:
  - when a POL_REQ command is detected, the PN531 sends back the POL_RES contained in **FeliCaParams**.
    If requested by the initiator (4th byte of the POL_REQ = 0x01), the **SystCode** information is added to in the POL_RES.
  - then the PN531 waits for a command coming from the initiator that closes the AutoColl process.
  - This command may be an ATR_REQ (Fig 64) or a proprietary command (Fig 65). In case of the reception of an ATR_REQ, the PN531 sends automatically the ATR_RES frame (except if the flag *fAutomaticATR_RES* is not set (§4.2.9, p.67); this is the case in Fig 64).

    If the bit *DEPOnly* is set, the command received must be an ATR_REQ.
    The PN531 refuses all other commands and starts a new AutoColl sequence

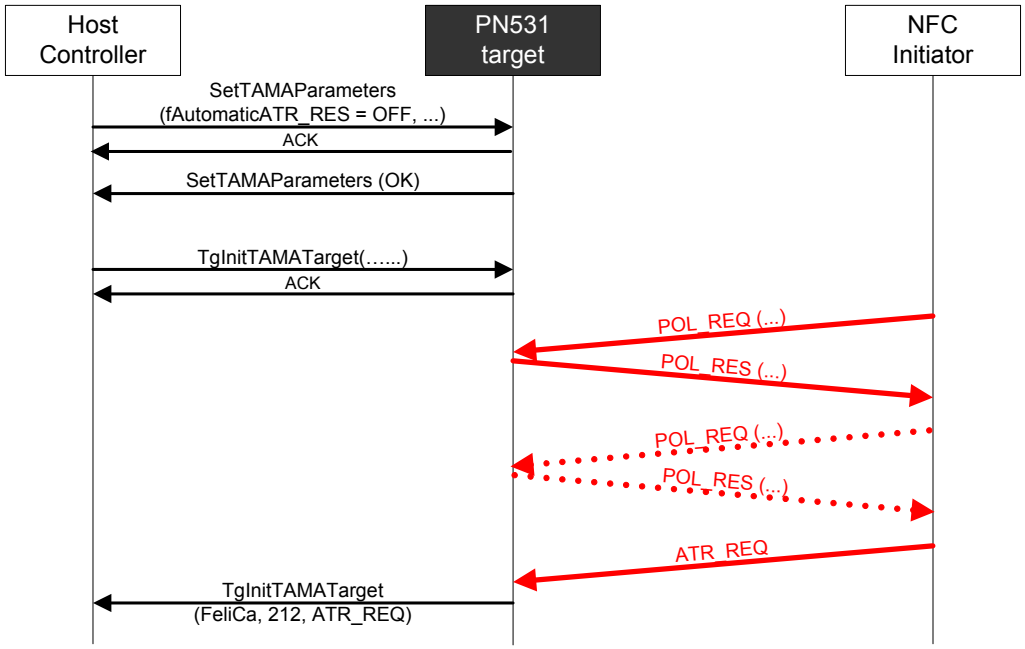


Fig 64. TgInitTAMATarget –Passive DEP 212/424 kbps

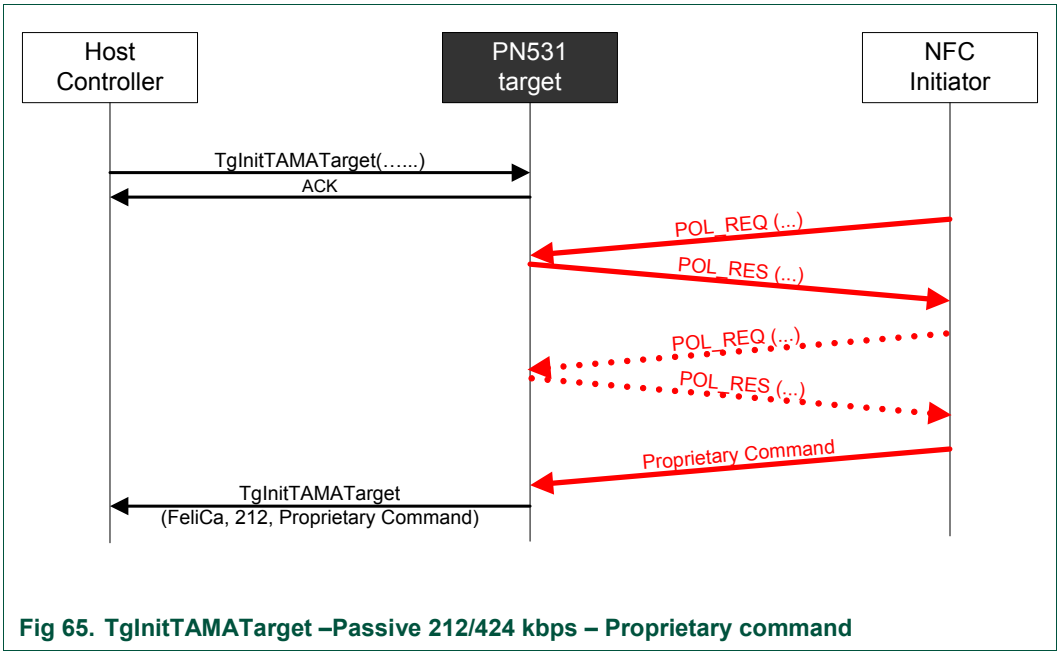


Fig 65. TgInitTAMATarget –Passive 212/424 kbps – Proprietary command

- 106/212/424 kbps active (if bit *PassiveOnly* is not set):
  - The PN531 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 PN531 starts a new sequence of AutoColl.
  - Depending on the flag *fAutomaticATR_RES* (§4.2.9, p.67), the PN531 sends automatically the ATR_RES frame (Fig 66) or not.
    In that case (Fig 67), the host controller uses the **`TgSetGeneralBytes`** (§4.3.14, p.125) to give the General Bytes that the PN531 puts in the ATR_RES.

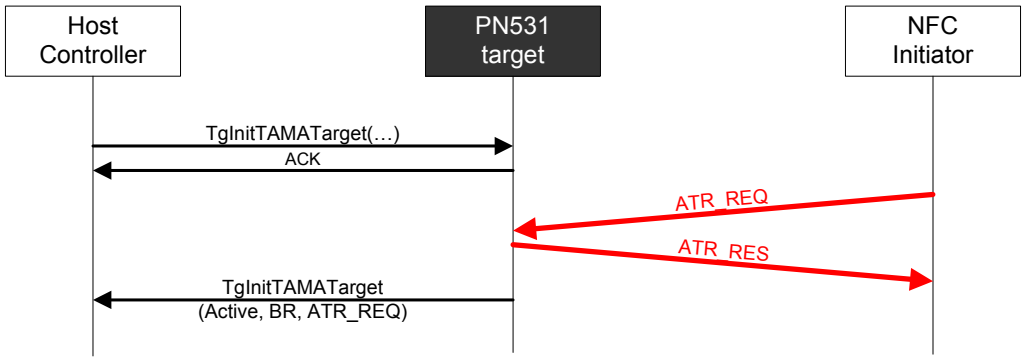


Fig 66. TgInitTAMATarget – Active mode

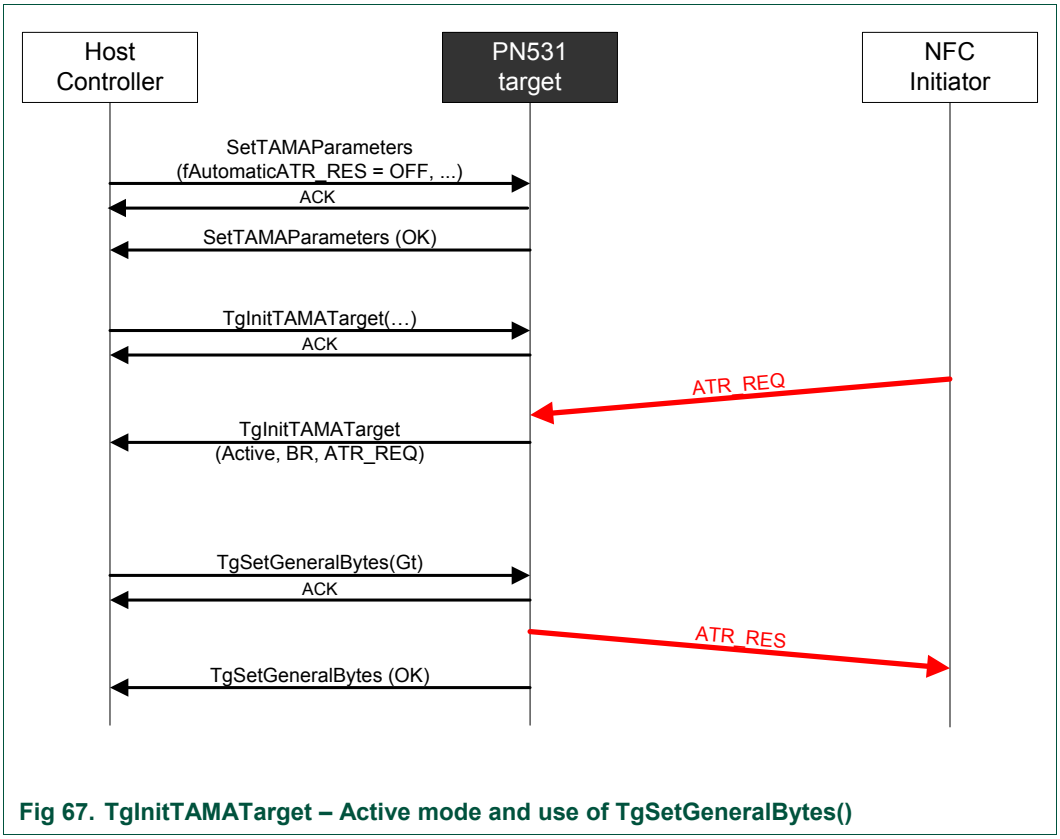


Fig 67. TgInitTAMATarget – Active mode and use of TgSetGeneralBytes()

Once the PN531 is configured as a target, it can handle some of the DEP commands without any help of its host controller.
The PN531 builds the corresponding answer frame and updates its internal state (released, deselected, activated, …).

This is the case for the following command frame:

Table 23: Target configuration – Automatic response

Command Received	Automatic Response
ATR_REQ	ATR_RES may need **`TgSetGeneralBytes`** if *fAutomaticATR_RES* is not set
PSL_REQ	PSL_RES
DSL_REQ	DSL_RES
RLS_REQ	RLS_RES
WUP_REQ	WUP_RES

4.3.14 TgSetGeneralBytes

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 §4.1, p.48).

Syntax Error Conditions:

- Gt[] length exceeds 47 bytes

Description:

This command is used in combination with the **`TgInitTAMATarget`** command (§4.3.13, p.118) to give the General Bytes.

The PN531 uses them to build the ATR_RES sent to the initiator.

By default (flag *fAutomaticATR_RES* set, §4.2.9, p.67), the PN531 uses the general bytes given in **`TgInitTAMATarget`** 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 **`TgInitTAMATarget`**, as described in the following figure (Fig 68).

The PN531 does not send ATR_RES before the command **`TgSetGeneralBytes`**. Then, the PN531 prepares the ATR_RES with the **Gt[]** bytes and sends the complete ATR_RES to the initiator.

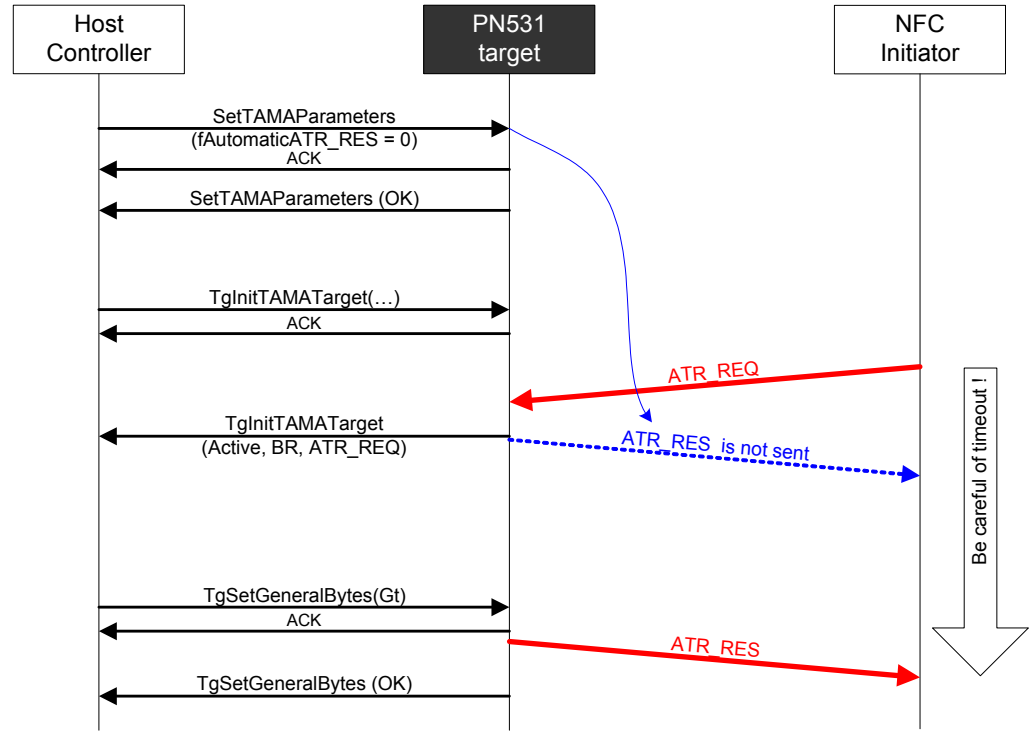


Fig 68. TgSetGeneralBytes

Remark: The NFC initiator controls a timeout after having sent an ATR_REQ (ref. **[1]**), so the host controller of the PN531 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.

4.3.15 TgGetDEPData

Input:

D4	86

Output:

D5	87	Status	[DataIn[]]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48).
  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 §4.4.5, p.143).
- **DataIn** is an array of data received by the PN531 coming from the initiator.

Description:

This command is used in case of the PN531 configured as a target for Data Exchange Protocol (DEP).

It allows the host controller to get back the data received by the PN531 from its initiator (in **DataIn**[] array).

In case of the PN531 configured as a target, 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 PN531 control (the host controller may take a long time to prepare the data to be returned).
To bypass this potential problem, the PN531 will automatically generates the necessary Supervisory pdu ($S(TO)_{REQ}$).

The difference between this command and the **`TgGetInitiatorCommand`** command (§4.3.18, p.133), is that this one is dedicated to the NFC Data Exchange Protocol (DEP) as the other one may be used to carry information whatever the protocol used.

A typical data exchange between the PN531 as a target and a NFC Initiator can be represented as follows (Fig 69):

- when the host controller wants to retrieve a command message coming from the initiator, it uses the **`TgGetDEPData`** command.
- In that case, the PN531 sends back an ACK frame to the host 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 PN531 uses a supervisory frame to ask for time extension to the initiator (7 x 154 ms = 1.078s)[7].

[7] 7 is the default value of the RTOX parameter sent by the PN531 in a $S(TO)_{REQ}$.
154 ms corresponds to the default value of RWT (Response Waiting Time) for the PN531 configured as a target.

- Then, the PN531 can send the received RF frame back to the host controller.
- After having processed these data, the host controller will use the **`TgSetDEPData`** command to send to the PN531 the RF answer frame. The PN531 transfers this frame to the initiator by using the DEP mechanism.

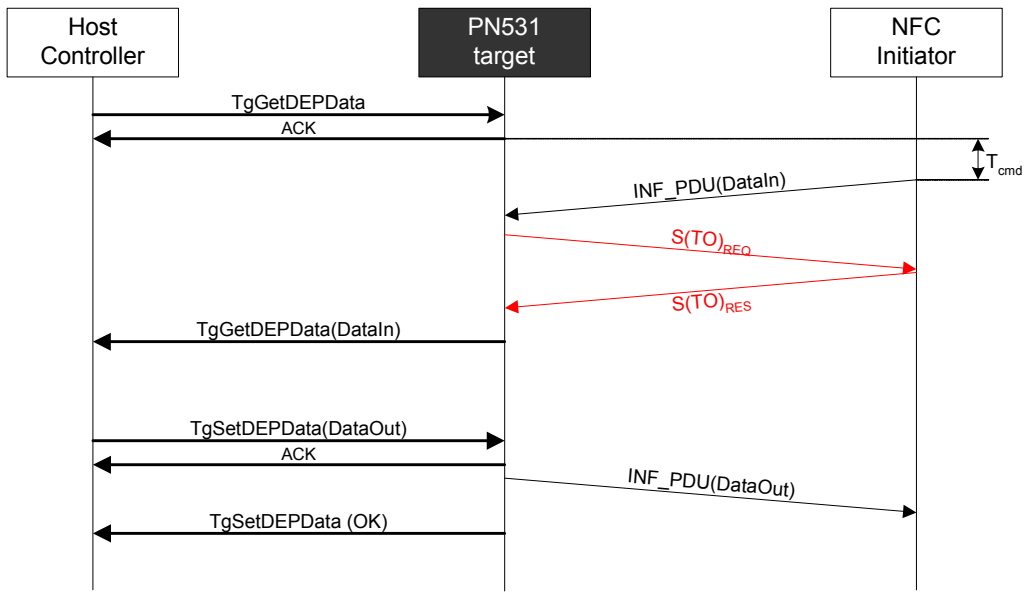


Fig 69. TgGetDEPData (1)

In this schematic representation, no chaining is shown.
Refer to §4.4.5, p.143 to have a more detailed explanation of the chaining mechanism in case of the PN531 configured as a target.

Possible errors returned:

- Target is not in a correct state to perform this operation (not in DEP protocol) ➔ error 0x25
- Target has been deselected ➔ error 0x29

The PN531 can also accept a INF_PDU incoming frame from the initiator even if it has not received yet a **`TgGetDEPData`** command from the host controller. The protocol exchange with the initiator is then preserved by using $S(TO)_{REQ}$.

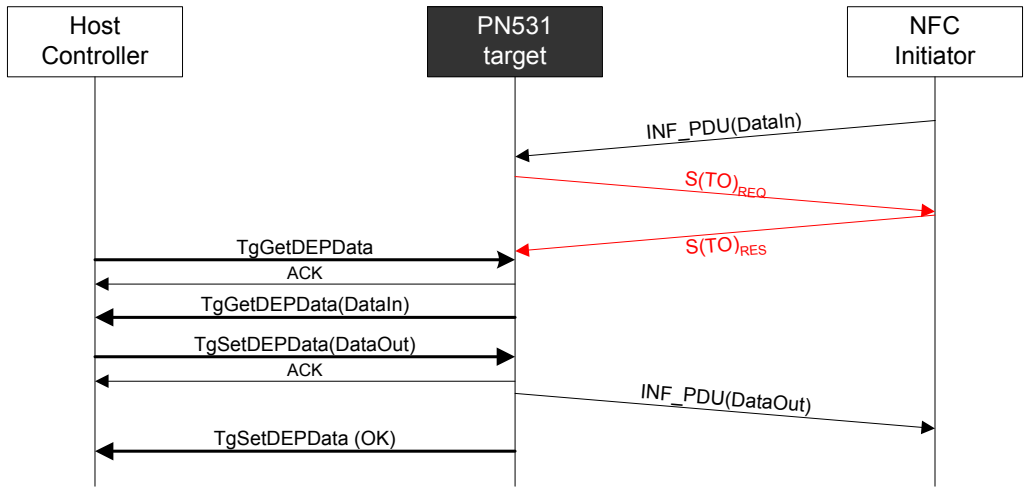


Fig 70. TgGetDEPData (2)

4.3.16 TgSetDEPData

Input:

D4	8E	[DataOut[]]

- **DataOut** is an array of data to be sent by the PN531 as a response to its initiator. The total length of data is determined by the host ⇔ PN531 protocol layer.

Output:

D5	8F	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48)

Description:

This command is used in case of the PN531 configured as a target for Data Exchange Protocol (DEP).

It allows the host controller to supply the PN531 with the data that it wants to send back to the initiator (in response of the previous RF DEP_REQ frame(s)).

The PN531 sends in the RF link the data contained in **DataOut**[] array.
The protocol management (chaining, error handling) is completely managed internally by the PN531.

A typical data exchange between the PN531 as a target and a NFC Initiator is represented in the **`TgGetDEPData`** command description.

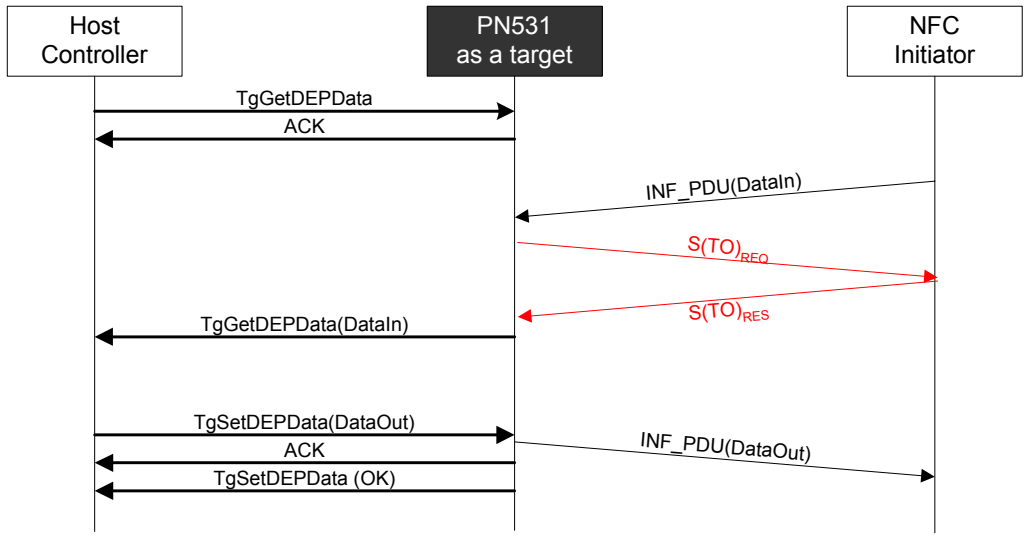


Fig 71. TgSetDEPData

The examples given in §4.4.5, p.143 show how the chaining is handled either by the initiator or by the target.

4.3.17 TgSetMetaDEPData

Input:

D4	94	DataOut[]

- **DataOut** is an array of data to be sent by the PN531 as a response to its initiator. The total length of data is determined by the host ⇔ PN531 protocol layer.

Output:

D5	95	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48)

Description:

This command is nearly the same than the **`TgSetDEPData`** one (see §4.3.15, p.127), except that it is used when the host controller of the target wants to transfer data which length is greater than what the PN531 can send in a single transaction.

The main difference compared to the previous command is therefore that in the last chained packet sent by the PN531 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:

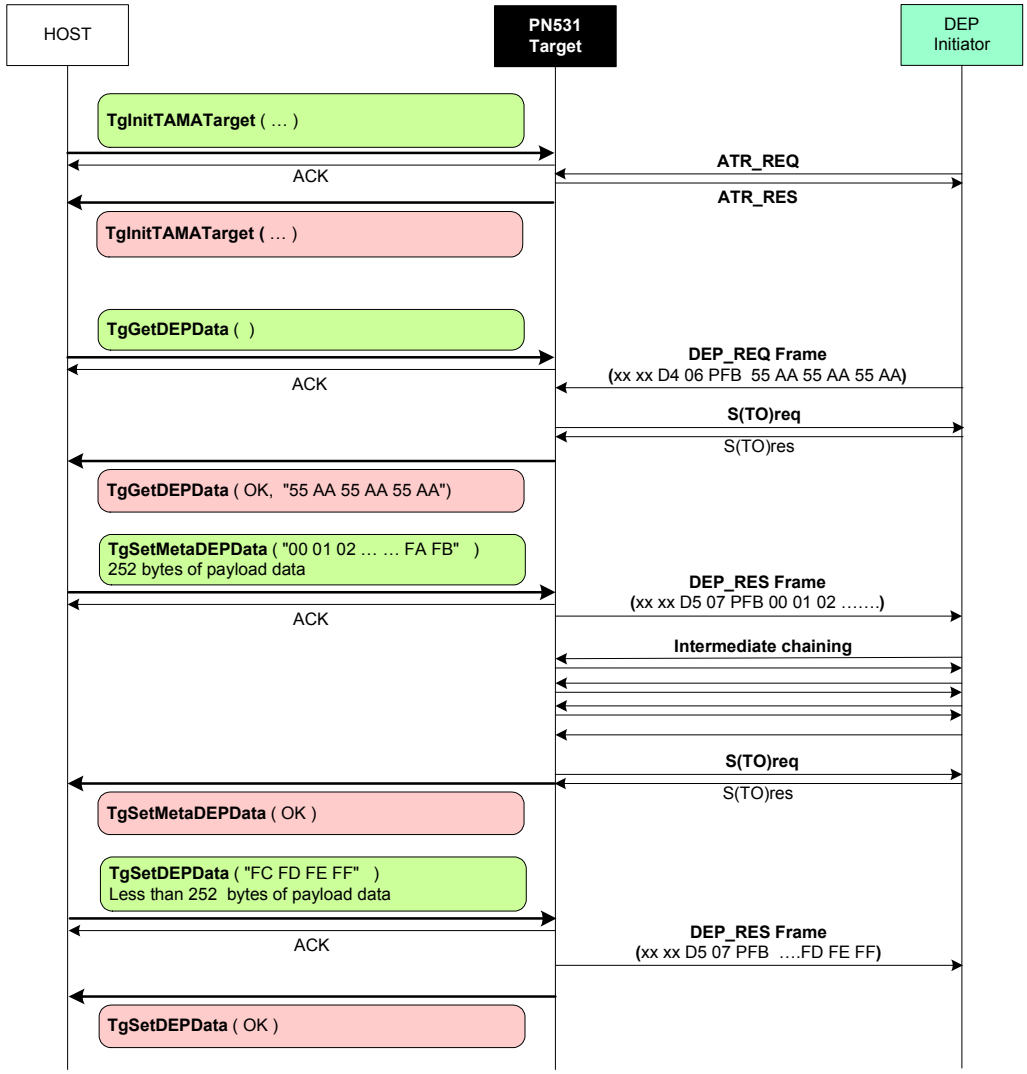


Fig 72. TgSetMetaDEPData

4.3.18 TgGetInitiatorCommand

Input:

D4	88

Output:

D5	89	Status	InCommand[]

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48).
- **InCommand** is an array of raw data received by the PN531 (command of the initiator).

Description:

This command is used when the PN531 is configured as a target (see **`TgInitTAMATarget`**, §4.3.13, p.118).

This command is used to get a packet of data from an initiator and to send it back to the host controller. The received data are simply returned to the host controller (in **InCommand[]** array) that will process them and then use the **`TgResponseToInitiator`** command (§4.3.19, p.135) 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.

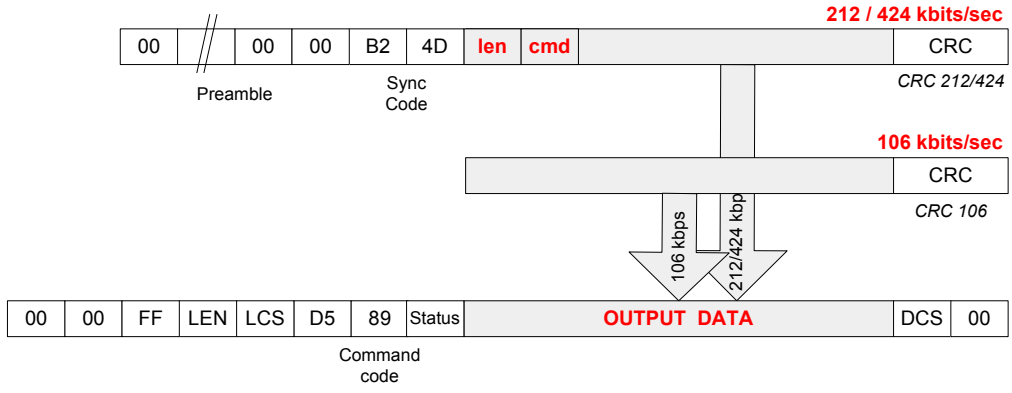


Fig 73. TgGetInitiatorCommand (1)

This command is complementary of the **`TgGetDEPData`** command.
The main difference compared to **`TgGetDEPData`** is that here the PN531 does not handle at all the protocol (DEP) features (Supervisory, chaining, error handling, …).

So, this command combined with **`TgResponseToInitiator`** (§4.3.19, p.135) may be used to build exchanges without using the protocol handled by the PN531 (DEP).

The following figure depicts the linking of the exchanges between the initiator and the PN531 and between the host controller and the PN531.

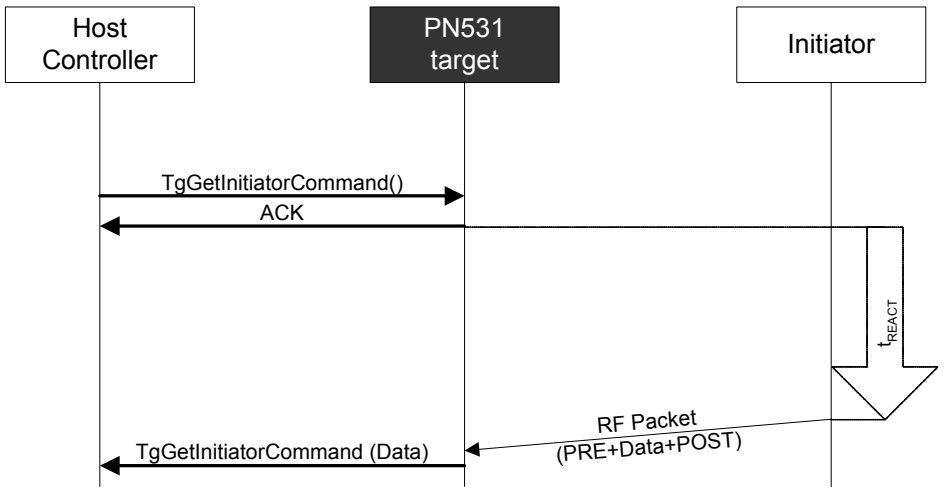


Fig 74. TgGetInitiatorCommand (2)

No control is done on the delay (t_{REACT}) used by the initiator to send its command frame. The host 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).

4.3.19 TgResponseToInitiator

Input:

D4	90	TgResponse[]

- **TgResponse** is an array of raw data to be sent by the PN531 (response to the initiator).

Output:

D5	91	Status

- **Status** is a byte indicating if the process has been terminated successfully or not (see §4.1, p.48).

Description:

This command is used to send a response packet of data to an initiator. It is usually used in co-operation with **`TgGetInitiatorCommand`** (§4.3.18, p.133).

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.

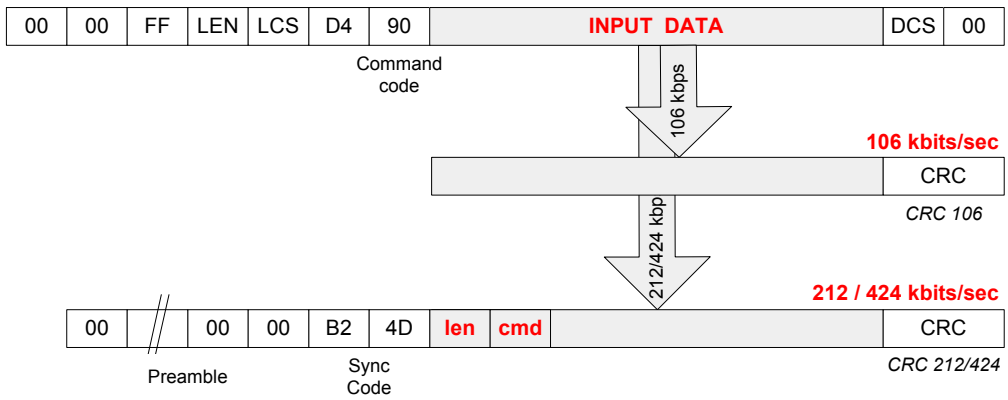


Fig 75. TgResponseToInitiator (1)

This command is complementary of the **`TgSetDEPData`** command.
The main difference compared to **`TgSetDEPData`** is that here the PN531 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`** command from the initiator side.

The following figure depicts how the exchanges between the initiator and the PN531 and between the host controller and the PN531 are cascaded.

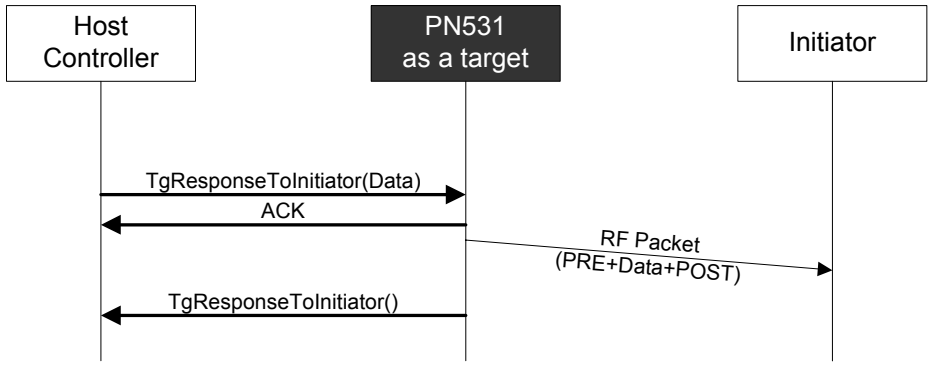


Fig 76. TgResponseToInitiator (2)

The **`TgResponseToInitiator`** command ends as soon as the RF frame is sent to the initiator.
If no error is detected, the status byte is cleared otherwise the **Status** byte is filled in with an error code indicating the communication error.

4.3.20 TgGetTargetStatus

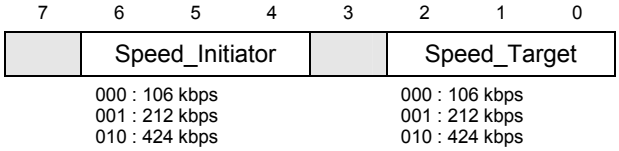
Input:

D4	8A

Output:

D5	8B	State	BRit

- **State** gives information about the current state of the PN531.
  - 0x00 IDLE / RELEASED
    the PN531 waits for an initiator or has been released by its initiator
  - 0x01 ACTIVATED
    the PN531 is activated as a target
  - 0x02 DESELECTED
    the PN531 has been de-selected by its initiator
- **BRit** gives information about the baud rate used only when in ACTIVATED state:



Description:

This command is used by the host controller to know what is the current state of the PN531.

The goal of this command is to offer the possibility for the host controller to know if the PN531 target has been either de-selected or released.

In addition, the baud rates used in both directions are also returned.

4.4 Commands summary

The host controller has several commands that can be used when the PN531 is configured either as an initiator or as a target:

4.4.1 Commands for Initiator mode

The Fig 77 summarizes all the possible commands that can be used when the PN531 is configured as an initiator.

Initialisation / Activation:

- InJumpForDEP
- InJumpForPSL
- InListPassiveTarget
- InATR
- InPSL

Data Exchange:

- InDataExchange
- InCommunicateThru

Selection / De-Selection / Release:

- InSelect
- InDeselect
- InRelease

4.4.2 Commands for Target mode

The Fig 78 summarizes all the possible commands that can be used when the PN531 is configured as a target.

Initialisation:

- TgInitTAMATarget
- TgSetGeneralBytes

Data Exchange:

- TgGetDEPData
- TgSetDEPData
- TgSetMetaDEPData
- TgGetInitiatorCommand
- TgResponseToInitiator

4.4.3 Availability of the commands

The following table lists the commands relative to initialisation / communication / deactivation of targets.

For each command, the availability is indicated in case of:

- active or passive communication mode
- target is Transport Protocol Equipped (TPE) or not

Table 24: Availability of the commands

Command	Communication mode		Data Exchange Protocol	
	Passive	Active	Not TPE	TPE
InDataExchange	x	x	x	x
InCommunicateThru	x	x	x	x
InListPassiveTarget	x	N/A	x	x
InPSL	x	x	N/A	x
InATR	x	N/A	N/A	x
InSelect	x	x	x	x
InDeselect	x	x	x	x
InRelease	x	x	x	x
InJumpForDEP	x	x	N/A	x
InJumpForPSL	x	x	N/A	x
TgGetDEPData	x	x	N/A	x
TgGetInitiatorCommand	x	x	x	x
TgInitTAMATarget	x	x	x	x
TgSetDEPData	x	x	N/A	x
TgSetMetaDEPData	x	x	N/A	x
TgResponseToInitiator	x	x	x	x
TgSetGeneralBytes	x	x	N/A	x

4.4.4 Target states summary

The Fig 79 details all the possible states for the PN531 configured as a target in passive communication mode.

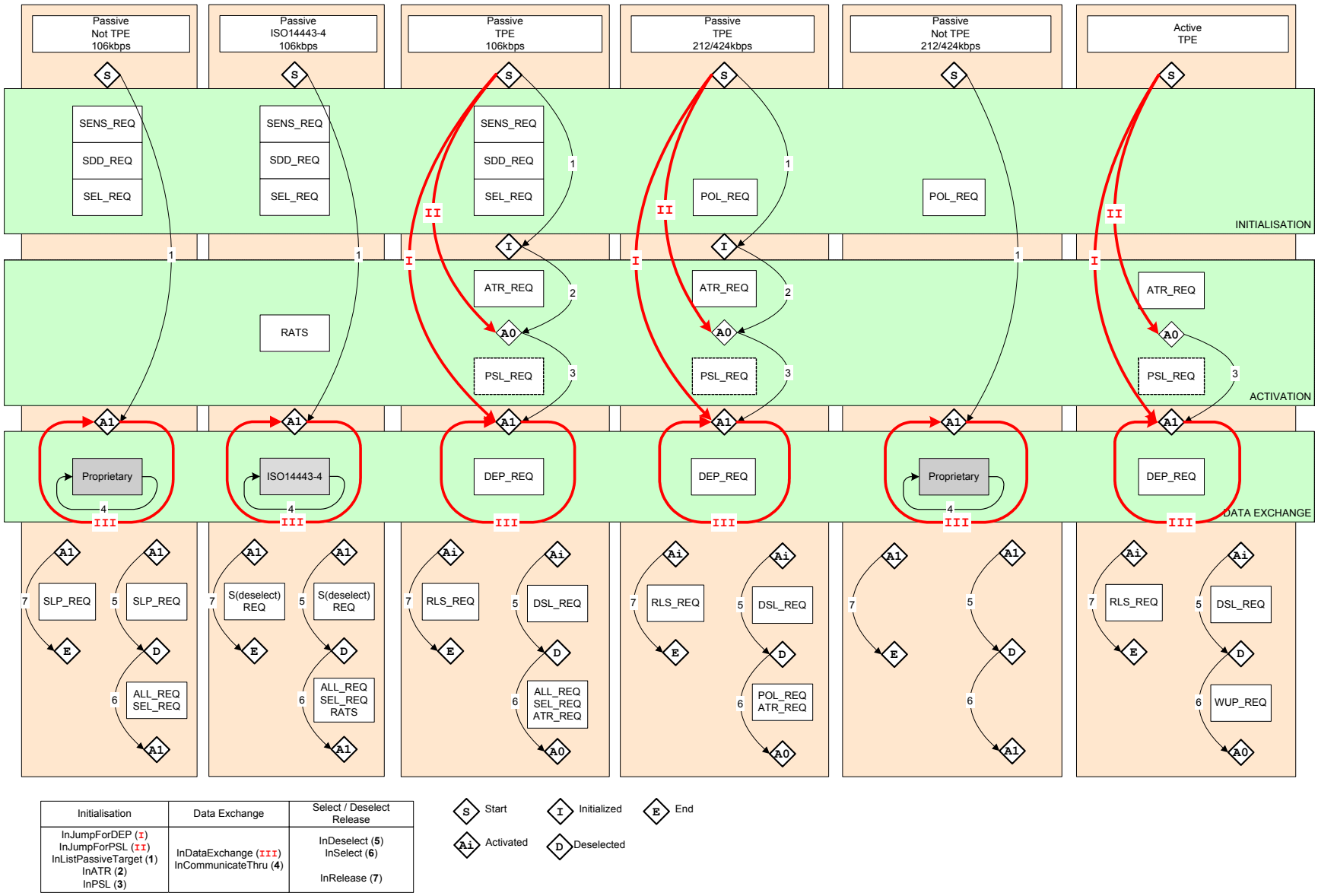


Fig 77. Initiator commands

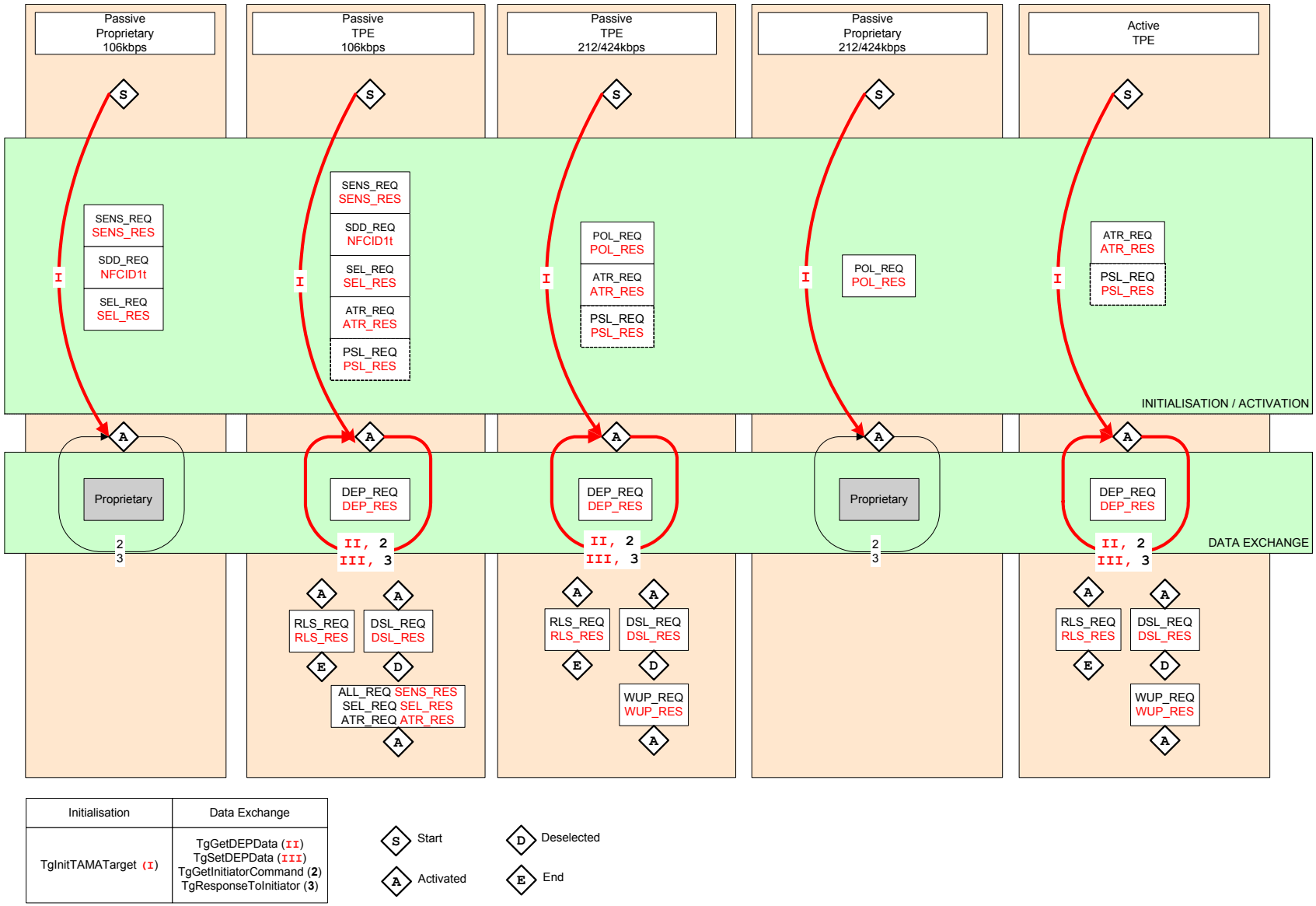


Fig 78. Target commands

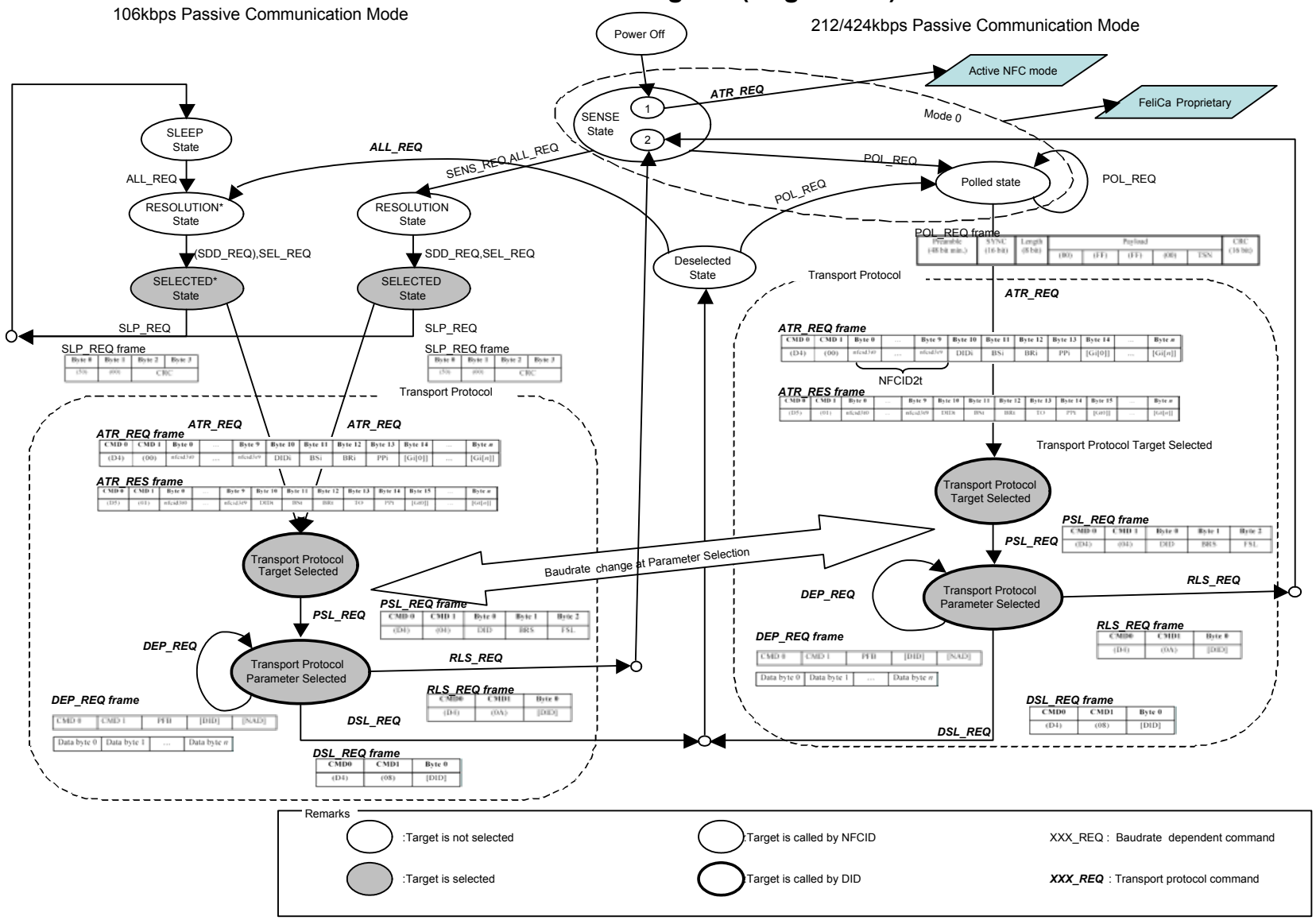


Fig 79. Target states

4.4.5 Chaining mechanism

This chapter details how the PN531 configured as an initiator or as a target handles the DEP chaining mechanism. Four examples are given.

The following symbolic representation is used:

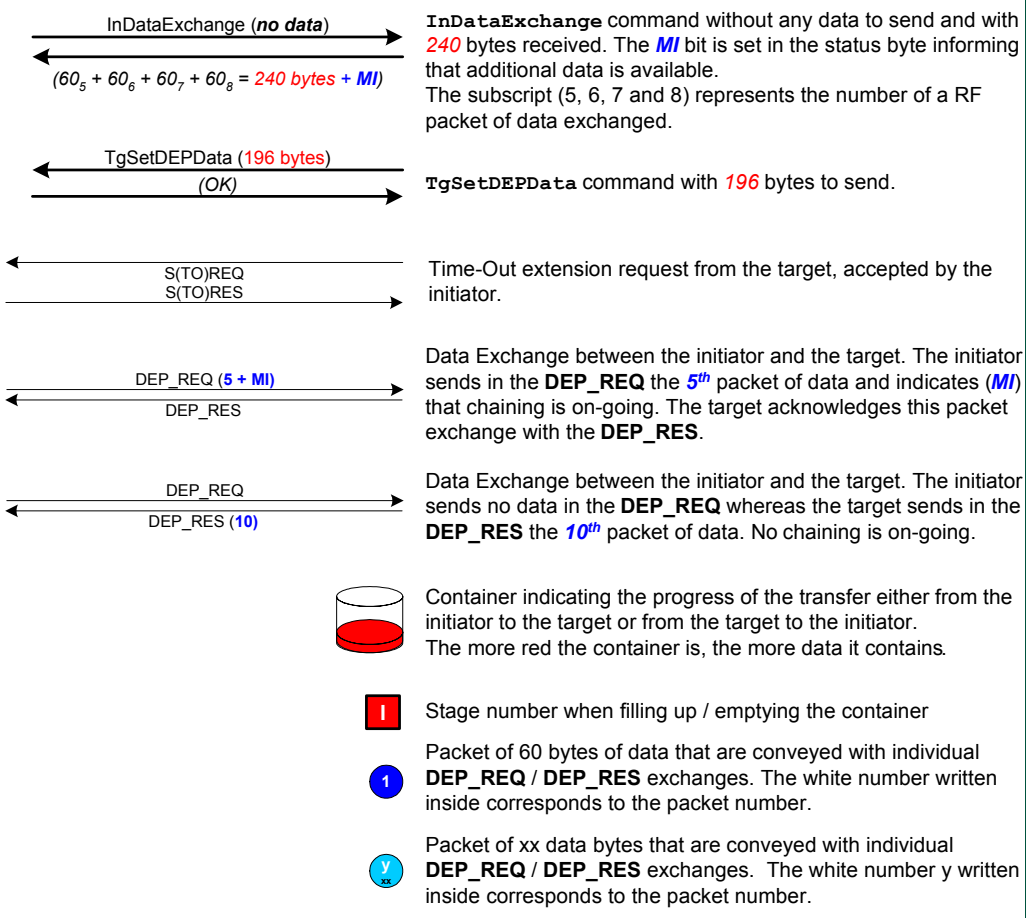


Fig 80. Legend used for the figures describing the chaining of data

In the *first example* shown (Fig 81), both the initiator and the target are supposed to be a PN531 which are **not using** Meta-Chaining.
The host controller of the initiator (A) sends packets of 252 bytes (maximal capacity of the **InDataExchange** command, see §4.4.6, p.149).
The target indicates a length reduction of 64 bytes, i.e. payload of 60 bytes[8].

The PN531 initiator cuts out the 252 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 PN531 target sends back a $S(TO)_{REQ}$ RF frame to the initiator, and sends the information data (252 bytes) to its host controller (**B**).

In this example, **B** knows that these first 252 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 **TgSetDEPData** without data.

When the PN531 target receives this **TgSetDEPData** 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 82), the initiator has a large memory area, meaning that the complete "file" to be transferred is ready in this memory.

The story 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 only thing that is really changed compared to the first example is that the PN531 target returns always the **MI** information to the host controller **B**. In that case, **B** does not need to use **TgSetDEPData** as in the first example.

In the *third and fourth examples*, the PN531 uses Meta-Chaining functionality to allow transfer of "large" amount of data.

The Fig 83 represents the case of data to be transferred from the initiator to the target and the Fig 84 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 **TgGetDEPData** commands.

On the other hand, in the fourth example, the comparative use of the **TgSetMetaDEPData** and **TgSetDEPData** commands is shown.

[8] 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 **Cmd1**, **Cmd2** and **PFB** bytes to deduct
(see Reference[**1**], §12.1 and Fig.23).

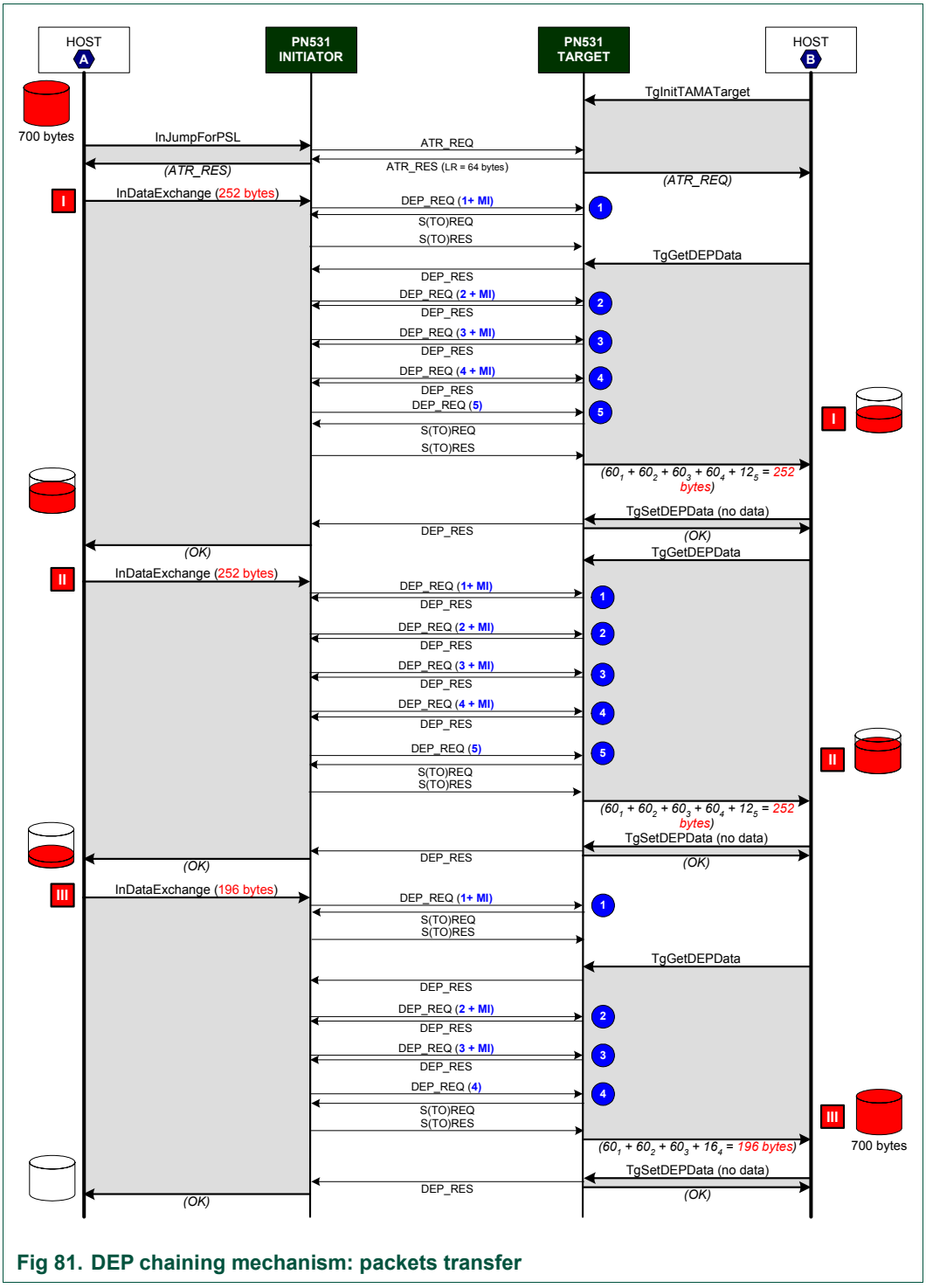


Fig 81. DEP chaining mechanism: packets transfer

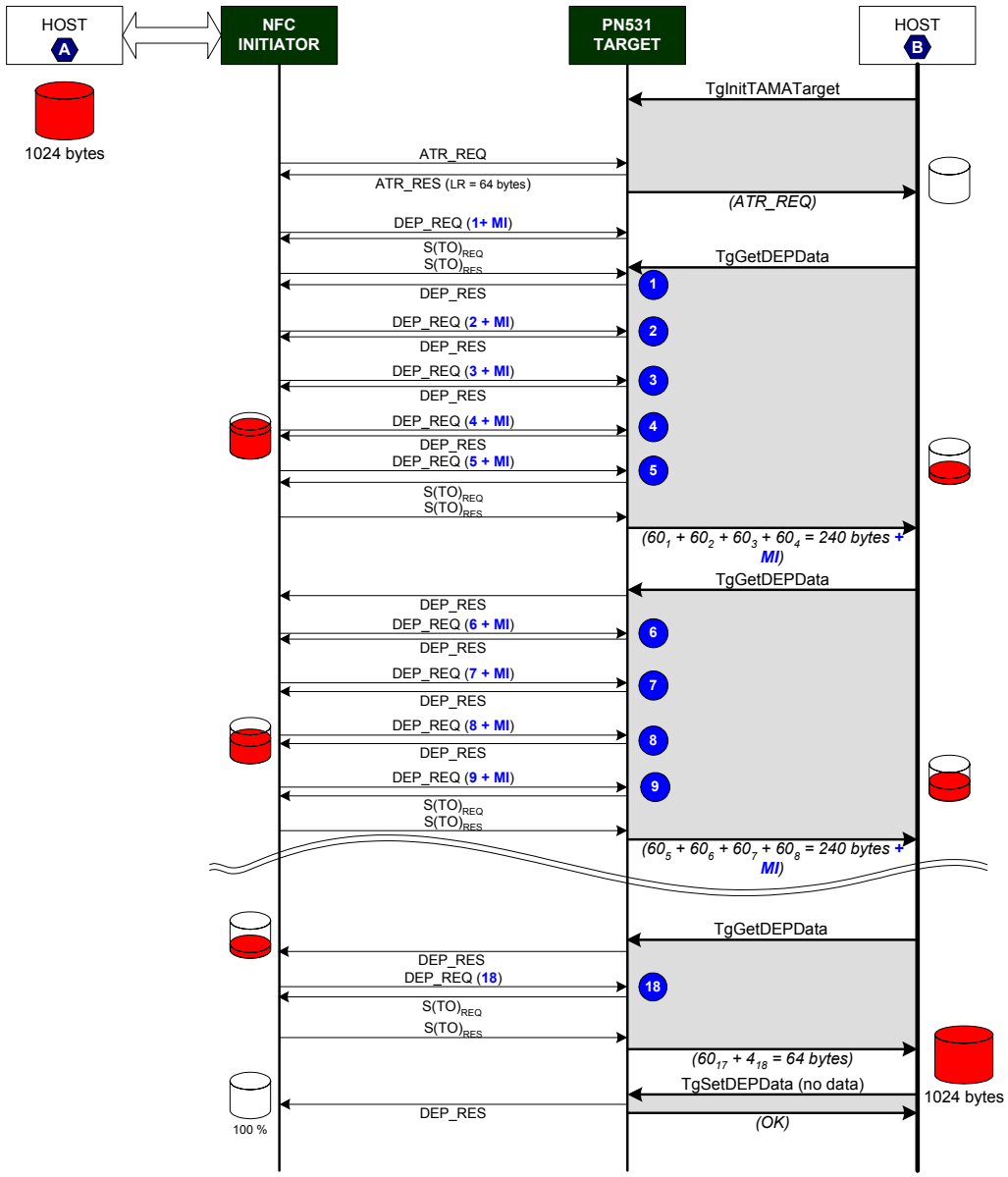


Fig 82. DEP chaining mechanism: “streaming” transfer from initiator to target

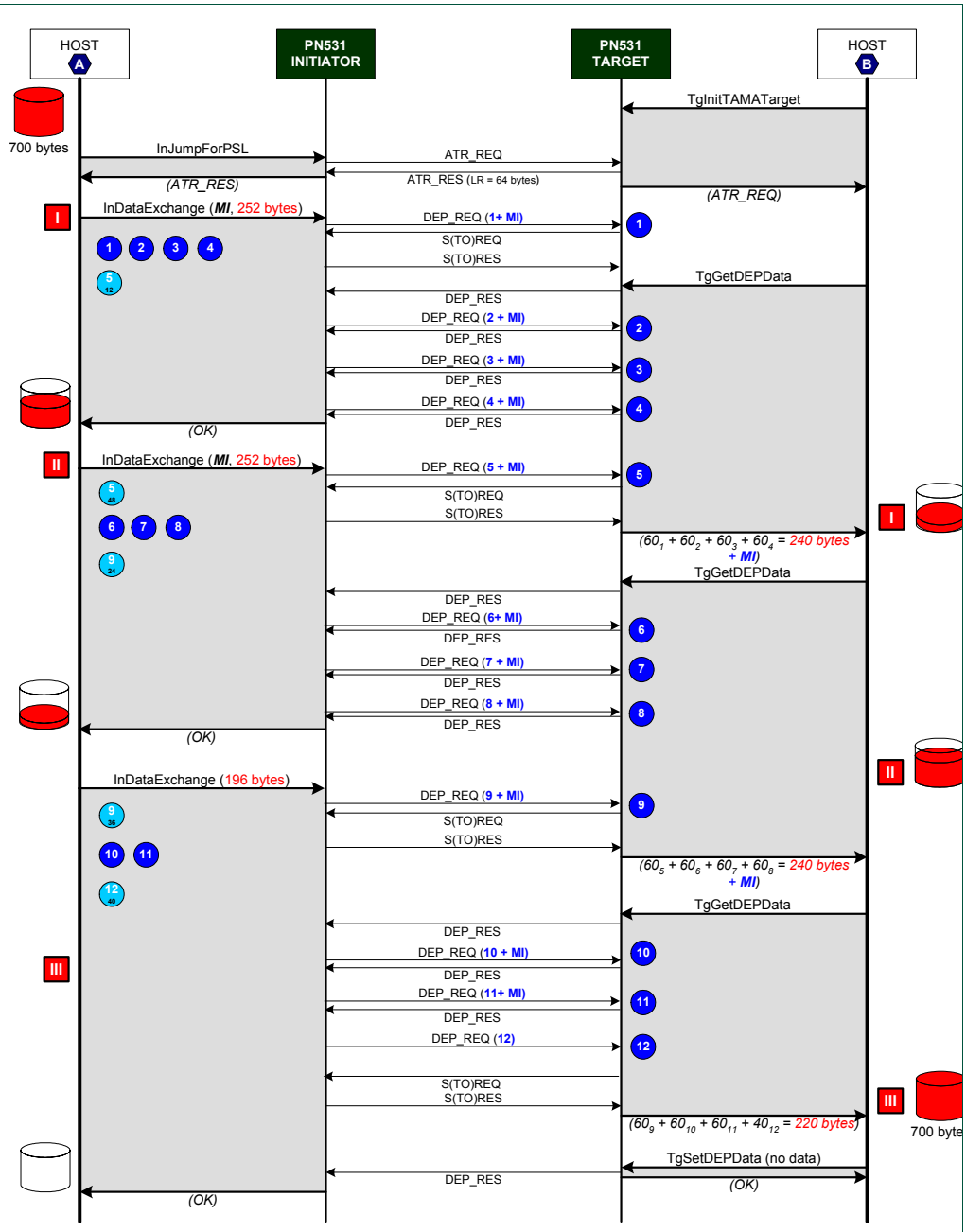


Fig 83. DEP chaining mechanism: transfer with Meta-Chaining – Initiator case

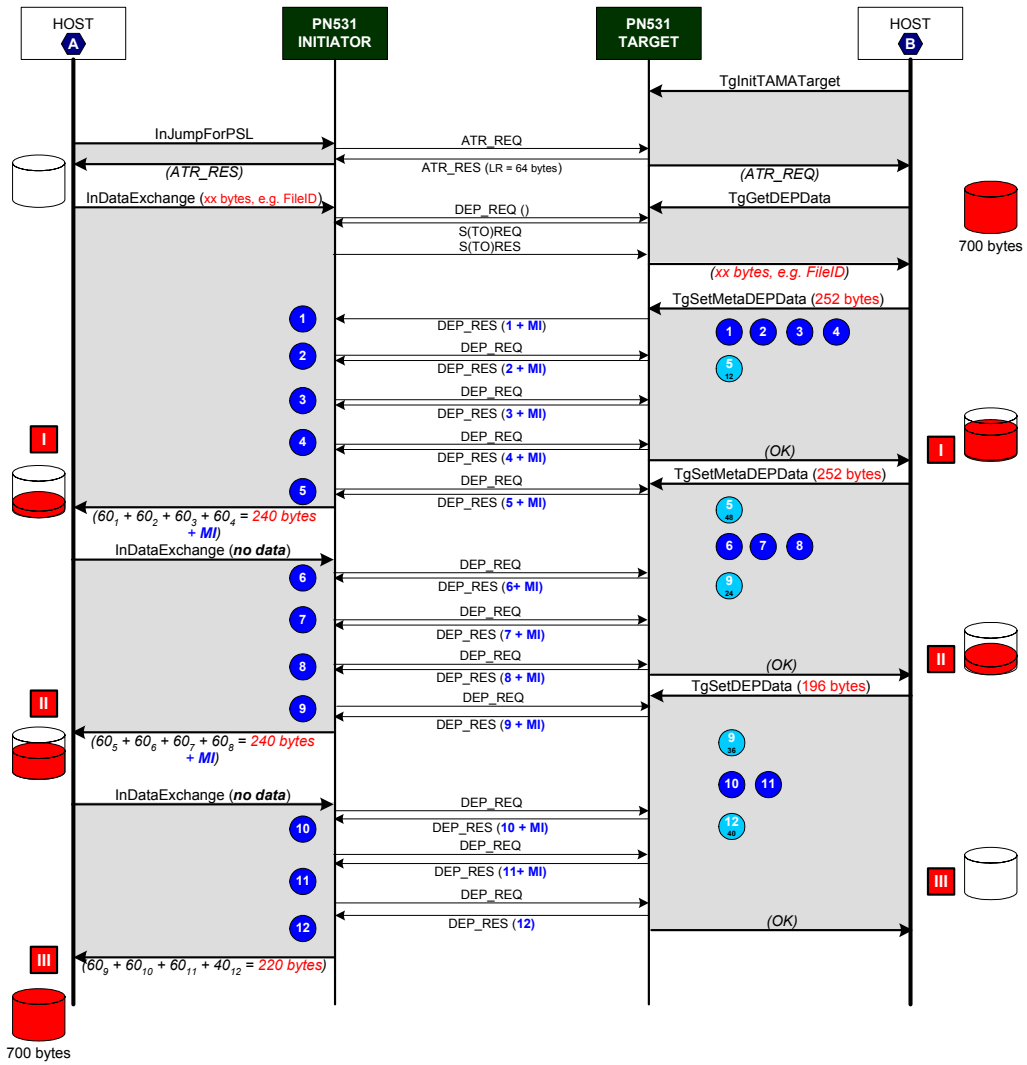


Fig 84. DEP chaining mechanism: transfer with Meta-Chaining – Target case

4.4.6 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`**
  - **`TgGetDEPData`**, **`TgSetDEPData`** and **`TgSetMetaDEPData`**
- 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 PN531, even when DID field is not used (251 bytes vs. 252).

That means that if the host controller uses the total capacity of the **`InDataExchange`** command, the PN531 will have to handle chaining even with a target having a length reduction of 255 bytes (example shown in Fig 81).

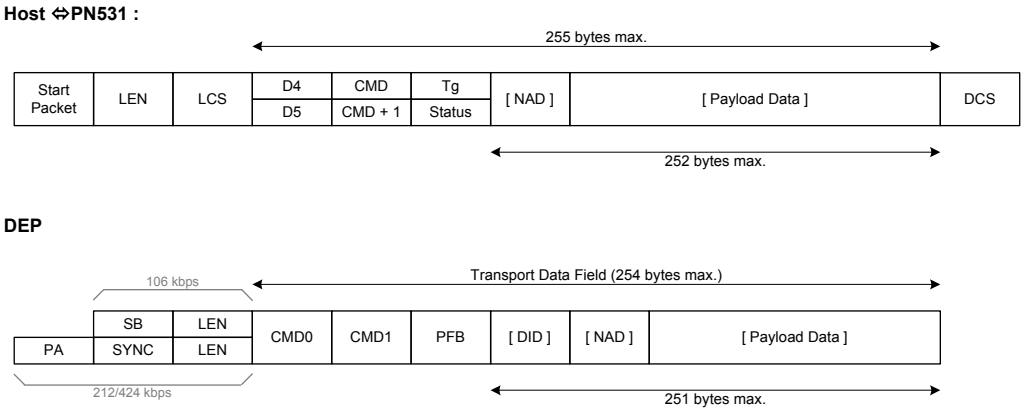


Fig 85. Payload data field maximum capacity

4.5 Examples of use

This paragraph gives some examples of use, detailing the commands used.

4.5.1 PN531 acting as a Mifare® Reader/Writer

The following example describes a short session with a Mifare® Standard card.

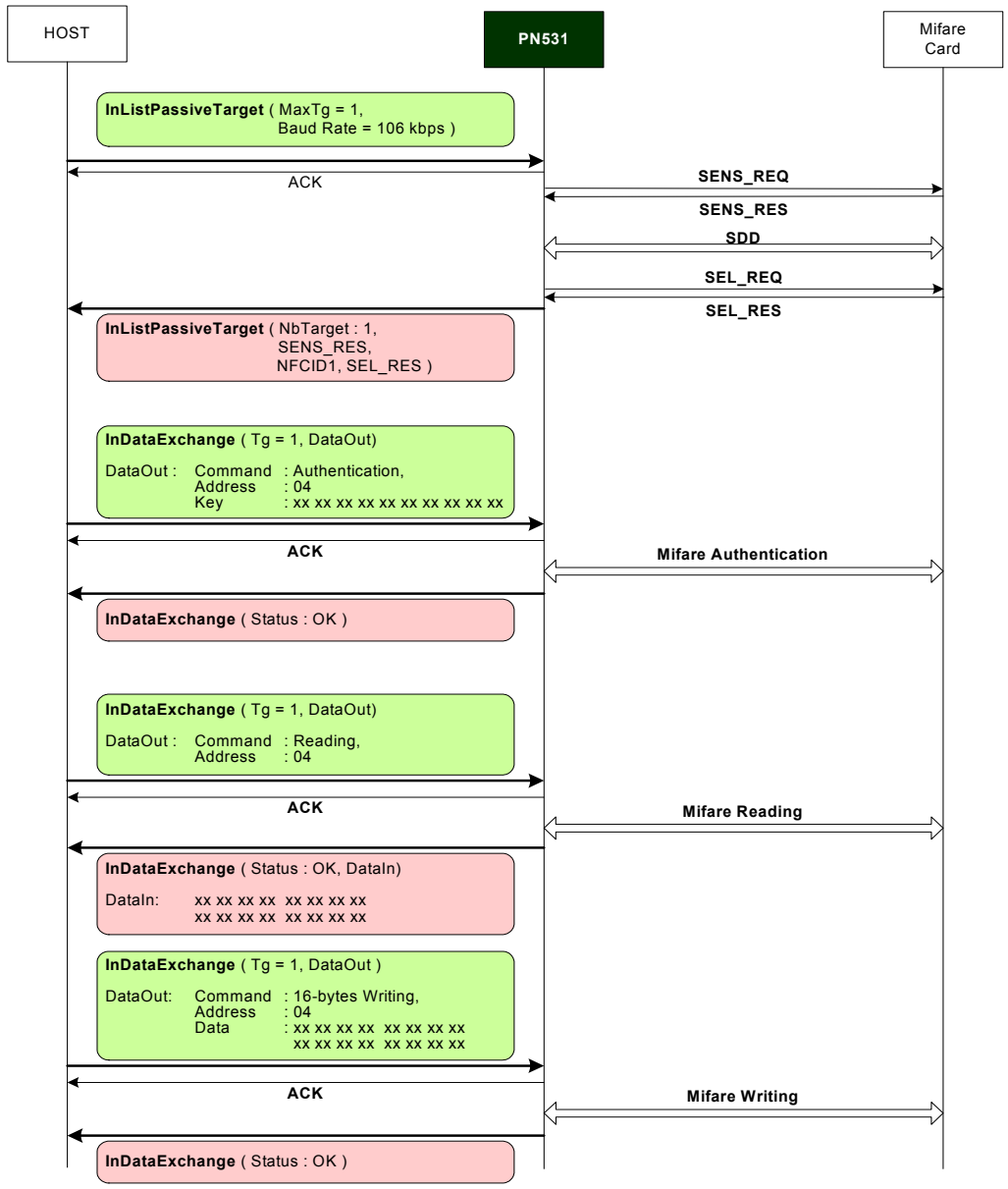


Fig 86. Mifare® Reader/Writer example

The first step consists of initializing the Mifare® card (the PN531 does not answer to the host controller as long as there is no target detected).
Then, in the second step, the PN531 makes the authentication of the Mifare® card to allow reading and writing operations.

4.5.2 PN531 acting as a FeliCa™ Reader/Writer

The following example describes a short session with a FeliCa™ card.

The first step consists of initializing the FeliCa™ card (the PN531 does not answer to the host controller as long as there is no target detected). Then, in the second step, the PN531 exchanges data with the FeliCa™ card using the **`InDataExchange`** command.

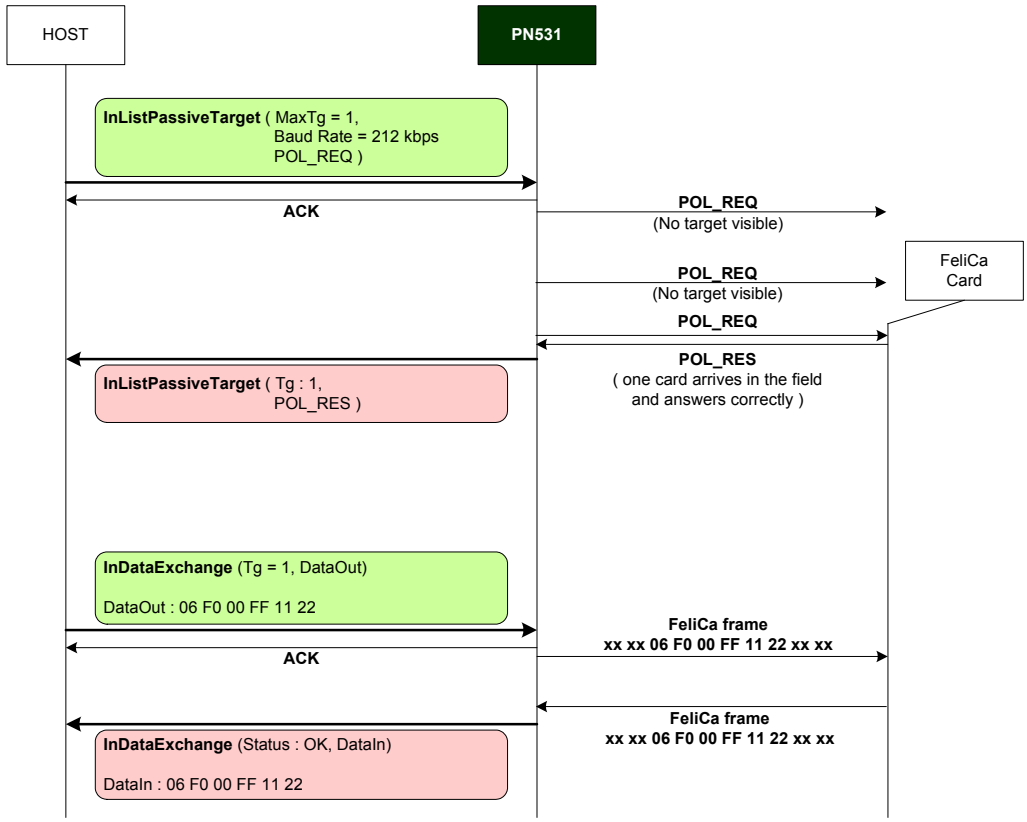


Fig 87. FeliCa™ Reader/Writer example

4.5.3 PN531 acting as a 106 kbps target

This example shows how the PN531 behaves in front of a Mifare® Reader/Writer, when it has been configured as a target:

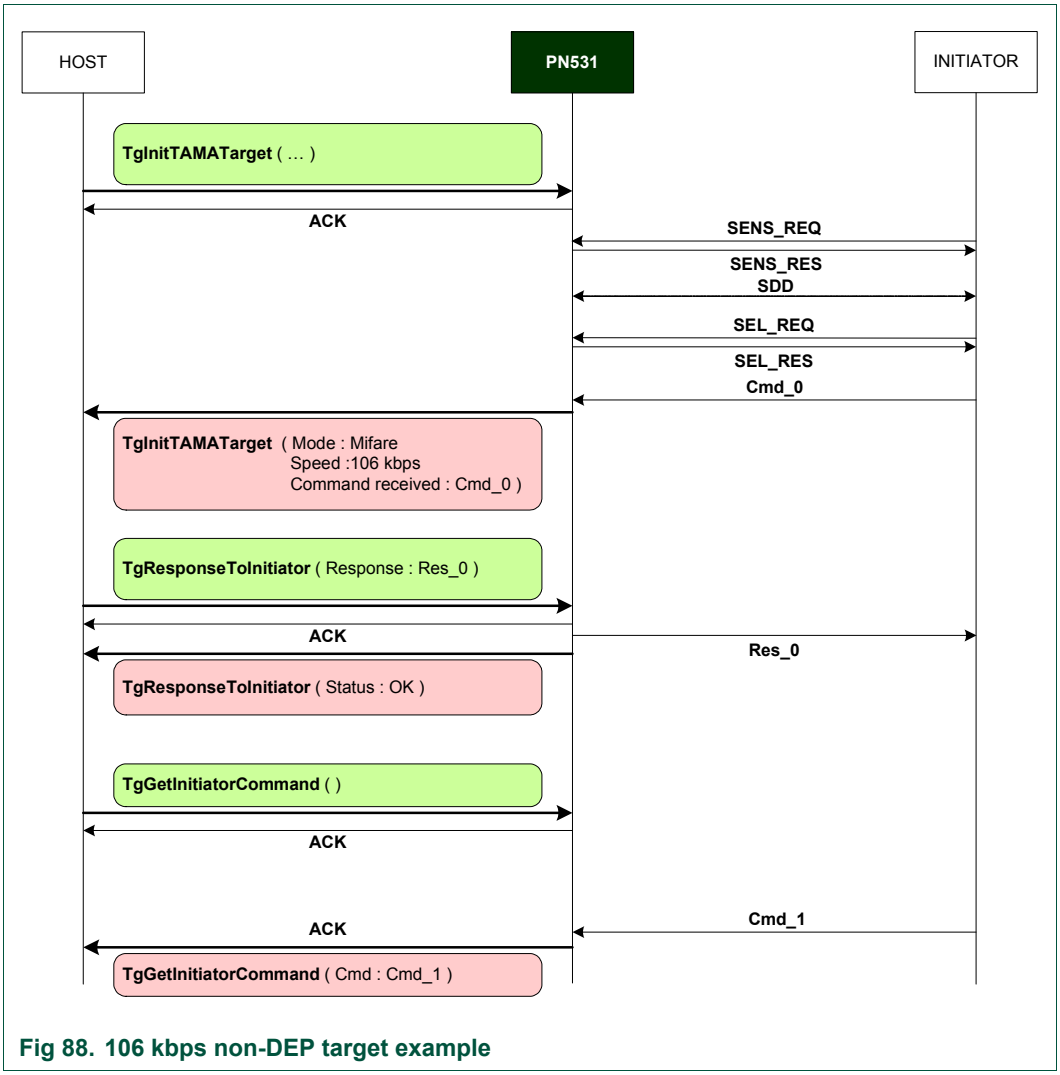


Fig 88. 106 kbps non-DEP target example

4.5.4 PN531 acting as a 212 kbps target

This example shows how the PN531 behaves in front of a 212 kbps initiator, when it has been configured as a target:

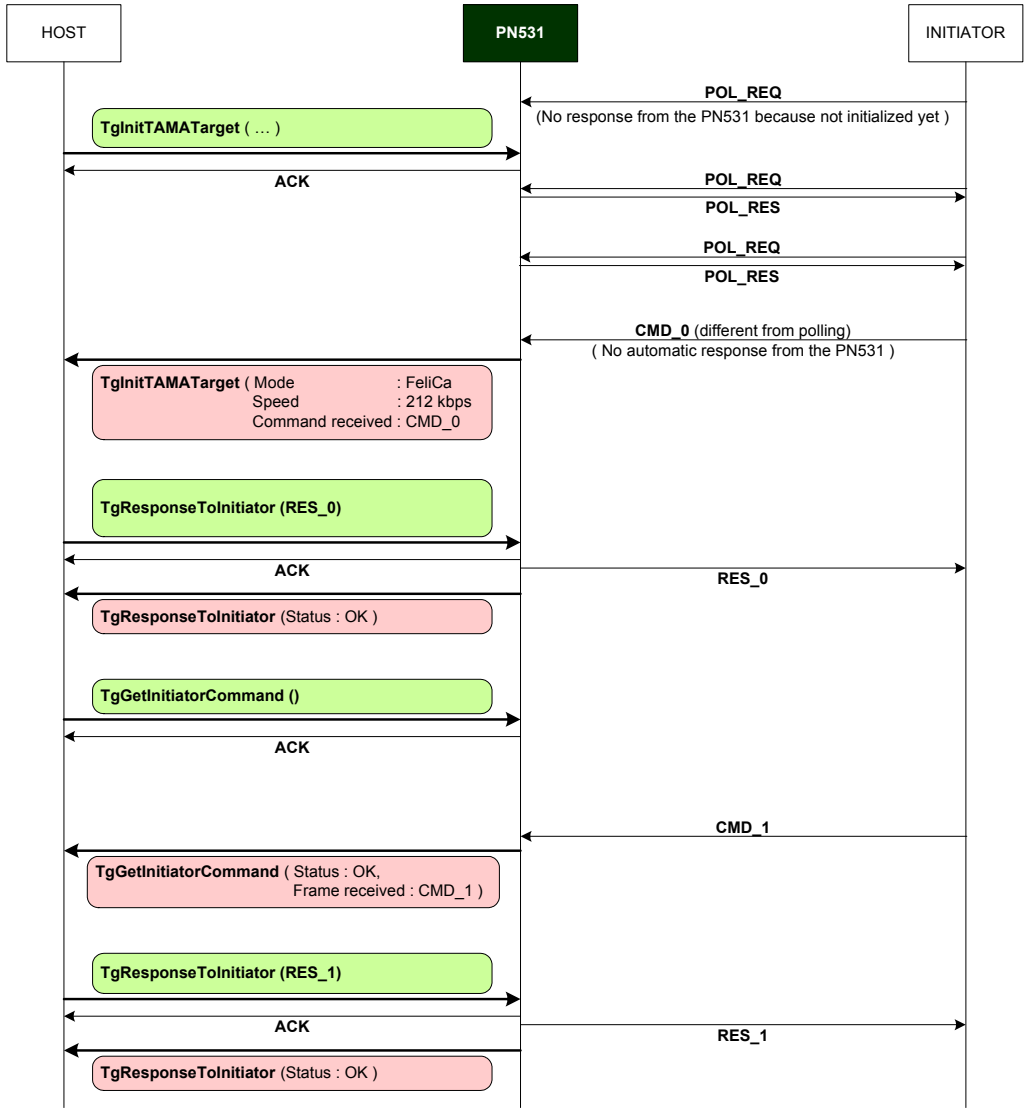


Fig 89. 212 kbps target example

In this example, there are two POL_REQ / POL_RES exchanges during the initialisation of the target.

This is just to show that **`TgInitTAMATarget`** quits only after having received a command frame different from POL_REQ.

4.5.5 Peer to Peer example with two PN531 (passive mode)

This example shows how to make a peer to peer communication in passive mode using two PN531 ICs.

The three commands **`InDataExchange`** at the initiator side and **`TgGetDEPData`** and **`TgSetDEPData`** at the target side allow to build 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 §4.4.5, p.143, where the DEP chaining mechanism is considered.

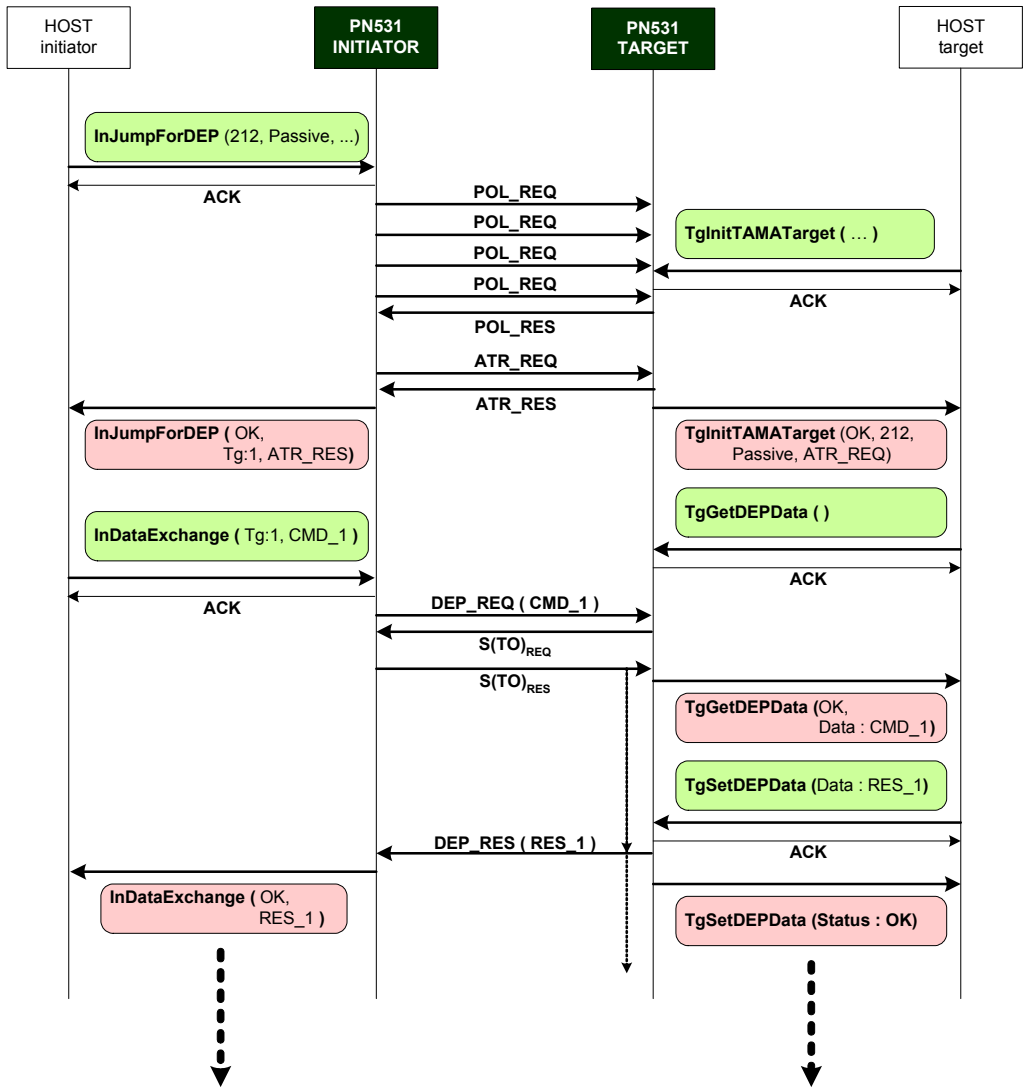


Fig 90. Peer to Peer communication example (passive mode)

4.5.6 Peer to Peer example with two PN531 (active mode)

This example shows how to make a peer-to-peer communication in active mode using two PN531 ICs.

The three commands **`InDataExchange`** at the initiator side and **`TgGetDEPData`** and **`TgSetDEPData`** 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.

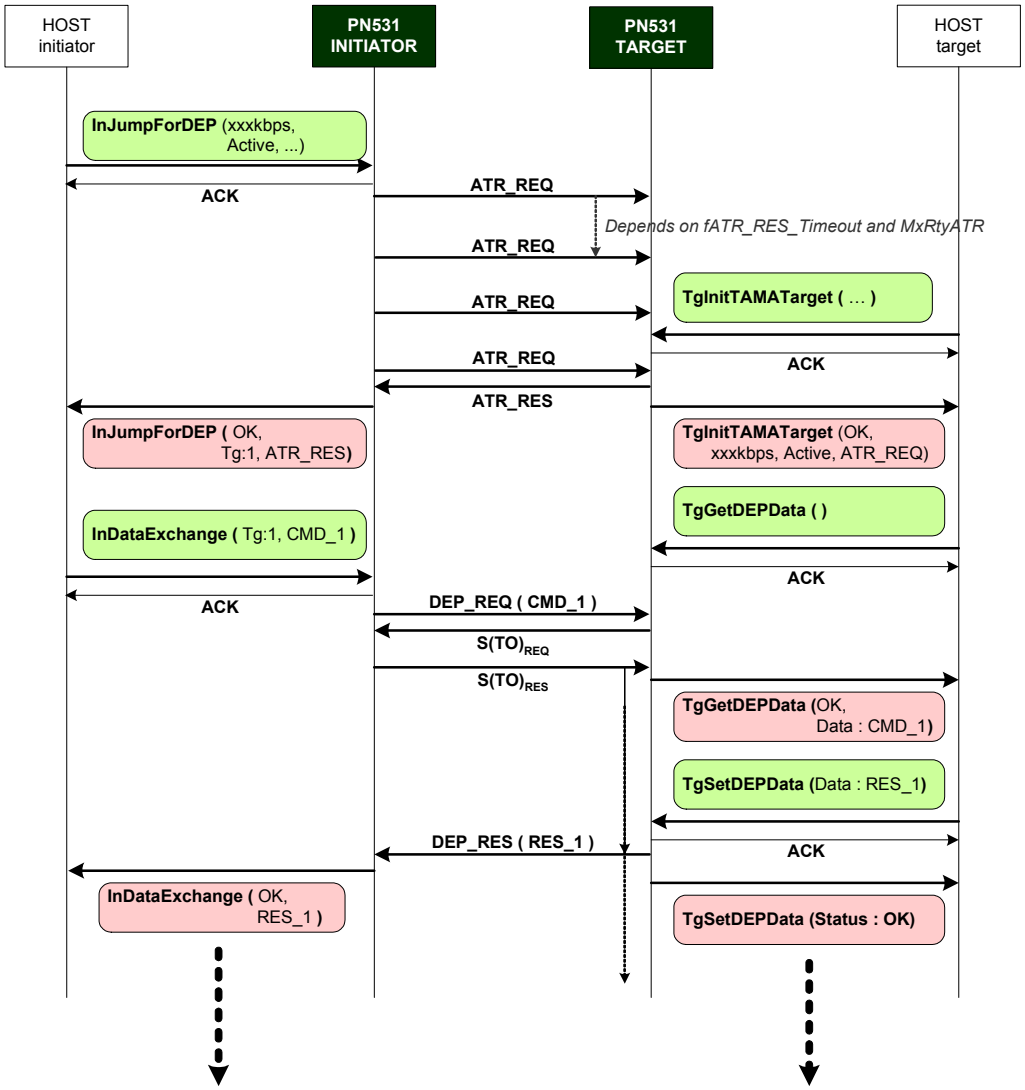


Fig 91. Peer to Peer communication example (active mode)

This example in active mode shows that from the host controller point of view (the one of the PN531 initiator and the one of the PN531 target), the set of commands to use is the same than in passive communication mode.

5. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

6. Licenses

Purchase of Philips I²C components

Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components in the I²C system provided the system conforms to the I²C specification defined by Philips. This specification can be ordered using the code 9398 393 40011.

Purchase of Philips RC5 components

Purchase of Philips RC5 components conveys a license under the Philips RC5 patent to use the components in RC5 system products conforming to the RC5 standard UATM-5000 for allocation of remote control commands defined by Philips.

7. 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.

8. Trademarks

FeliCa™ is a trademark of Sony Corporation

Mifare® is a registered trademark of Koninklijke Philips Electronics N.V.

9. Tables

10. Figures

11. Contents

Date of release:25 November 2005

Document number: UM0501-02

Published in The Netherlands

Date of release:25 November 2005

Document number: UM0501-02

Published in The Netherlands