AN12551

PN5190 design-in recommendations

Rev. 1.1 — 25 October 2022 **Application note**

Document information

Information	Content
Keywords	PN5190, NFC Cockpit, PN5190 Receiver, Contactless Test Station, CTS, TX shaping
Abstract	This document gives additional PN5190 design in recommendations. It provides some tips and tricks to simplify the NFC reader design based on PN5190.

Revision history

Revision history

Rev	Date	Description
1.1	20221025	Package corrected from HVQFN to official naming VFLGA40.
1.0	20220516	First release

1 Introduction

The overall concept of the PN5190 is based on powerful and flexible NFC front-end hardware, which is controlled by the internal µC and its firmware. This firmware functionality can mainly be defined by the "EEPROM" settings. The default settings, as e.g. distributed in the NFC Cockpit package, provide a proper overall functionality, either the optimum for the EMVCo compliance or the optimum for ISO and NFC compliance. Those default settings are optimized for the PNEV5190PB and its 45 mm x 45 mm antenna, so any other antenna and any other use case might require slightly different settings for an optimum performance.

It is essential to understand the function, which belongs to certain settings. The functionality itself is sometimes quite complex, while the complexity is encapsulated inside the FW, and the user settings keep the handling as simple as possible. This document describes some of this functionality and the related settings. It gives recommendations, which settings are more relevant than others (or which ones are even not relevant at all).

NFC Cockpit

It is strongly recommended to use the NFC Cockpit for the analog optimization. The NFC Cockpit provides a GUI and supports all major functions of the PN5190. So without touching a single line of code, the user can configure the FW settings. The NFC Cockpit allows to dump the EEPROM settings into a file, which then allow to distribute those settings into any user's production device. Many configurations can be handled easily with the NFC Cockpit, and the outcome can be used in the user's environment.

The CTS functionality is a good example: The NFC Cockpit can be used to configure the CTS in the GUI, and then the related CTS configuration data can be dumped from the NFC Cockpit and can be used in the user's software, if needed. The retrieved data capture from the user's CTS then can be visualized with the NFC Cockpit.

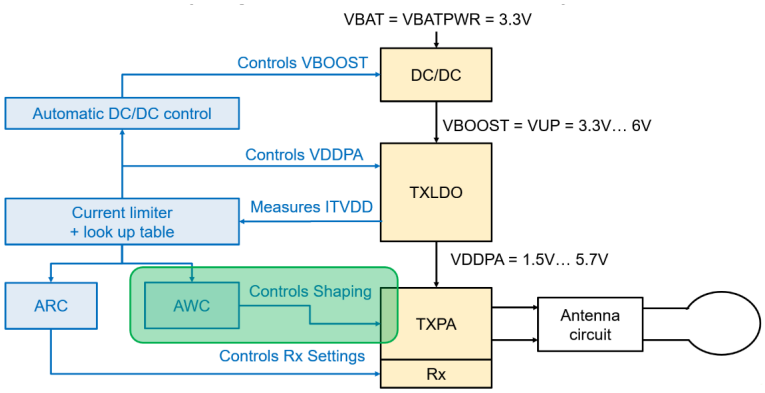
NXP provides a software stack, which can be used to connect any user's hardware (µC driving the user's PN5190) via a virtual com port to the NFC Cockpit. The required µC function for this use case is very limited, and only needs to pass through the data between PN5190 and NFC Cockpit, since the smart functionality is implemented inside the NFC Cockpit.

This document uses the NFC Cockpit and describes many features together with the NFC Cockpit.

Note: *Some important design recommendations regarding the supply and RF layout can be found in [1] for both the VFBGA and VFLGA40 package.*

2 PN5190 transmitter

Normally the transmitter (TX) of the PN5190 is automatically controlled by the firmware. The overall block principle is shown in Figure 1. The shown configuration uses the DPC as per default settings. This includes the control of the TXLDO, the DC-DC converter and also the dynamic register changes for the TXShaping (AWC) as well as the receiver (ARC). The related functionality is described in this section.



1. The most flexible configuration uses the DC-DC and the TXLDO for the DPC.

Figure 1. DPC 2.0 principle

Note: *The automatic control is based on some dependencies, as can be seen from the block principle. These dependencies should be considered during the overall calibration and adjustment of settings. It does not make sense to adjust dynamic receiver settings, as long as the DPC power transfer is not working properly, since a change in the DPC LUTs will then also change the receiver behavior.*

The typical order of adjustments:

1. Check **power transfer at maximum distance**, the unloaded driver current ITVDD and ITVDD with loading at maximum operating distance. Check self.heating under typical and maximum power transfer conditions.
2. Derive and set the **target current**.
3. Calibrate the **DPC current reduction LUT**. Consider the minimum and maximum VDDPA, and RDON feature, if needed.
4. **Check power transfer** (minimum and maximum field strength within the required operating volume).
5. Check the HF Attenuator value to **adjust the RX resistor** (details see [1]).
6. Clear all dynamic TX Shaping setting, and **adjust static TX Shaping** at maximum distance for the relevant protocols.
7. **Adjust dynamic TX Shaping** settings (AWC), if required.
8. Clear all dynamic RX settings (ARC) and check and **adjust the static RX settings** for the relevant protocols.
9. **Adjust dynamic RX settings** (ARC), if required.

2.1 DC-DC and TXLDO

The PN5190 DC-DC boost converter can drive the VUP with up to 6 V. The VUP supplies the TXLDO, which then controls the VDDPA to supply the TX driver.

The TX driver can be supplied directly without using neither the DC-DC nor the TXLDO (VDDPA = VBAT = VBATPWR), as stated in [2] section Transmitter power supply. Not using the TXDO makes the DPC obsolete.

The TX driver can be supplied by the TXLDO output without using the DC-DC, as as stated in [2] section Transmitter power supply.

Not using the DC-DC enables the ULPCD function.

The typical use case is to use the built-in DC-DC and the TXLDO to supply the TX driver: The TXLDO allows to set the output voltage VDDPA (TXLDO_VDDPA_CONFIG register) and to measure the ITVDD (TXLDO_VOUT_CURR register). However, typically the TXLDO is automatically controlled by the PN5190 firmware, using the DPC settings.

In any case it is important to have a stable and clean input supply, i.e. to provide VBAT = VBATPWR, but with **separated supply traces and extra block capacitors** to keep VBAT on the same level as VBATPWR but without any noise or glitches. Refer to [1] for details.

- Typical use case "DPC enabled" (see Section 2.1.3): The firmware controls the DC-DC and TXLDO to ensure the correct output power level, based on the antenna loading.
- Use case "DPC disabled" (see Section 2.1.4): For the DPC adjustment (DPC Calibration), for the Automatic Antenna Tuning or for testing the DPC can be disabled, either permanently via EEPROM or temporarily via register.

Note: *Keep in mind the required power: Driving e.g. 300 mA @ VDDPA = 5.7 V, requires the output power of 6 V · 300 mA = 1.8 W. Using the DC-DC with 3.3 V input, and assuming an efficiency of 85 %, we can estimate the required input power of 2.11 W. With a supply voltage of 3.3 V, we need to provide 640 mA only for the Tx.*

Note: *Keep in mind the self-heating: Driving e.g. 300 mA @VDDPA = 4 V, while the VUP is fixed to 6 V, causes internal losses of (6-4) · 300 mA = 600 mW in the TXLDO alone.*

2.1.1 DC-DC enabled (default)

The DC-DC is enabled via the DCDC_PWR_CONFIG (EEPROM 0x0000).

Typically the DC-DC is used and enabled, and automatically controlled by the PN5190 firmware. There is no specific adjustment required, since the firmware automatically uses the DC-DC in such a way that it provides enough but not too much VUP supply input for the TXLDO and uses the optimum duty cycle to avoid additional noise on the RX.

It is possible to disable the firmware control of the DC-DC, but that is not recommended. An uncontrolled DC-DC with a fixed output voltage level, driving always the maximum required VUP, might overheat the TXLDO, when the DPC switches to a low VDDPA. The same risk of overheating the TXLDO might happen, if the DC-DC is not being used, but the VUP is externally supplied.

Note: *The DC-DC might generate an increased noise floor, if the VBAT is not clean and stable. This might end up in an RX performance decrease at certain VDDPA levels.*

The ULPCD cannot be used, if the DC-DC is enabled.

2.1.2 DC-DC disabled

The DC-DC can be disabled via EEPROM setting in the DCDC_PWR_CONFIG (EEPROM 0x0000).

The VUP supply option setting in the DCDC_PWR_CONFIG must match the used hardware connection to provide a proper function.

Disabling the DC-DC requires a more careful power management consideration, since the TXLDO might be required to cover a bigger voltage drop due to the fix VUP supply.

Note: *Currently the maximum possible voltage drop on the TXLDO is limited to 3.6 V. So even with enabled DPC, the VDDPA does not go below 2.4 V, if the VUP = 6 V.*

2.1.3 DPC enabled (default)

Normally the DPC is enabled and the PN5190 firmware controls the DC-DC and the TXLDO, using the LUTs from the EEPROM. This is the default configuration.

All the default settings of DPC itself are made for the operation of the NXP PN5190 evaluation board PNEV5190BP. Other antennas normally require different settings to achieve an optimum performance. The DPC calibration procedure, which derives those required settings to operate the DPC, is described in detail in [1].

Note: *The DPC calibration procedure requires to disable the DPC, which shall be done via EEPROM, followed by Soft Reset of the PN5190. Otherwise the VDDPA and ITVDD might not show up as expected.*

The DPC allows to use the AWC and ARC features to automatically adapt the waveshaping and receiver settings dynamically. The related settings also need to be carefully adjusted, preferably using the NFC Cockpit.

The DPC can be used with or without DC-DC. Using the DPC without PN5190 DC-DC might overheat the TXLDO, if the TXLDO voltage drop get to large at a high current.

2.1.4 DPC disabled

Permanent setting (via EEPROM)

The DPC can be disabled permanently via EEPROM in the DPC_CONFIG (address 0076h). After power-up or reset, the PN5190 then does not use the automatic firmware function to control the output power. After the RFON, the VDDPA then can be set manually in the TXLDO_VDDPA_CONFIG register. This is e.g. used in the NFC Cockpit DPC calibration tab to allow the DPC calibration.

Disabling the DPC, does not disable the DC-DC control, i.e. setting VDDPA manually then makes the firmware control the DC-DC accordingly. If needed, the DC-DC control can be permanently disabled, too, using the EEPROM PWR_CONFIG (address 0000h).

One reason to disable the DPC is to control the VDDPA manually during the DPC calibration. That is supported by the NFC Cockpit.

A second reason might be a simplified design, especially if even the TXLDO is not being used. Not using the TXLDO automatically makes the DPC obsolete, but saves a few 100 mV at the TX driver supply. This might be relevant for battery powered devices, where e.g. the usage of the ULPCD forbids the DPC anyway. Be aware that not using the DPC requires a different antenna tuning to protect the PN5190 as well as any tag. Disabling the DPC also makes the AWC and ARC obsolete.

After the DPC is permanently being disabled via EEPROM setting, using the DPC_CONFIG (address 0076h), it requires a Soft Reset to enable all related settings.

Note: *This disabling via EEPROM can be done in the NFC Cockpit to calibrate the DPC (in the DPC calibration tab). So here as well, disabling the DPC requires a Soft Reset before continuing the DPC calibration.*

Temporary setting (via register)

Another option is to temporarily disable the DPC. This functionality is available with the PN5190 FW2.3 or higher, using the DPC_CONFIG register. Setting the bit 31 writes the value of bit 0:

- DPC_CONFIG register = x8000 0000 → disables DPC temporarily
- DPC_CONFIG register = x8000 0001 → enables DPC temporarily
- PN5190 reset → enables or disables the DPC according to the EEPROM settings

Note: *The temporarily disabled DPC allows to adjust the VDDPA manually, e.g. for test purposes or to drive the Automatic Antenna Tuning (AAT).*

Note: *The DPC calibration shall not be done with the temporarily disabled DPC via register.*

2.2 Dynamic Power Control (DPC)

The Dynamic Power Control and its correct calibration are described in [1]. The correct calibration of the DPC, i.e. the correct filling of the current reduction LUT is essential for the correct operation of the DPC.

As soon as the DPC is calibrated properly, it controls the VDDPA setting based on the driver current to provide the correct level of power to the contactless card. In addition to the power control, the DPC allows to change wave shaping settings (AWC) as well as receiver settings (ARC) dynamically.

Note: *The DPC including the AWC and ARC is very flexible, but needs to be set properly. Normally, after the DPC (power) calibration is done, it makes sense to disable all the AWC and ARC settings, and adjust the* ***static protocol-related settings first****. The AWC and ARC should only be used, after the static settings had been adjusted, and if those static settings do not work in some test positions (i.e. under specific loading conditions).*

Note: *The best way to adjust the AWC and ARC is to use the NFC Cockpit, disable the DPC, and manually set the required VDDPA (range). Then the AWC and the ARC tab of the NFC Cockpit offers an easy adjustment.*

2.2.1 Adaptive Waveshape Control (AWC)

The DPC allows to control the output power based on antenna detuning and loading. So in combination with a correct antenna tuning and a given loading condition, this control can be used to dynamically adapt the wave shape settings to compensate strong coupling effects.

The DPC of the PN5190 offers a lookup table in the EEPROM: DPC_LOOKUP_TABLE (008Bh-0133h), which allows to define the AWC adjustment **per VDDPA** for

- **Current reduction**: this shall be calibrated as described in [1].
- **Modulation index change**: this signed value from -127 up to +127 is automatically added to the modulation index, as set in the protocol settings, related to the bits 8...15 from register SS_TX1_RMCFG (00016h). The default setting = 0 (i.e. no adaptive change of modulation index)

- Adaptive **change of the falling and rising edge time constant at 100 % AM:** this unsigned value (0...15) is added to the EDGE_STYLE, if the 100 % ASK and EDGE_TYPE = 1,2 or 3 ("FW based shaping, see Section 2.3.1) is chosen in the used protocol.
- Adaptive **change of the falling and rising edge time constant at 10 % AM**: this unsigned value (0...15) is added to the EDGE_STYLE, if the 10 % ASK and EDGE_TYPE = 1,2 or 3 ("FW based shaping, see Section 2.3.1) is chosen in the used protocol.

The best way to adjust these settings offers the NFC Cockpit, as shown in Figure 2.

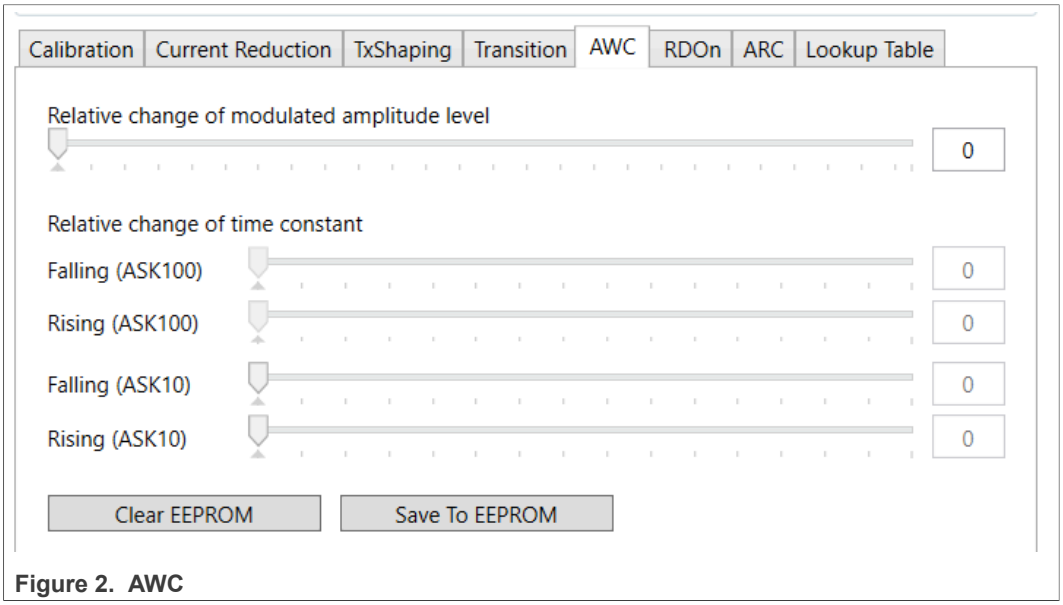


Figure 2. AWC

Note: *This LUT offers 43 32bit entries: one per VDDPA. So the setting, as shown in Figure 2 is only relevant for one VDDPA. Disable the DPC and select the VDDPA manually to fully control the adjustment of these entries.*

Note: *As soon as the EDGE_TYPE = 4, 5 or 6 is chosen ("LUT based Shaping, see Section 2.3.2), the settings for the relative changes of falling and rising time constant are not used.*

2.2.2 Adaptive receiver control (ARC)

The Adaptive Receiver Control (ARC) uses the DPC to dynamically change some receiver settings, i.e. the receiver can be dynamically adapted due to combine a very good sensitivity at low coupling with a better robustness at strong coupling. The control is done due to the DPC, i.e. the TX settings, even though the ARC purely influences the receiver.

Enable / Disable ARC

The ARC can be enabled or disabled globally, using the ARC_CONFIG (0137h) bit 7. The ARC is completely off, if this bit 7 = 0. However, even if this bit 7 = 1, the ARC might still be off, if not enabled in any protocol.

ARC VDDPA range

The ARC allows to define up to 5 different ranges of receiver settings. Each range is defined with two VDDPA values, and applies between these two VDDPA values. The number of range is defined in ARC_CONFIG (0137h) bit 0 … 2.

The VDDPA values are defined in ARC_VDDPA (0139h … 013Dh):

1.5 V … VDDPA_0 - 0.1 V → Range 0

VDDPA_0 … VDDPA_1 -0.1V → Range 1

VDDPA_1 … VDDPA_2 -0.1V → Range 2

VDDPA_2 … VDDPA_3 -0.1V → Range 3

VDDPA_3 … VDDPA_4 → Range 4

VDDPA_0 … VDDPA_4 can be any value between 1.5 V and 5.7 V, as long as

VDDPA_0 < VDDPA_1 < VDDPA_2 < VDDPA_3 < VDDPA_4

ARC per protocol

For every VDDPA range, there is a 16bit ARC entry per protocol:

RM_RX_ARC_0 … RM_RX_ARC_4

For each protocol, the ARC can be enabled or disabled in the bit 14 of the RM_RX_ARC0. The ARC is enabled, if both, bit 14 of RM_RX_ARC_0 = 1 and bit 7 of ARC_CONFIG (0137h) = 1.

ARC parameters

The ARC allows to control three parameters dynamically per VDDPA range per protocol:

1. **IIR filter**: Normally the IIR filter is disabled. Enabling the IIR filter provides a 10 dB signal damping. This can sometimes help to overcome too strong signal in strong coupling situations, especially for the tight ISO/IEC 14443 EMD low-level handling.
2. **MF_GAIN**: The 2-bit MF_GAIN (= RM_MF_GAIN) is normally fixed in the protocol settings, but can be overruled in the ARC dynamically. It might make sense to reduce the MF_GAIN in strong coupling situations to avoid signal clipping effects or meet the EMD levels.
3. **RX Threshold**: The 7-bit DPC_SIGNAL_DETECT_TH_OVR_VALUE (or "RX threshold") typically is set base on the SDT to provide the best RX sensitivity (see Section 3.1.4.1. However, at close coupling it might make sense to increase the RxThreshold (even in combination with reduced MF_GAIN or even enabled 10 dB damping) to increase the robustness or meet the required EMD low levels.

ARC within or outside the FDT

The bit 15 of every RM_RX_ARC_0 defines whether the defined ARC settings apply always or only within the Frame Delay Time (FDT). If the ARC is enabled for a protocol and the ARC settings only apply withing the FDT, the normal protocol settings apply outside the FDT.

The only exception is A106: For A106, there are **two ARC lookup tables** foreseen.
If the ARC is enabled for A106, and the ARC settings shall apply only within FDT (RM_RX_ARC_x bit 15 = 1), the settings as defined in RM_RX_ARC_x apply outside the FDT, while inside the FDT the settings from the ARC_RM_A106_FDT (0C9Dh) are used.

Note: *Be aware that with enabled ARC, the original protocol settings of IIR filter, MF_GAIN, and RX Threshold become obsolete (overwritten). So the ARC settings are absolute values, but no relative changes.*

Note: *The best and least confusing way to adjust the ARC is to use the NFC Cockpit a shown in Figure 3.*

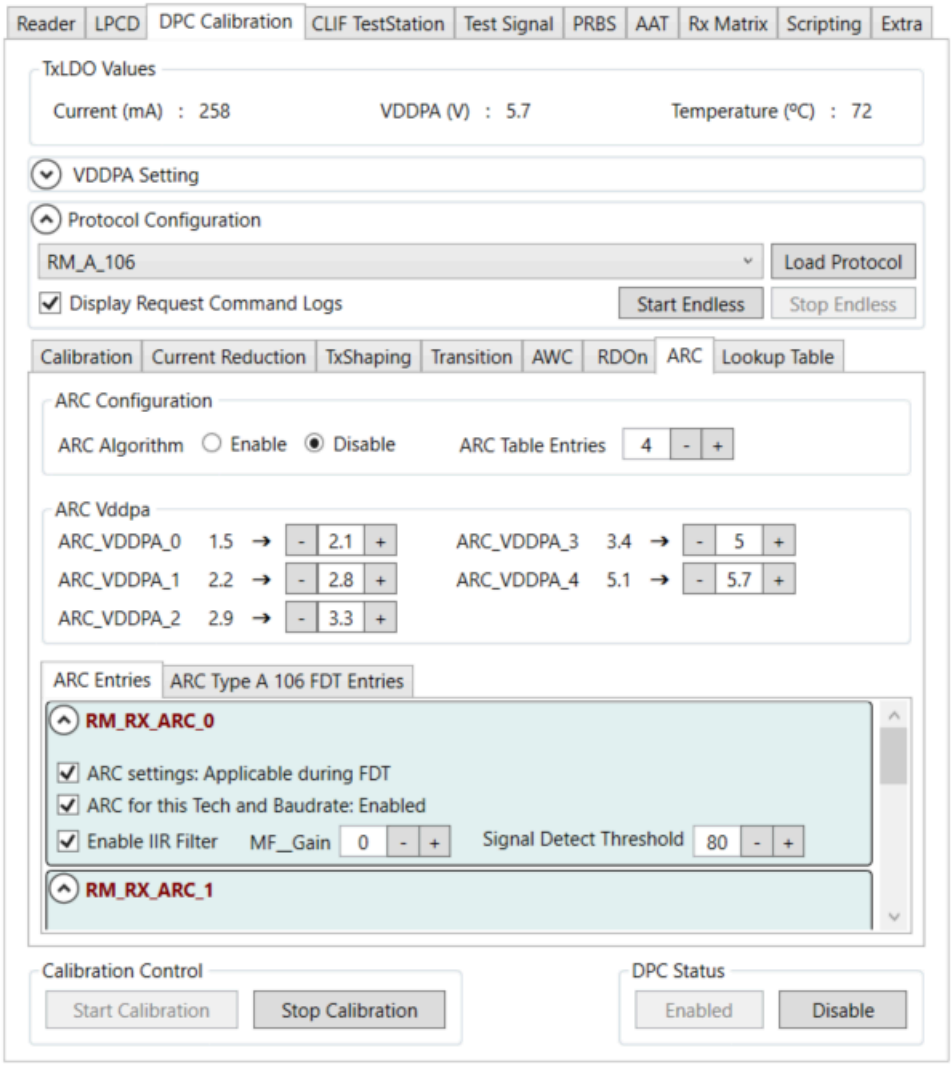


Figure 3. ARC adjustment in NFC Cockpit

2.3 TX shaping

The TX shaping is a specific PN5190 related feature, which allows to smooth the falling and rising edges of transmit modulation pulses. The principle is shown in Figure 4, using a very strong modulation pulse for demonstration purpose.

The example capture of the rising edge shows the additional state: instead of switching from "modulated" state to "unmodulated" state instantly, the PN5190 allows to define a third state, the "transition" state in between. A similar transition is defined for the falling edge (not shown in Figure 4). Both transition states are enabled per default.

The hardware of the PN5190 allows the rising and the falling transition each to have up to 16 carrier cycles, where each carrier cycle can have a different output level - or up to 32 carrier cycles, where every two carrier cycles can have a different output level.

The latter setting with 32 carrier cycles only makes sense, if the modulation pulses are quite long, i.e. for the ISO/IEC 15693 modulation. Normally for all ISO/IEC 14443 pulse shapes the standard setting of up to 16 carrier cycles works best. Whether 16 or 32 carrier cycles are used for a TX shaping, is defined in the bit 7 of the EDGE_LENGTH_protocol for each protocol separately.

The output levels during the 16 cycles of the rising transition are taken from the registers SS_TX1_RTRANS0 … SS_TX1RTRANS3, while the output levels of the falling edge are taken from the registers SS_TX1_FTRANS0 … SS_T1FTRANS3.

Note: *Tx2 is treated in the same way as TX1, if TX2_USE_TX1_CONF (bit 13) in the SS_TX_CONFIG register is set (= default). In that case the settings of SS_TX2_RTRANSx and SS_TX2FTRANSx do no matter.*

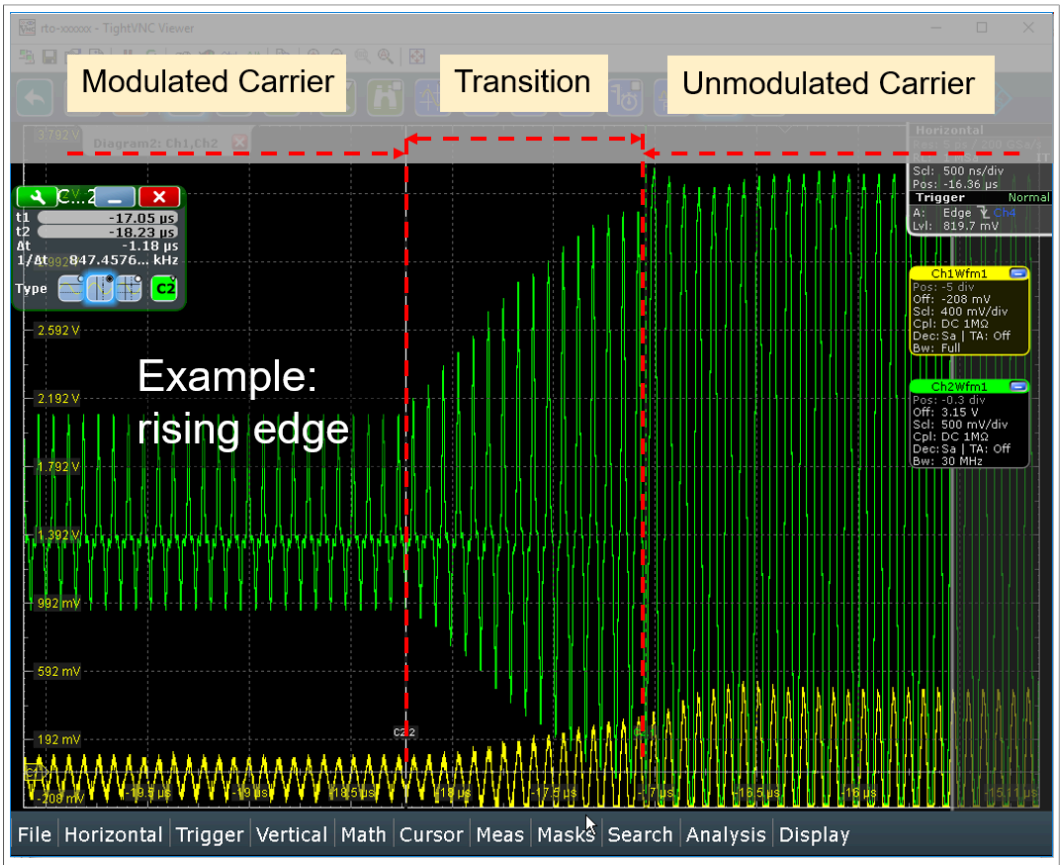


Figure 4. PN5190 TX shaping principle

Normal operation of TX shaping: Normally the PN5190 firmware takes care of the TX shapings, and no user interaction is required. There are two different options to use the normal TX shaping. One option enables a firmware-based transition, the other option is to create a lookup table (LUT) which contains all 16 settings for the transition. The two options and how to adjust the related settings is described in the following.

Adjusting the TX shaping settings: The register settings SS_TX1_RTRANS0 … SS_TX1RTRANS3 and SS_TX1_FTRANS0 … SS_T1FTRANS3 can be modified by the user anytime. The NFC Cockpit provides a transition tab, where all 16 values are read

out of the registers, shown in decimal and hexadecimal, and then can be modified (see Figure 5).

However, there are always firmware functions active to define a certain TX shaping for each protocol, which then even might be set in a way that **the DPC can dynamically change the transition**. So when changing the registers directly (e.g. through the NFC Cockpit transition tab), the DPC settings must be controlled in such a way, that the PN5190 firmware does not overwrite the user values. This might require to temporarily freeze the DPC for such an adjustment.

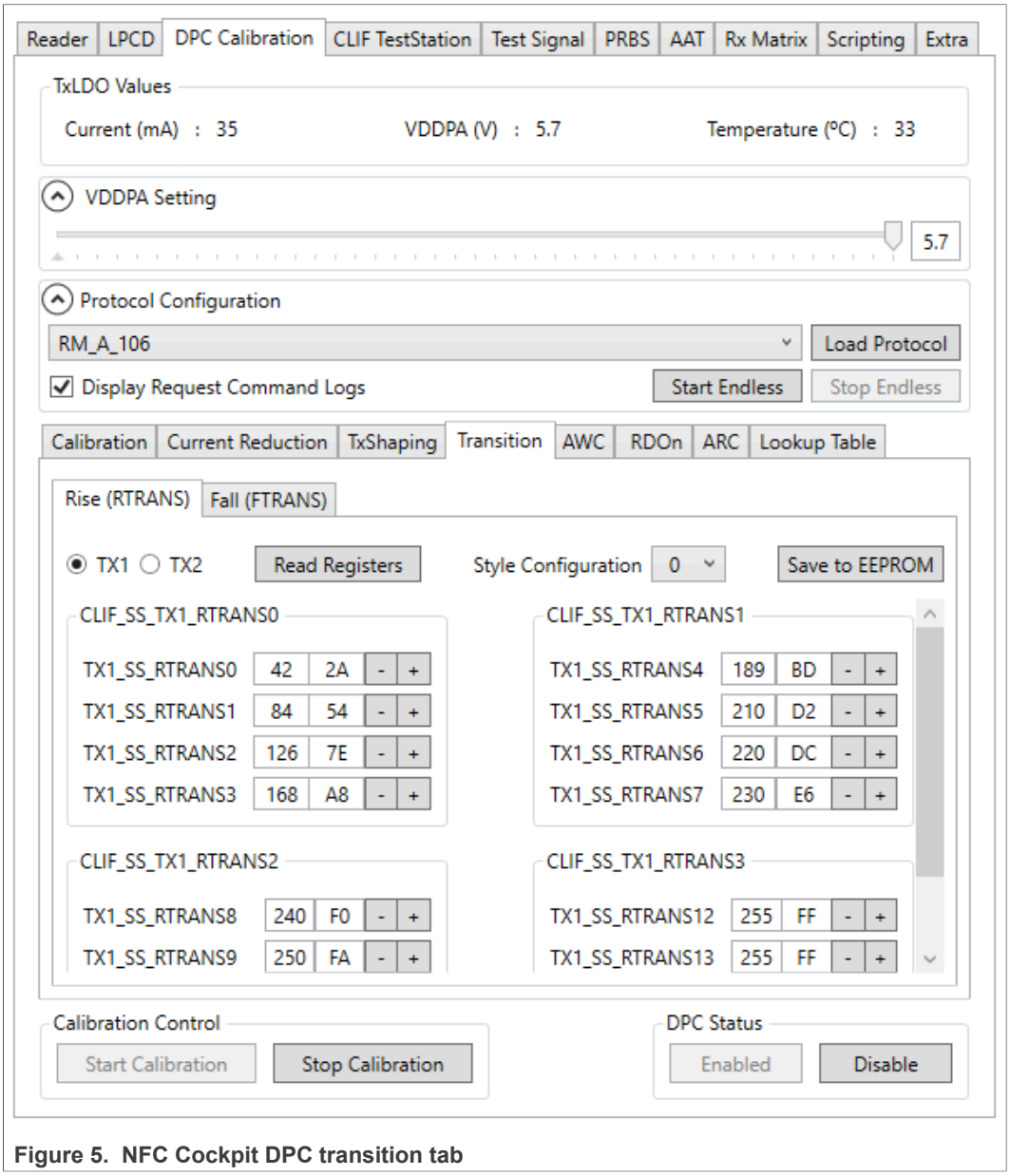


Figure 5. NFC Cockpit DPC transition tab

2.3.1 FW-based TX shaping

To easily handle the up to 16 values for a falling and / or rising transition, the PN5190 firmware offers the FW-based TX shaping. As shown in Figure 6, the user can choose between three different transitions.

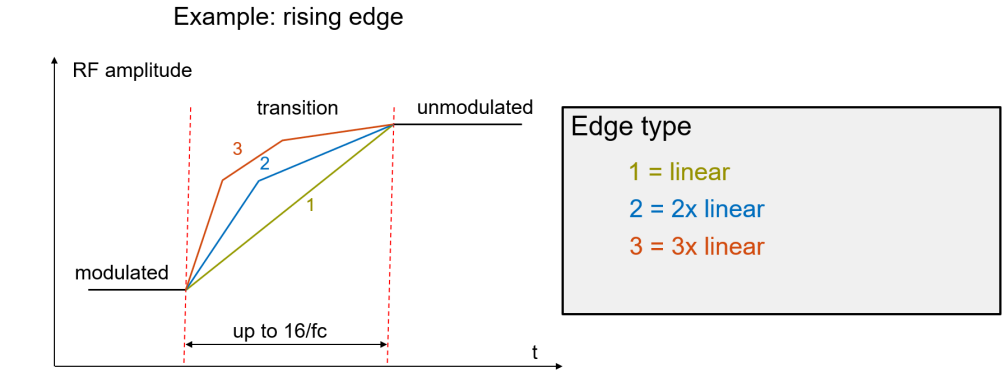


Figure 6. FW-based TX shaping

The EDGE_TYPE_protocol of each protocol defines the used transition function for the rising edge in the least significant four bits (bits 0...3) and for the falling edge in the most significant four bits (bits 4...7).

2.3.1.1 EDGE_TYPE = 1

With EDGE_TYPE = 1, the transition from one level to the other level is done with a linear curve. The corresponding EDGE_STYLE then defines the risetime, respectively the fall time. The Figure 7 shows the linear shape of a falling and a rising transition, using the EDGE_TYPE = 1 and a residual carrier setting of 200 (0xC8). The figure shows the 8 available different fall and rise times, defined by the EDGE_STYLE in the protocol settings. An EDGE_STYLE = 0 creates a fastest transition, an EDGE_STYLE = 7 creates the slowest transition. In the Figure 7, the AWC is not being used, i.e. relative change of the time constant is 0: The setting is purely linked to the used protocol setting.

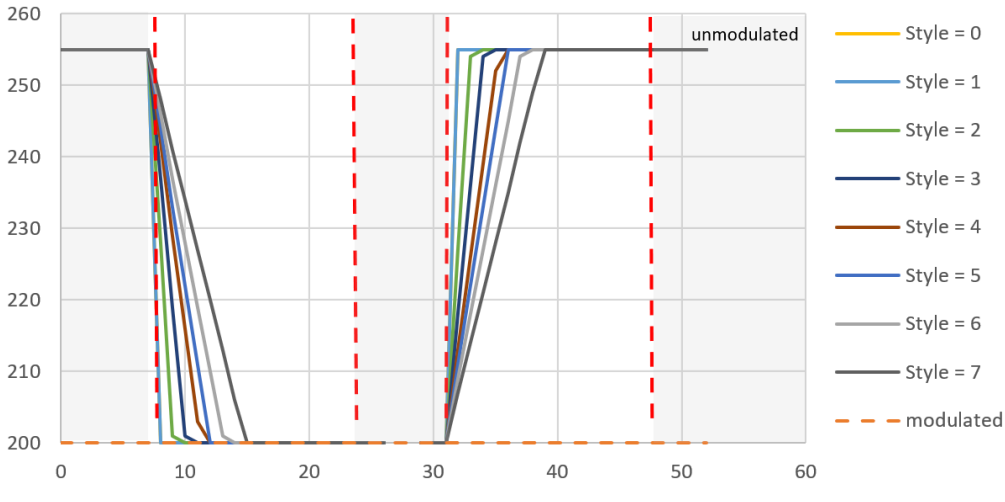


Figure 7. Transition with EDGE_TYPE = 1

The resulting rise (resp fall) time is built out of two settings:

1. The "static" EDGE_STYLE setting in the EDGE_STYLE_protocol (e.g. the EDGE_STYLE_B106 in the EEPROM 0x0034). This allows one setting per protocol.
2. The "dynamic" EDGE_STYLE setting in the DPC_LOOKUP_TABLE entry for either the 100%ASK or 10%ASK per VDDPA ("AWC"). This allows one setting per VDDPA, and so a dynamic shaping change controlled by the DPC.

"Static" protocol setting

The "static" setting allows an EDGE_STYLE setting from 0 to 7, which is defined together with the used protocol (e.g. type B 106).

"Dynamic" AWC setting

The "dynamic" setting defines a relative change, which added to the static setting, using an unsigned 4-bit value per modulation type (10 % ASK in case of B106) and VDDPA setting.

So the combination of both settings allows a full range of 0 … 15. However, as the Figure 8 shows, sometimes higher settings do not make much sense, since the number of transition states is limited to 16. For the 1x linear transition, theoretically the values up to 12 or 13 (same transition) makes sense, but 14 already causes an extra glitch.

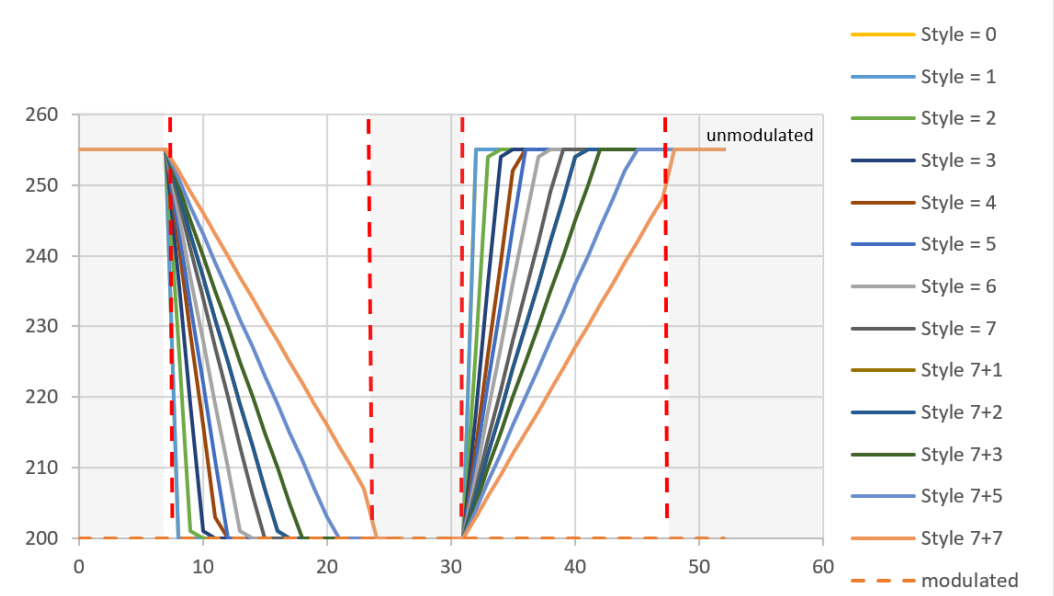


Figure 8. Transition with EDGE_TYPE =1 including AWC

2.3.1.2 EDGE_TYPE = 2

With EDGE_TYPE = 2, the transition from one level to the other level is done with a 2x linear curve, starting with a fast rising or falling, and then switching to slower curve. The corresponding EDGE_STYLE then defines the risetime, respectively the fall time. The Figure 9 shows the 2x linear shape of a rising and falling transition, using the EDGE_TYPE = 2 and a residual carrier setting of 200 (0xC8). The figure shows the 8 available different rise times, defined by the EDGE_STYLE of the protocol. An EDGE_STYLE = 0 creates a fastest transition, an EDGE_STYLE = 7 the slowest transition.

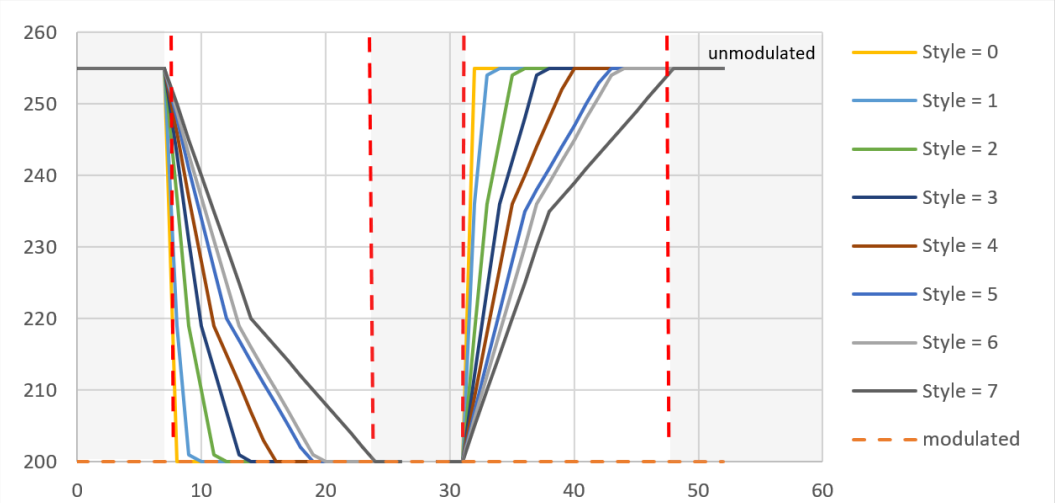


Figure 9. Transition with EDGE_TYPE = 2

The resulting rise (resp fall) time is built out of two settings:

1. The "static" EDGE_STYLE setting in the EDGE_STYLE_protocol (e.g. the EDGE_STYLE_B106 in the EEPROM 0x0034). This allows one setting per protocol.
2. The "dynamic" EDGE_STYLE setting in the DPC_LOOKUP_TABLE entry for either the 100%ASK or 10%ASK per VDDPA ("AWC"). This allows one setting per VDDPA, and so a dynamic shaping change controlled by the DPC.

"Static" protocol setting

The "static" setting allows an EDGE_STYLE setting from 0 to 7, which is defined together with the used protocol (e.g. type B 106). The "dynamic" setting defines a relative change, using an unsigned 4-bit value per modulation type (10 % ASK in case of B106) and VDDPA setting.

So the combination of both settings allows the full range of 0 … 15. However, as the Figure 10 shows, sometimes the higher settings do not make much sense, since the number of transition states is limited to 16. For the 2x linear, theoretically the values up to 7 makes sense, and higher values cause an extra glitch.

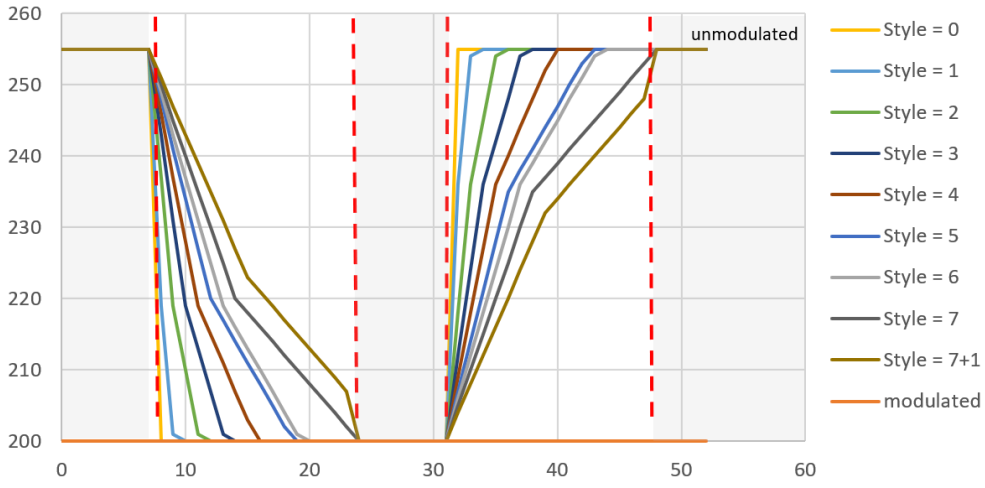


Figure 10. Transition with EDGE_TYPE = 2 including AWC

2.3.1.3 EDGE_TYPE = 3

With EDGE_TYPE = 3, the transition from one level to the other level is done with a 3x linear curve, starting with a fast rising or falling, then switching to a slightly slower curve, and then switching to an even slower curve. The corresponding EDGE_STYLE then defines the risetime, respectively the fall time. The Figure 11 shows the triple linear shape of a rising and falling transition, using the EDGE_TYPE = 3 and a residual carrier setting of 200 (0xC8). The figure shows 7 different rise times, defined by the EDGE_STYLE. An EDGE_STYLE = 0 creates a fastest transition, an EDGE_STYLE >4 is too slow in this example and does not make much sense.

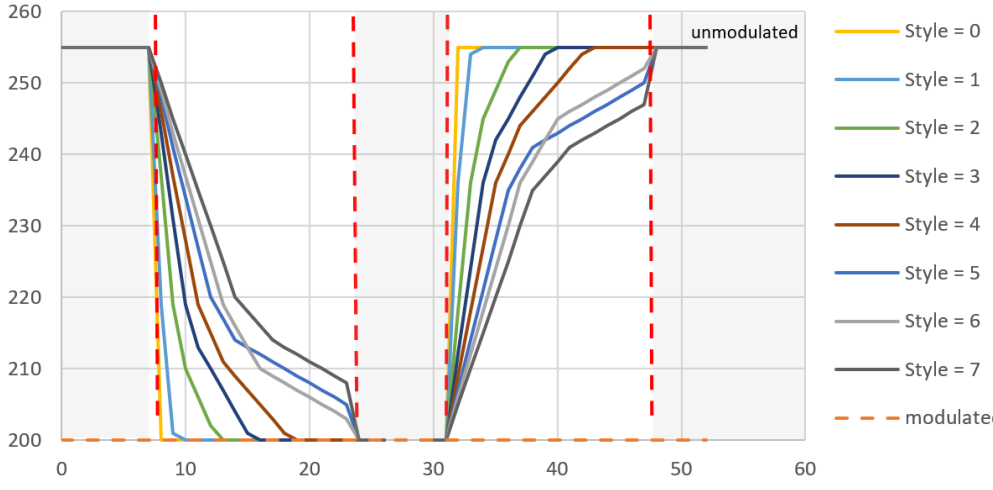


Figure 11. Transition with EDGE_TYPE = 3

The resulting rise (resp fall) time is built out of two settings:

1. The "static" EDGE_STYLE setting in the EDGE_STYLE_protocol (e.g. the EDGE_STYLE_B106 in the EEPROM 0x0034). This allows one setting per protocol.

2. The "dynamic" EDGE_STYLE setting in the DPC_LOOKUP_TABLE entry for either the 100 % ASK or 10 % ASK per VDDPA. This allows one setting per VDDPA, and so a dynamic shaping change controlled by the DPC.

"Static" protocol setting

The "static" setting allows an EDGE_STYLE setting from 0 to 7, which is defined together with the used protocol (e.g. type B 106).

"Dynamic" AWC setting

The "dynamic" setting defines a relative change, using an unsigned 4-bit value per modulation type (10 % ASK in case of B106) and VDDPA setting.

So the combination of both settings allows a full range of 0 … 15, even though values > 4 do not make sense.

2.3.2 Lookup table-based TX shaping

To easily handle the up to 16 values for a falling and / or rising transition, the PN5190 firmware offers a transition function, which allows to manually adjust the transition in free way and then fix the related settings into a lookup table (LUT). This LUT then is being applied with the protocol.

Start values for a LUT

NXP provides an excel sheet NXP-NFC-Reader-PN5190-LUT_based_Shaping_Tool.xlsx in [3], which allows to generate a smooth transition based on an e-function. The Figure 12 shows an example. It is only required to enter the residual carrier setting (as used in the protocol settings) and to choose a time constant ("tau"), which is represented by a number >0. Values > 50 do not make much sense, since then the transition is limited by the 16 available transitions and the generated curve will cause an extra glitch.

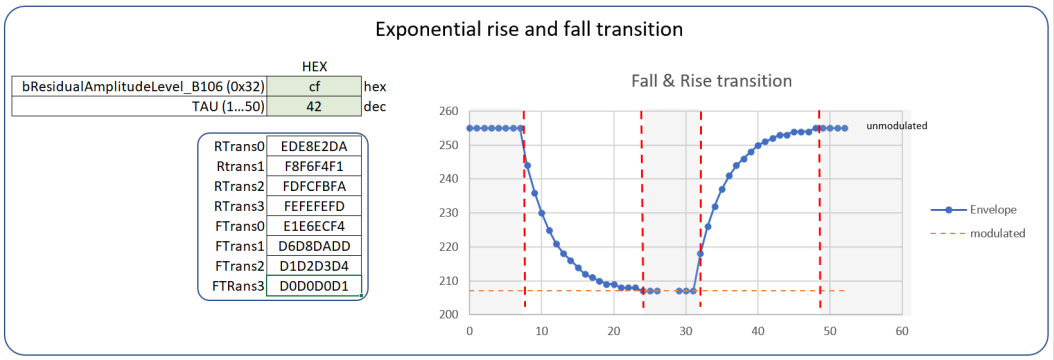


Figure 12. Synthesis of exponential rise and fall time transition

The excel sheet shows the transition and the related hexadecimal register values, which can be directly used to be written ("copy and paste") into the registers (NFC Cockpit) or into one of the LUT areas inside the EEPROM XML, which contains all settings.

Adjusting the transition

The typical way to adjust the transition in the NFC Cockpit manually is this:

1. Disable DPC (via EEPROM), reset the PN5190, and start the DPC calibration.
2. Select the Transition tab, and execute a <Load Protocol> of the desired protocol.
3. Select the correct VDDPA (check current consumption to avoid overloading!).

4. <Start Endless> with or without displaying the related log info: this starts an endless repeat of "Request command" (i.e. the first command of the selected protocol, which might even be an artificial "non-existing" command like a REQA using higher bit rates) to execute the modulation.
5. Select Rise or Fall tab and <Read Registers>. This updates the view of all 16 states in the NFC Cockpit tab from the RTRANS or FTRANS registers.
6. Check the TX Shaping with an external tool (Oscilloscope, Wave Checker, ISO or EMVCo test tool with TestPICC).
7. Adjust any of the transition states manually, until the overall shaping fits.

When all transition values are OK, the transition can be saved into a LUT: Select a Style-Configuration from 0 to 3 to save the transition values into the related LUT.

EDGE_STYLE setting for the LUT based shaping

As soon as the TX shaping uses the LUT base shaping (i.e. EDGE_TYPE = 4, 5 or 6), the EDGE_STYLE can be 0 … 3, which chooses one out of 4 available LUT tables.

The chosen LUT can have different settings for rising and falling edges. For the same modulation, different shaping methods or LUTs can be used for the falling and rising edges.

2.3.2.1 EDGE_TYPE = 4

The EDGE_TYPE = 4 chooses the LUT based shaping, i.e. the related EDGE_STYLE can be 0 … 3, which then refers to the used LUT. This LUT is loaded into the shaping registers with the corresponding "load protocol". The register values are not changed dynamically by the DPC.

Note: *The EDGE_TYPE = 4 normally causes problems, if the modulation index is changed. So even with an AWC setting, which changes the modulation index, the LUT is not changed, i.e. the "static" transition of the LUT can cause glitches due to the changed residual carrier. So normally the setting EDGE_TYPE = 4 is only used for debugging purposes, where the user wants to "freeze" the LUT on purpose.*

2.3.2.2 EDGE_TYPE = 6

The EDGE_TYPE = 6 chooses the LUT based shaping, i.e. the related EDGE_STYLE can be 0 … 3, which then refers to the used LUT. This LUT is loaded into the shaping registers with the corresponding "load protocol". The register values are automatically adapted by the DPC, as soon as the residual carrier changes due to the AWC setting.

Note: *The EDGE_TYPE = 6 is automatically adapted based on AWC changes of the residual carrier. So a change of the residual carrier level in the AWC is considered by the DPC, and no extra glitch is caused. Therefore the setting EDGE_TYPE = 6 typically makes more sense than 4 in real applications.*

Note: *Be aware, that a manual change of the static residual carrier setting in the protocol does not affect the TX shaping LUT. This can cause problems, therefore it is recommended to define and adjust the LUT only, after the residual carrier level has been fixed.*

2.3.2.3 EDGE_TYPE = 5

The EDGE_TYPE = 5 chooses the LUT based shaping, i.e. the related EDGE_STYLE can be 0 … 3, which then refers to the used LUT. This LUT is loaded into the shaping

registers with the corresponding "load protocol". The register values are automatically adapted by the DPC, as soon as the residual carrier changes due to the AWC setting.

This behavior is basically the same as with EDGE_TYPE = 6, but on top of that, the user can define an additional correction of the LUT-based transition in the CORRECTION_ENTRY_TABLE (0BDAh).

CORRECTION_ENTRY_TABLE

This LUT contains 43 entries, one for each VDDPA. These entries are linked to the DPC and define a shifting of the "residual carrier level", which is used to adapt the original transition LUT, with +/-127. This correction, however, only makes sense for the rising edge.

The Figure 13 shows an example.

The light blue curve shows the original LUT, as stored in the RTRANS_FTRANS_TABLE (0C03h).

The AWC in this case shifts the residual carrier from 0xC8 (200 decimal) to 0xDC (220 decimal). The falling edge uses the EDGE_TYPE = 6, so the AWC change of the residual carrier automatically adapts the falling transition.

The rising edge uses the EDGE_TYPE = 5, with a + 20 correction in the related sCorrection_EntryX (brown curve) or a -12 correction in the related sCorrection_EntryX (dark blue curve). So in both example cases, the rising edge is not only adapted due to the AWC change of the residual carrier, but also "corrected" based on an entry in CORRECTION_ENTRY_TABLE (0BDAh), which corresponds to the related VDDPA step.

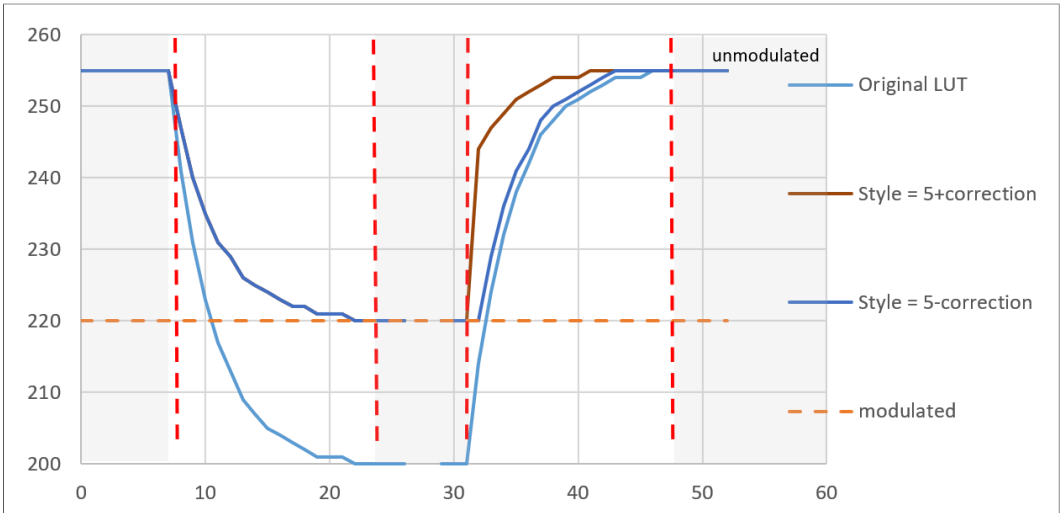


Figure 13. LUT based shaping with EDGE_TYPE = 5

3 PN5190 receiver

The PN5190 receiver (RX) is optimized to provide a good RX sensitivity combined with a strong robustness against noise and interference. For many use cases, the provided default settings in combination with the PN5190 FW functionality already cover the needs.

However, the relevant control settings are available for fine-tuning. The NFC Cockpit provides support functions to optimize some of those settings, if needed. The build in Contactless Test Station (CTS) provides an easy access to capture the available debug signals.

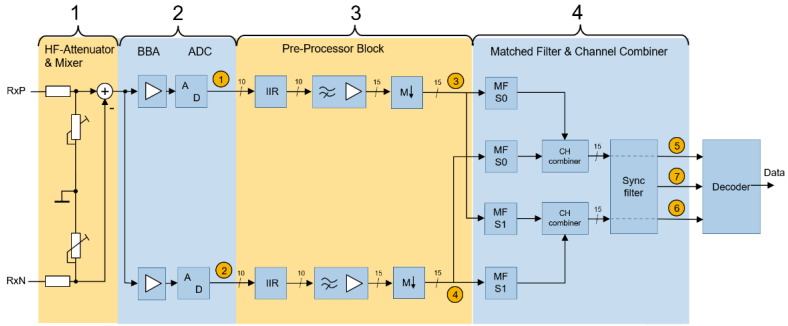
The receiver settings are stored either in the protocol area of the EEPROM, or in the DPC-related lookup tables (ARC). Refer to Section 2.2.2 for details.

3.1 RX block diagram

Figure 14 shows the PN5190 RX block diagram, indicating the four major blocks:

1. HF-attenuator and mixer
2. Baseband amplifier (BBA) and analog-digital converter (ADC)
3. Pre-processor block
4. Matched filter and channel combiner

. Each of the four blocks is described in the following sections.



1. Numbers 1…7 in orange circles indicate the available test signal: see Table 1.

Figure 14. PN5190 receiver block diagram

3.1.1 HF attenuator and mixer

RxP and RxN need to be connected to the antenna circuitry via a serial capacitor (for the DC decoupling) and a serial resistor (for the voltage level reduction). The adjustment of that external resistor is described in [1].

In combination with that external resistor, the HF attenuator block controls the RX input voltage level in such way that the RX always gets a proper input signal. This control loop is done automatically, and there is no direct user setting required.

Note:

The HF Attenuator and its control loop used to be called "AGC" in the PN5180 / PN7462 RX concept. However, the HF Attenuator of the PN5190 is not used to control the Dynamic Power Control (DPC): this is different than the PN5180 / PN7462 DPC concept.

Reading the HF_ATT_VAL (RXCTRL_STATUS register) retrieves a 6-bit value, which corresponds to the RX carrier input level, and therefore can be used to optimize some settings and adjustments.

3.1.1.1 Optimum RX connection

In the beginning of the RX optimization, it is required to read the HF_ATT_VAL to ensure a proper RX connection. The HF_ATT_VAL is a 6-bit value: The value in unloaded reader mode condition, which corresponds to “full power”, shall be ≈ 40 dec. This value then guarantees or proper control over the full DPC without exceeding the value range. If the read value is different, then the external R must be adjusted (see [1]).

3.1.1.2 Input value for the ULPCD

The mixer provides the I and Q channel base band signal.

3.1.2 BBA and ADC

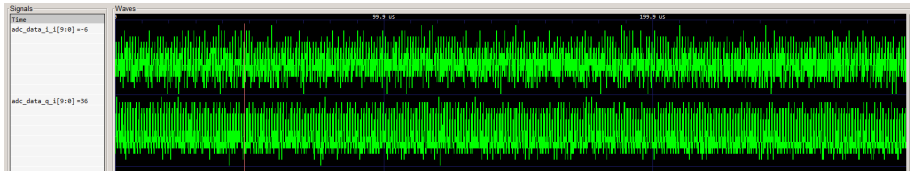
The I and Q channel signals are amplified in the base band amplifier (BBA) and converted into a digital signal.

3.1.2.1 ADC_I and ADC_Q

This digital signal is available for both I and Q channel as test signal 1 and 2 (see Table 1) with a 10-bit resolution and a maximum sampling frequency of 13.56 MHz, either at one of the AUX test pins or as an internal capture with the CTS.

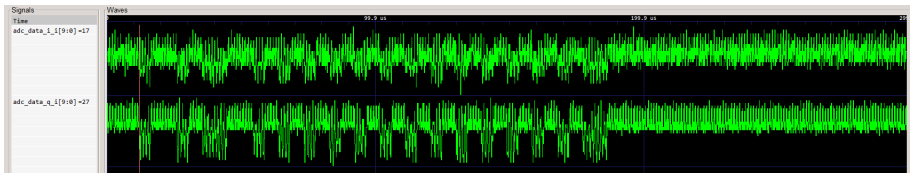
These two signals (ADC_I and ADC_Q) can be taken to judge the overall RX signal quality: Any system noise coming into the RX, which might impact the receiver performance, can be seen here in the I and Q signal.

The following three figures show a type A capture with no card response (Figure 15: only noise floor), a weak ATQA (Figure 16: card at maximum reading distance still with 100 % reception) and a good ATQA (Figure 17: card in the middle of the reading volume).



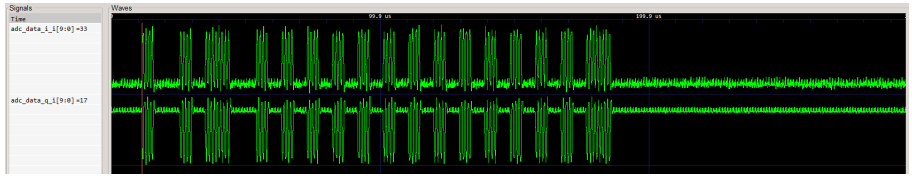
1. Trigger = falling edge of TxActive, Post capture with 60 µs delay
2. Sample rate fs = 6.78 MHz
3. No card response

Figure 15. CTS example with ADC_I and ADC_Q, no card response



1. Trigger = falling edge of TxActive, Post capture with 60 µs delay
2. Sample rate fs = 6.78 MHz
3. Card response = ATQA with weak coupling
4. Marker shows the begin of SOF

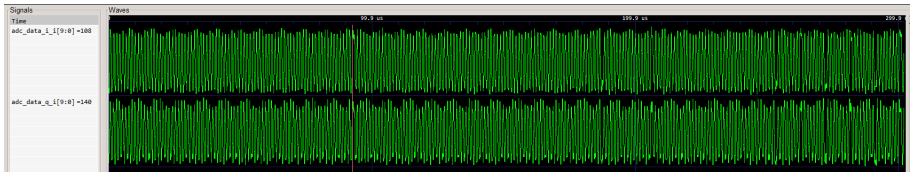
Figure 16. CTS example with ADC_I and ADC_Q, weak ATQA



1. Trigger = falling edge of TxActive, Post capture with 60 µs delay
2. Sample rate fs = 6.78 MHz
3. Card response = ATQA with good coupling
4. Marker shows the begin of SOF

Figure 17. CTS example with ADC_I and ADC_Q, good ATQA

The Figure 18 shows the same scenario with a clean ATQB. The marker shows the start of the SOF.



1. Trigger = falling edge of TxActive, Post capture with 200 µs delay
2. Sample rate fs = 6.78 MHz
3. Card response = ATQB with good coupling
4. Marker shows the begin of the SOF

Figure 18. CTS example with ADC_I and ADC_Q, good ATQB

3.1.2.2 Base band amplifier gain

The base band amplifier (BBA) can be set in the DGRM_BBA register (0x2D), using a minimum and a maximum gain setting as well as an initial gain setting.

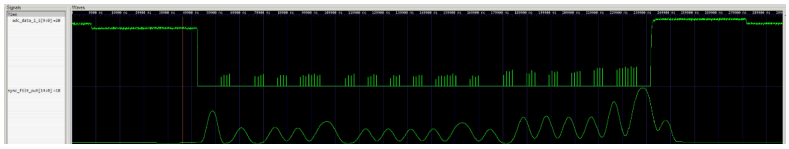
The recommendation is to not change the default settings, as provided in the default Load Protocol settings, except for type A 106 (ASK).

For the optimum, **ASK demodulation (type A 106) the BBA gain shall be fix**, i.e. the three related settings in the DGRM_BBA register shall be set to the same value:

DGRM_BBA_MIN_VAL = DGRM_BBA_MAX_VAL = DGRM_BBA_INIT_VAL

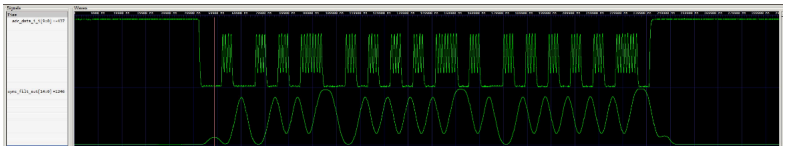
For a sensitive RX, the BBA gain might be set to its maximum value = 4. However, this might result in some negative clipping effects, as shown in Figure 19. Such clipping might cause a wrong RX trigger due to a weird load change, as applied by EMVCo L1 test equipment.

Reducing the BBA gain to a value = 2, fixes this clipping issue and provides a correct RX data decoding as shown in Figure 20.



1. BBA gain = 4

Figure 19. I channel signal clipping

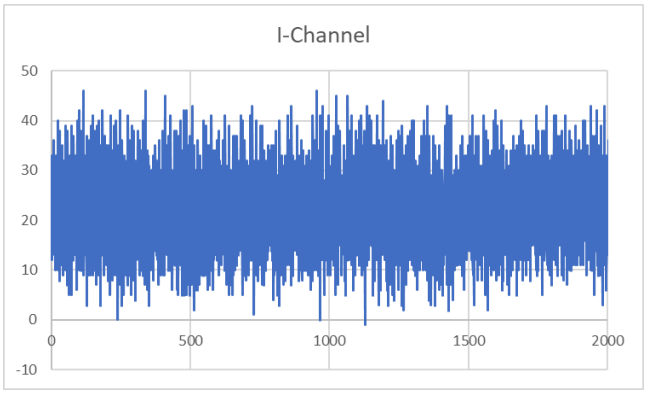


1. BBA gain = 2

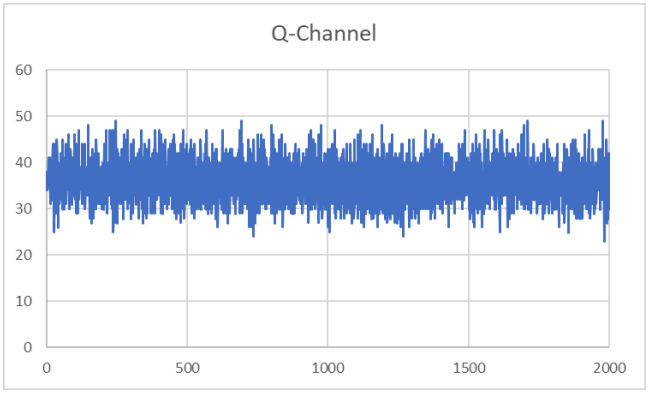
Figure 20. I channel signal not clipping

3.1.2.3 CTS data post processing

The Figure 21 shows a capture of a clean ADC_I and ADC_Q signal example in time domain. The signal captures have been copied into MS Excel for further post processing. It can be seen that there is some noise floor of up to 40..50 digits in Q channel and around 25 digits in Q channel.



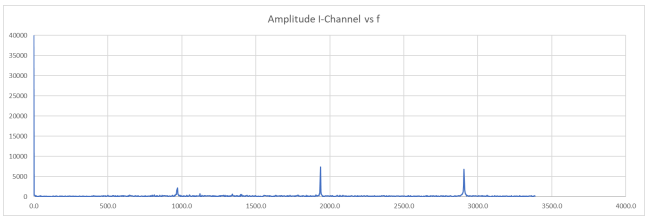
I channel signal of clean design



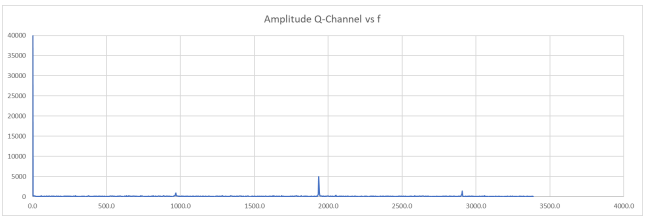
Q channel signal of clean design

Figure 21. I and Q channel time domain example (clean)

The Figure 22 shows the same signals in frequency domain. It can be seen that the noise floor is a specific frequency including harmonics. In this case, this frequency is related to the internal DC-DC converter, which is synchronized with the RX in such a way that this remaining noise level does not impact the RX performance.



I channel signal of clean design

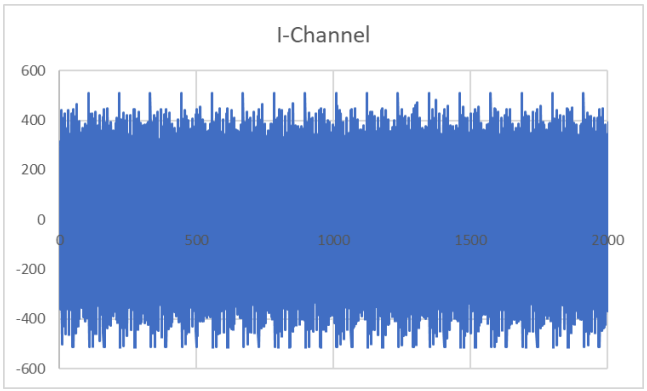


Q channel signal of clean design

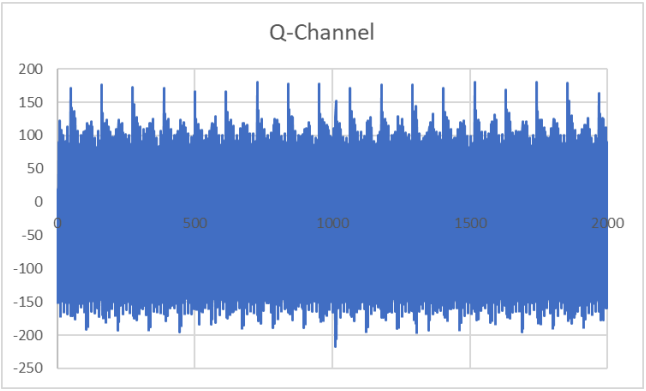
Figure 22. I and Q channel frequency domain example (clean)

The same type of signals is shown in Figure 23 and Figure 24, but in this case, there is additional noise coupled into the system. This noise (in this example) has a much higher amplitude than the card response signal itself.

The time domain shows +/-500 on the I channel signal and +/- 200 on the Q channel. The frequency domain indicates a peak at 1.56 MHz, but with a huge number of spurious. Such kind of noise with such high level can impact the RX performance, i.e. it might reduce the RX sensitivity.

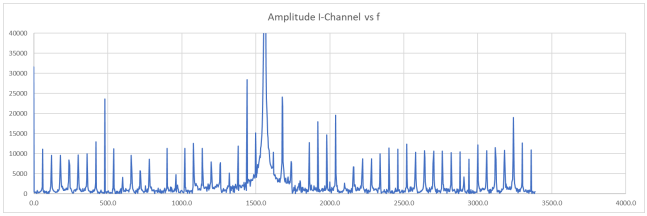


I channel signal of a noisy design

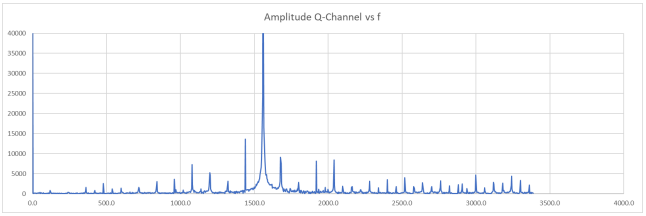


Q channel signal of a noisy design

Figure 23. I and Q channel time domain example (noisy)



I channel signal of a noisy design



Q channel signal of a noisy design

Figure 24. I and Q channel frequency domain example (noisy)

Note:

The CTS implementation and the support of the NFC Cockpit allows a single capture to be taken and shown in time domain. For the deeper analysis, the NFC Cockpit saves the capture, and then a post processing can be done in other SW (e.g. Math lab or MS Excel).

3.1.3 Pre-processor block

The pre-processor block does the processing of the I and Q channel signal, providing an optional IIR filter (not used per default), the base band amplifier (BBA) and the down sampling.

The Infinite impulse response filter (IIR filter) of the PN5190 RX is a 2nd order feedback and 3rd order direct path filter. The pre-programmed coefficients provide this filter to be an all pass filter with ≈ 3 dB damping. This damping can be enabled with the bit 0 of SIGPRO_IIR_CONFIG0 register (address 002Ah).

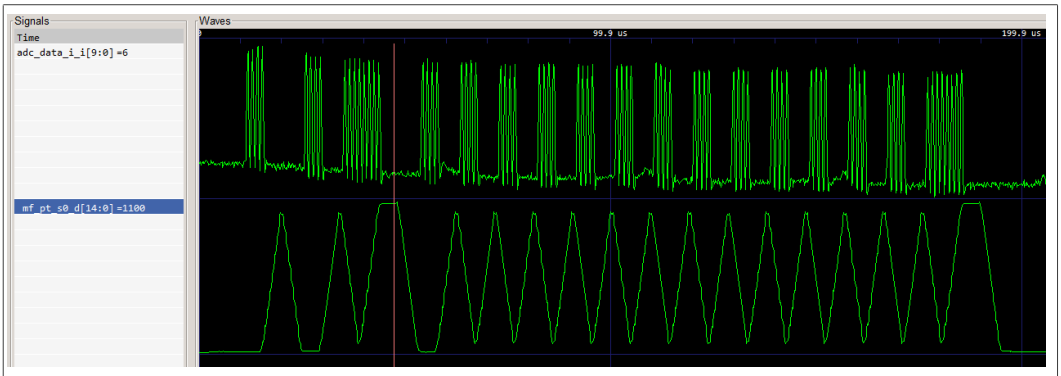
Note: *The IIR filter settings are part of the protocol settings as well as the ARC. The ARC overrules the protocol settings.*

The Pre-processor block further contains a decimation to reduce the sample rate depending on the used protocol. The output signal is 15 bit signed. There is no further adjustment available in this block.

3.1.4 Matched filter block

The matched filter block contains two matched filters, an S0 and S1 filter, for each I and Q channel. The matched filter S0 detects the subcarrier and is used for the ASK (e.g. ISO/IEC 14443 type A 106) as well as the BPSK (e.g. ISO/IEC 14443 type B 106) decoding. The matched filter S1 detects phase shifts in the subcarrier and is used for the BPSK only.

Both S0 and S1 outputs for I and Q channel are then combined in the channel combiner. The combined S0 and S1 output signals can be captured with the CTS, as shown in Figure 25 and Figure 26.

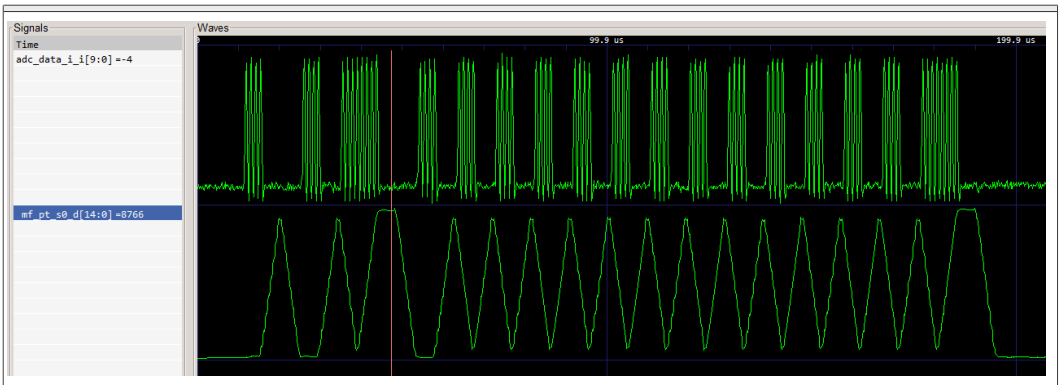


1. Top signal = ADC_I
2. Bottom signal = Combined matched filter output S0
3. Matched Filter Gain = 0, i.e. the marker of S0 indicates the value of ≈ 1000.

Figure 25. CTS example with ADC_I and S0 with MF_Gain = 0, ATQA

The matched filter gain (MF_Gain) can be adjusted from 0 to 3 in bit 15 and 16 of SIGPRO_RM_TECH register (address 0022h).

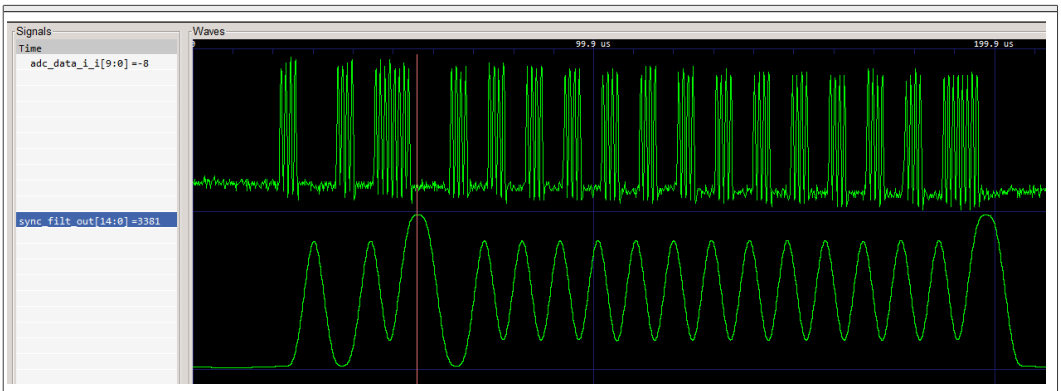
Note: *The MF_Gain is part of the protocol settings and the ARC. The ARC overrules the protocol settings.*



1. Top signal = ADC_I
2. Bottom signal = Combined matched filter output S0
3. Matched Filter Gain = 3, i.e. the marker of S0 indicates the value of ≈ 8000.

Figure 26. CTS example with ADC_I and S0 with ADC_I and S0 with MF_Gain = 3, ATQA

The combined signal of S0 (and S1 at BPSK) passes an averaging, before entering the bit grid synchronization and decoding block. The averaged sync filter output signal can be captured in the CTS, as shown in Figure 27.



1. Top signal = ADC_I
2. Bottom signal = Sync filter output (Sout)
3. Matched filter gain = 2

Figure 27. CTS example with ADC_I and Sout, DTQA

The following figures show the CTS examples with AC_I, S0, S1 and Sout for a clean ISO/IEC 14443 type B response.

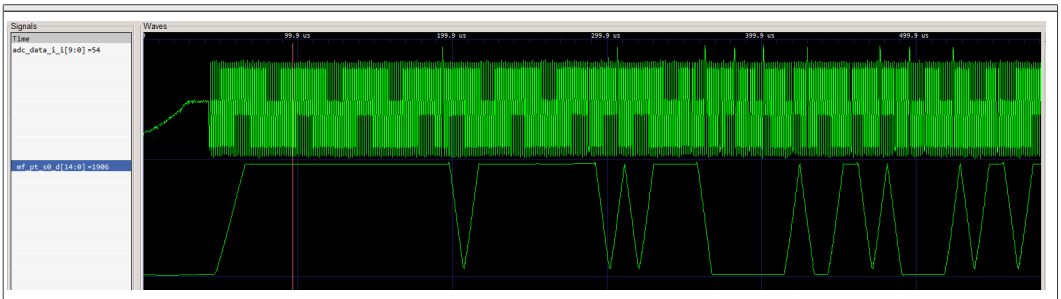


Figure 28. CTS example with ATQB, ADC_I and S0

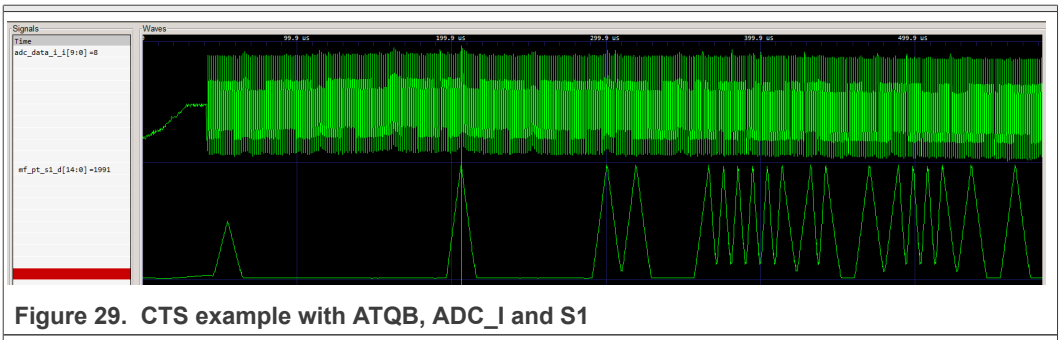


Figure 29. CTS example with ATQB, ADC_I and S1

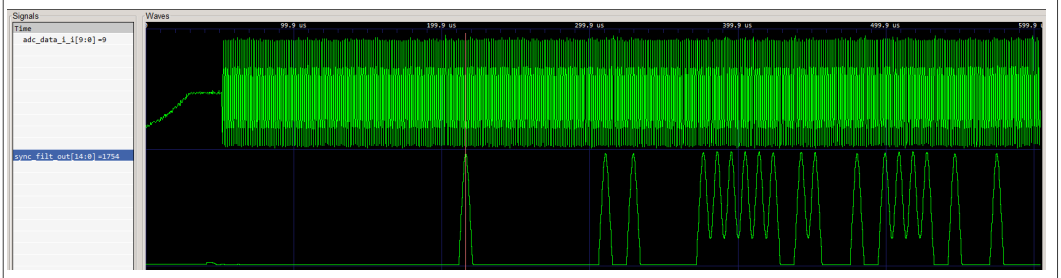


Figure 30. CTS example with ATQB, ADC_I and Sout

The only adjustment within this block is the RxThreshold (= Signal detection threshold = DGRM_SIGNAL_DETECT_TH_ OVR_VAL). This level is a 7-bit value, which is applied in the DGRM_RSSI register (address 0030h). This value is part of the protocol settings as well as the ARC. The ARC overrules the protocol settings.

3.1.4.1 Signal Detection Threshold

The major RX settings adjustment, especially for the ASK decoding (e.g. ISO/IEC 14443 type A 106), is the RxThreshold (signal detection threshold = SDT). This SDT is set in the DGRM_RSSI register (address 0030h) bit 0..6, and therefore is part of the protocol settings as well as the ARC. It applies at the Sout signal. Every Sout signal, which exceeds the SDT, triggers the RX decoder.

So the SDT must always be set above the noise level, otherwise the noise floor will trigger the RX decoder, which leads to wrong decoding (collision error, integrity error, incomplete byte error, …). If the SDT is set too high, the RX looses sensitivity. Especially for the AKS decoding this value must be set properly.

The NFC Cockpit provides an easy to use SDT functionality, which allows to determine a good SDT value. It requires certain settings, as shown in Figure 31:

- Sample rate = 6.78 MHz
- Capture size = maximum (8192 bytes)
- Trigger = Tx_Active, falling edge, post trigger with ≈ 90 µs delay on the first occurrence
- Test bus = obs_clif_sigpro_rm9 and obs_clif_sigpro_rm8 = Sync filter output

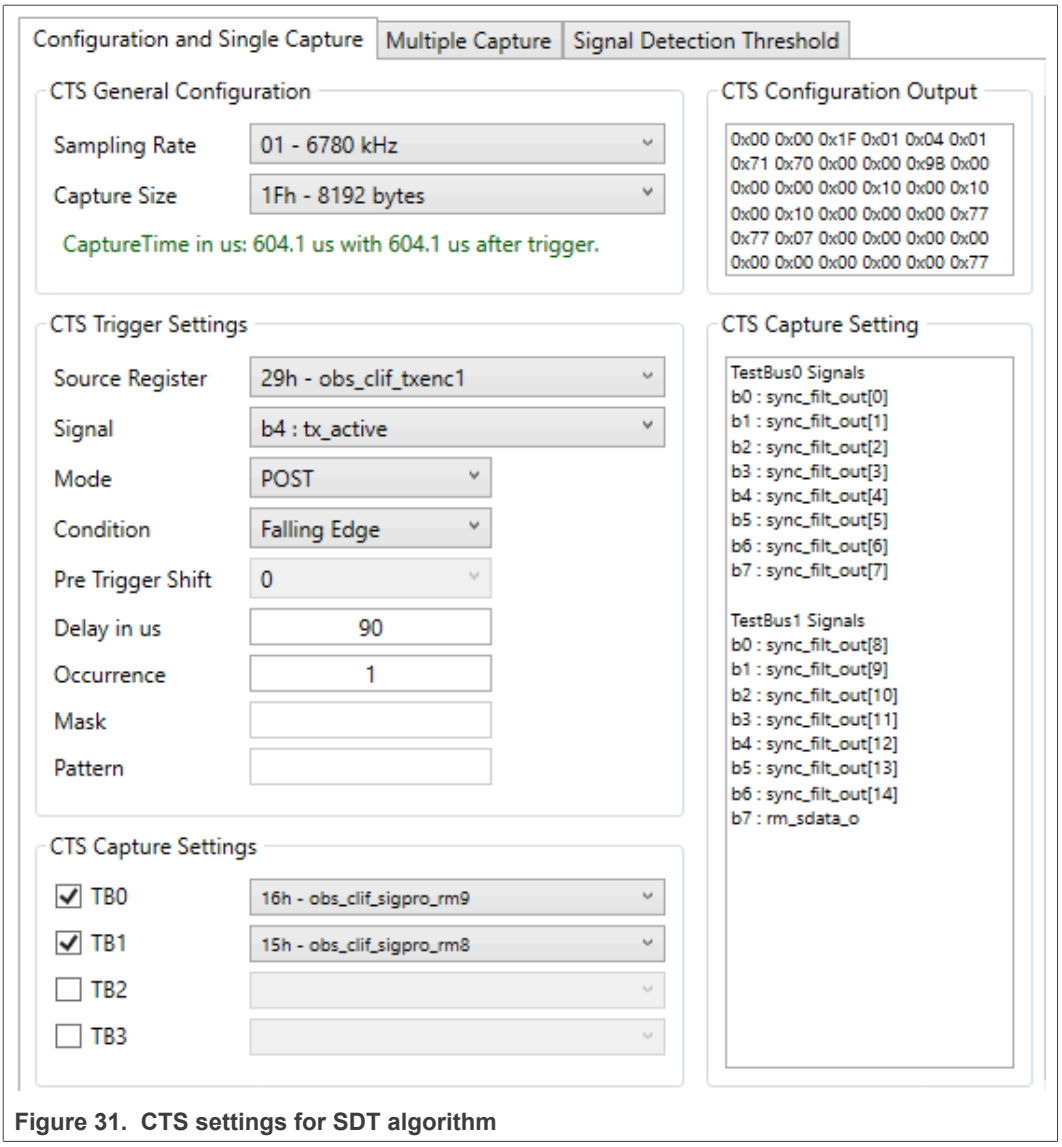


Figure 31. CTS settings for SDT algorithm

With these settings, a pure Sync filter output signal at 90 µs after the TX can be captured, using the ISO/IEC 14443 type 106 protocol settings. It returns 600 µs of typical noise floor, if no card is placed, as shown in Figure 32.

Note: *This capture requires an extended time-out setting and SDT = maximum, otherwise the time-out stops the RX before or a noise trigger might cause the RX to stop before the complete capture is taken. In the NFC Cockpit, this is automatically applied in the CTS multiple capture tab.*

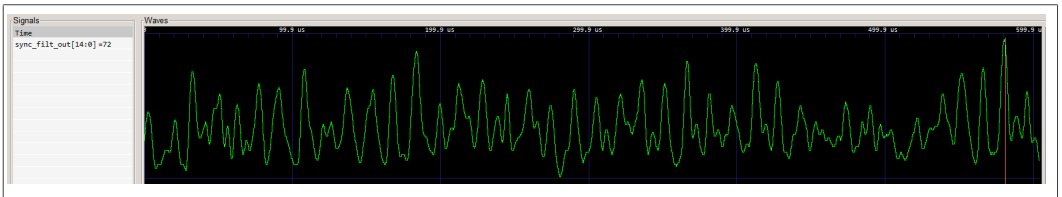


Figure 32. CTS capture of SDT, no card response

With this configuration, the tab Multiple Capture can be opened. After executing a <Load Protocol>, the user can select a number of captures from 0 up to 500, and then simply start the capturing (see Figure 33.)

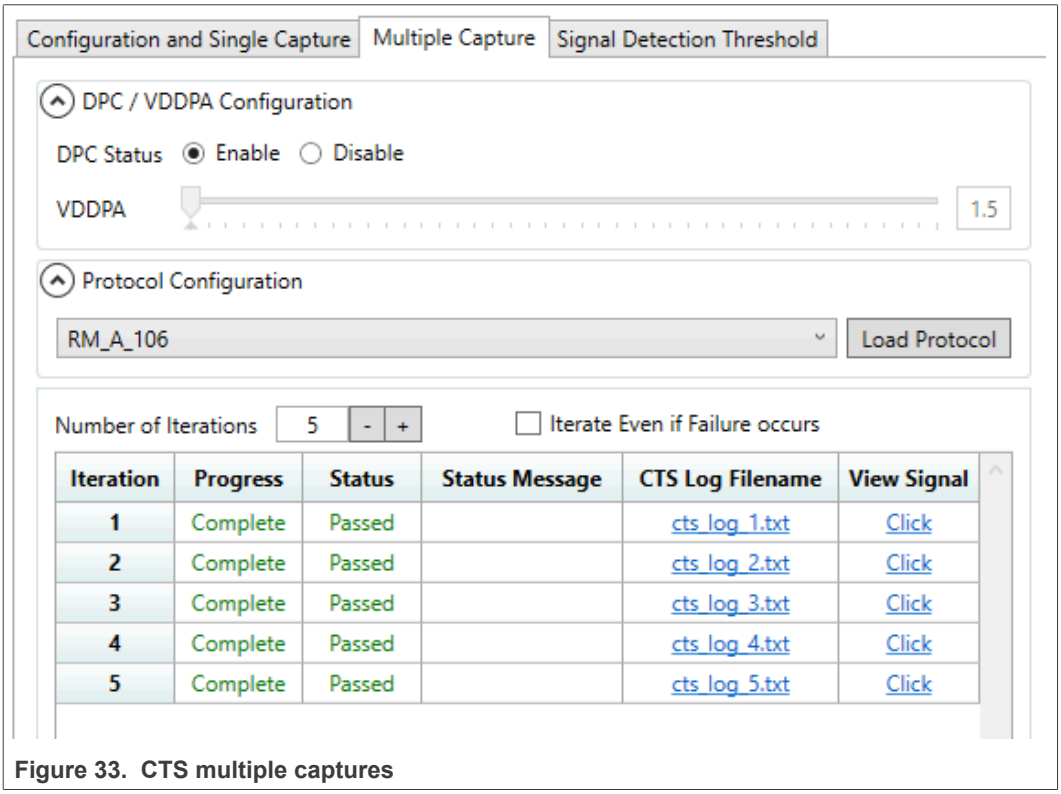
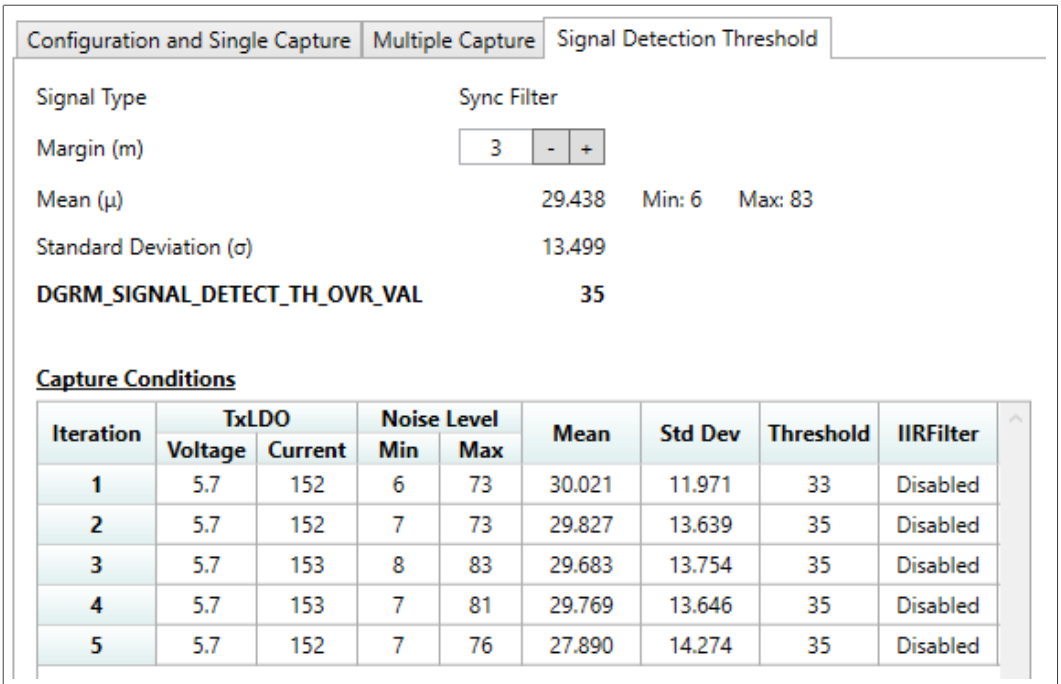


Figure 33. CTS multiple captures

All the captures are saved in separate sub folders with the NFC Cockpit folder, and can be saved into another folder, if needed (e.g. for further postprocessing). Each of the captured traces can be opened with the GTKWave viewer in the same way as with a single capture by clicking on <Click> in the View Signal column.

Note: *It might make sense to check some signals visually, especially if the SDT analysis shows unexpected values or huge differences between several traces (indicated in the Signal Detection Threshold tab).*

When all captures are taken, the Signal Detection Threshold tab can be chosen to analyze the data. The Figure 34 shows an example. The NFC Cockpit calculates some statistics: It derives the mean value, min and max values and the standard deviation. It allows to set a margin to derive a reasonable SDT. Under "normal" noise conditions a margin of 3 is sufficient: The SDT value, which is then used in the (14443 type A106) protocol settings, shall not be less than this value with margin =3.



Configuration and Single Capture | Multiple Capture | Signal Detection Threshold

Signal Type: Sync Filter

Margin (m): 3 - +

Mean (µ): 29.438 Min: 6 Max: 83

Standard Deviation (σ): 13.499

DGRM_SIGNAL_DETECT_TH_OVR_VAL: 35

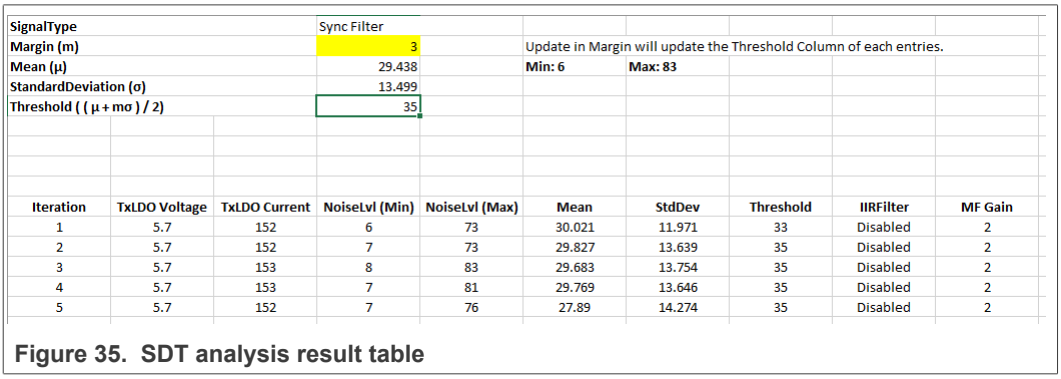
Capture Conditions

Iteration	TxLDO		Noise Level		Mean	Std Dev	Threshold	IIRFilter
	Voltage	Current	Min	Max				
1	5.7	152	6	73	30.021	11.971	33	Disabled
2	5.7	152	7	73	29.827	13.639	35	Disabled
3	5.7	153	8	83	29.683	13.754	35	Disabled
4	5.7	153	7	81	29.769	13.646	35	Disabled
5	5.7	152	7	76	27.890	14.274	35	Disabled

Figure 34. CTS SDT analysis

The capture conditions must be checked carefully, since different DPC (VDDPA and current) and or ARC (IIR filter and MF_Gain) settings can cause different SDT results. The statistics only make sense, if enough samples with same settings are captured. This can be guaranteed either by fixing (disabling) the DPC and / or by keeping the rf conditions stable. In the shown example the DPC is enabled, the device is used unloaded, but the current is limited, because the DC-DC is disabled, and the VUP = VBATPWR = 3.3 V.

The analysis result can be saved in an MS Excel table format for reports or further post processing, as shown in Figure 35.



SignalType	Sync Filter								
Margin (m)	3	Update in Margin will update the Threshold Column of each entries.							
Mean (µ)	29.438	Min: 6	Max: 83						
StandardDeviation (σ)	13.499								
Threshold ((µ + mσ) / 2)	35								

Iteration	TxLDO Voltage	TxLDO Current	NoiseLvl (Min)	NoiseLvl (Max)	Mean	StdDev	Threshold	IIRFilter	MF Gain
1	5.7	152	6	73	30.021	11.971	33	Disabled	2
2	5.7	152	7	73	29.827	13.639	35	Disabled	2
3	5.7	153	8	83	29.683	13.754	35	Disabled	2
4	5.7	153	7	81	29.769	13.646	35	Disabled	2
5	5.7	152	7	76	27.89	14.274	35	Disabled	2

Figure 35. SDT analysis result table

Note: *This SDT analysis is a useful tool to check the overall noise performance of an NFC reader system. It might even be useful to repeat the analysis under different frame conditions (different loading, different VDDPA, environmental changes like an LCD on or off). The visual check of the captures might help to identify noise sources and coupling mechanisms.*

3.1.4.2 Strength of matched filter concept

The matched filter concept works very efficiently.

There are only very parameters to adjust the RX operation. The theoretical advantage of the BPSK over ASK can be seen in a performance measurement, since the BPSK decoding requires less signal to noise (S/N) to achieve the same bit error rate (BER). Especially under real noise conditions the BPSK is more robust than ASK. So in a normal PN5190 design, measuring the sensitivity under similar frame conditions, the ASK decoding requires a stronger LMA level than a BSPK signal.

Note: *EMVCo LMA tests require to test "positive" and "negative" LMA, where the "negative" LMA is generated with an inverted modulation input signal at the TestPICC. Since the TestPICC cannot handle a constant high level at the modulation input, the modulation input is inverted (from low to high) shortly before the card modulation begins and back from high to low right after the card modulation has ended. This modulation input step before and after the card response causes a load switch, which is seen like an EMD event by the reader (example see Figure 19 and Figure 20). The switching times are set to the border of the EMD timings, which means that due to analog response behavior (delay), the LMA seen by the matched filter also contains the load switch. Especially in case of the ASK this causes an overall reduction of signal to noise. This is a behavior, which typically only applies to EMVCo analog tests.*

Use of IIR filter

There is hardly any benefit (if any at all) using the IIR filter other than with the pre-programmed damping: Either the noise frequency is outside the band and then is blocked / ignored by the matched filters anyhow, or it is inside the band and then cannot be blocked by the IIR filter without impacting the signal itself, too.

3.2 Test signals

The analog test signals as listed in Table 1 can either be captured internally, using the contactless interface test station (CTS), or routed to one of the available AUX pins.

Table 1. PN5190 RX analog test signals

#	Name	Nickname	Format	Recommended fs	Explanation
1	adc_data_i_i	ADC_I	10 bit signed	Fs ≥ 13.56 MHz / 4	Unfiltered I channel signal
2	adc_data_q_i	ADC_Q	10 bit signed	Fs ≥ 13.56 MHz / 4	Unfiltered Q channel signal
3	rm_cor_adc_i_o	Filtered_ADC_I	15 bit signed	Fs ≥ 13.56 MHz / 8	Pre-processed ADC data I channel, after IIR Filter and down-sampling
4	rm_cor_adc_q_o	Filtered_ADC_Q	15 bit signed	Fs ≥ 13.56 MHz / 8	Pre-processed ADC data Q channel, after IIR Filter and down-sampling
5	mf_pt_s0_d	S0	15 bit signed	Fs ≥ 13.56 MHz / 16	Delayed matched filter S0 output, after CH combiner
6	mf_pt_s1_d	S1	15 bit signed	Fs ≥ 13.56 MHz / 16	Delayed matched filter S1 output, after CH combiner
7	sync_filt_out	SyncOut (Sout)	15 bit signed	Fs ≥ 13.56 MHz / 16	Synchronization filter output

Table 2. PN5190 RX digital test signals

Name	Test bus	Adr.	Bit	State	Protocol	Description
mstate [1:0]	obs_clif_ rxdeco0	0x2C	1,0	0->1	A106, B106	RX active
mstate [1:0]	obs_clif_ rxdeco0	0x2C	1,0	1->2	A106, B106	first start bit in a frame
fpstate	obs_clif_ rxdeco3	0x2D	2	high	A106, B106	RX decoding active
fp_flg_collon_ detected	obs_clif_ rxdeco3	0x2D	4	high	A106	Collision detected

3.3 Debugging with Contactless Interface Test Station

The Contactless Interface Test Station (CTS) of the PN5190 is a powerful tool to analyze and optimize the analog performance of the contactless interface. The CTS offers many options to capture internal test signals. Since the capture size is limited due to memory limitations, and since the number of captures test buses is limited to 4 (or even only 2, if the sample rate is 13.56 MHz), it makes sense to carefully set the CTS configuration. The following section introduces a starting point and describes some considerations regarding the practical usage of the CTS.

In general, it is recommended to use the NFC Cockpit to start with the CTS. The NFC Cockpit offers a GUI, which gives a good overview to configure and use the CTS. And it even provides a good link to a graphical viewer (the GTKWAVE) to visualize the captured data. Even in a real debugging use case it normally makes sense to use that functionality to

- prepare the CTS configuration and to
- visualize the capture.

However, the application software needs to manage the CTS, otherwise a debugging in the real application is not possible. The CTS usage in principle is quite straight forward, since it only is related to three PN5190 commands:

CTS_CONFIGURE

The CTS_CONFIGURE is required to set up the CTS as such. It defines the sample rate, the trigger details and the test signals, which shall be captured. The configuration data, which is part of the command, can be derived easily from the NFC Cockpit, since the NFC Cockpit provides a GUI to configure the CTS. Then the configuration is shown in the GUI itself (see Figure 36), but it can also be saved. Saving the configuration with <Save Configuration> stores two different file formats into a folder:

- TXT-file: this file contains the raw hex data in ASCII, which can be copied into a source code.
- XML-file: this file saves the interpretation of the configuration. It can be reopened with the NFC Cockpit, but might also help the documentation.

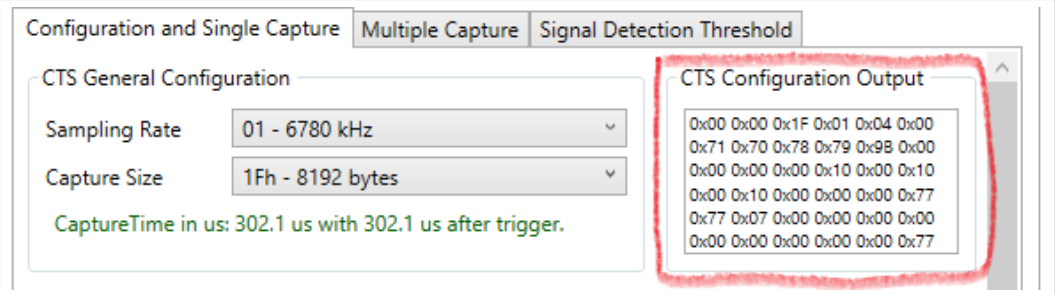


Figure 36. CTS Configuration shown in NFC Cockpit

The CTS_CONFIGURE is required before executing the CTS_ENABLE.

CTS_ENABLE

The CTS_ENABLE simply enables the CTS. After enabling, the CTS is active and waits until the trigger condition is met to capture the data.

CTS_RETRIEVE_LOG

When the trigger condition has occurred, i.e. some CTS has been captured, the CTS data can be retrieved with the CTS_RETRIEVE_LOG command. This data normally is saved into a file, which then afterwards can be used to do some post processing and visualization. The visualization can be done with the NFC Cockpit (in combination with the GTKWAVE).

3.3.1 Prepare and Configure the CTS

To prepare and configure the CTS, the NFC Cockpit can be used. However, the preparation depends on the use case.

Sample rate:

The sample very much depends on the captured test signals. For the I and / or Q channel signals typically a sample rate of at least 3390 kHz or better 6780 kHz makes sense. For the Matched Filter signals a lower sample rate makes sense, since it extends the capture time, i.e. increases the time period of the captured data.

Note: *Be aware that with a sample rate ot 13560 kHz, the number of test signals (test buses) is limited to 2.*

Capture size:

Normally the maximum capture size can be taken, since there is hardly any reason to reduce it.

Trigger option:

Normally there are two types of useful trigger conditions for debugging:

1. Trigger on TX-Event with post trigger (and delay)
2. Trigger on RX-Event (error) with pre trigger

The trigger on **TX-Event** typically uses the falling edge of the TxActive signal (Bit 4 of signal 0x29 obs_clif_txenc1) and some delay. This works specially good in type A during the card activation, where the PCD command and PICC response use a fixed timing. The advantage of using the TxActive as a trigger event is the fact that with every Transceive command there must be a CTS capture.

Note: *It might be useful to use the TxActibe as trigger event, but then let the application software decide, whether or not to retrieve the data. It might be good option to capture of every transaction, but only retrieve in case of an error.*

The trigger on **Rx-Event** typically uses the rising of the RxError signal (Bit 4 of the signal obs_clif_tbcontrol_patchbox14). As soon as the PN5190 receiver detects a receive error, it triggers the CTS capture. It makes sense to use a pre-trigger condition to check the test bus signals **before** the error happened.

Note: *Be aware that not every transaction necessarily causes a RxError trigger event, but only if an RX error happened. "Time out" is no RX error.*

Note: *Without trigger event, there is no data being captured. In such a case the CTS_RETRIEVE_DATA returns an error, since no data can be retrieved.*

A trigger event can only trigger one capture, i.e. the CTS needs to be configured and enabled again afterwards to capture a second time. However, the occurrence can be used to skip trigger events. An occurrence of e.g. 3, skips the first 2 trigger events, and captures the data, a soon as the trigger event occurs the third time.

Test signals:

Normally a good debugging test signal to start with is the SyncOut (see Table 1). It actually shows the "analog representation" of the response data, and allows a quick check on the quality of the overall RX signal.

In some cases it might make sense to capture the ADC_I and / or ADC_Q, which indicates the quality of the pure input. Any kind of noise or disturbance can be seen on these signals.

Note: *Be aware that the analog test signals require 2 test buses, since the test bus only offers 8 bit, while the test signals are 10 bit or even 14 bit wide. Use the patch box signals to always include the MSBit (sign) into the signal, even if only 8 bit (= one test bus) of the 10 or 14 bit are captured.*

There are two examples for the configuration, indicating a "good" capture for both type A and type B at 106 kbit/s in the Section 5.

3.3.2 CTS handling in the application software

The CTS handling must be done in the application software to allow debugging of the application itself. This CTS handling includes the execution of the three CTS commands as well as the control of events and capture data storage. Especially the event handling can support very useful and powerful mechanisms to select required trigger events or ignore capture data, which does not contain debug information. This section describes two main scenarios, using the 2 type of trigger events (TxActive and RxError).

A typical example might be a debug case, where every once a while some card response fails without obvious reason. So how can we get the CTS of such a fail case?

Using TXActive as trigger

In case of using the TXActive as trigger, every transceive command triggers a capture. So in such a capture scenario, there are many possible captures taken, before some "valid" data is captured, which shows the error case. However, since the application software knows, when an error occurs, it can control whether to capture and store the log data or not, as shown in a simple example in Figure 37. Instead of the only executing the transceive command, the software configures and enables the CTS before executing

the transceive command. Afterwards, only in case of an error the capture is retrieved and stored.

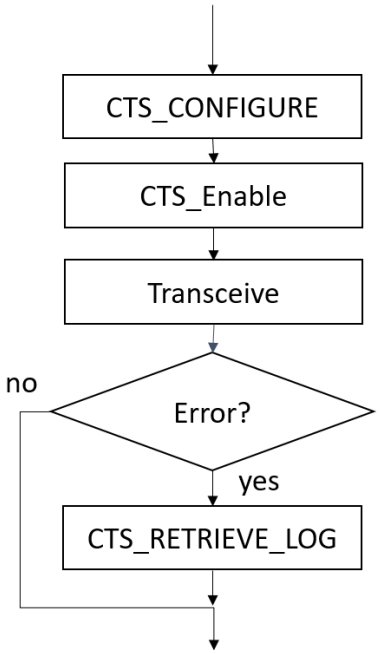


Figure 37. Simple example of CTS handling for debugging

Using RxError as trigger

When using the RxError as a trigger, only the RX error really triggers the CTS and creates capture data. Assuming that during the standard application flow there is supposed to be nor error, in such a case the application can simply configure and enable the CTS, then start the application, and then retrieve the captured data after the application has stopped due to the error.

3.3.3 Visualization of the CTS data

The visualization of the CTS can easily be done with the NFC Cockpit. The captured data needs to be saved as a text file (name.txt) into a folder. Then using the NFC Cockpit allows to click <Display Signals>, which then opens a file browser. Locate the text file, and then the NFC Cockpit does the postprocessing of the data and opens the GTKWave.

4 Low-Power Card Detection

The PN5190 offers a low-power card detection feature. This feature can be used to operate the PN5190 with an average low or very low power consumption, while the PN5190 wakes up automatically, as soon as a card is detected in proximity distance. Instead of an active card polling (according to ISO/IEC 14443 or NFC), the PN5190 uses a very short RF carrier pulse to detect an antenna detuning. The Figure 38 shows a capture of a sniffer signal to indicate the (U)LPCD principle. The cycle time defines the period of time, where the PN5190 "sleeps". At the end of every cycle, the PN5190 creates a short RF pulse ("RF ping"), which is used to detect a possible antenna detuning. The PN5190 automatically falls back to "sleep", if no detuning has been detected. A detuning causes the PN5190 to wake up, i.e. to stop the (U)LPCD, and then trigger the host µC via interrupt.

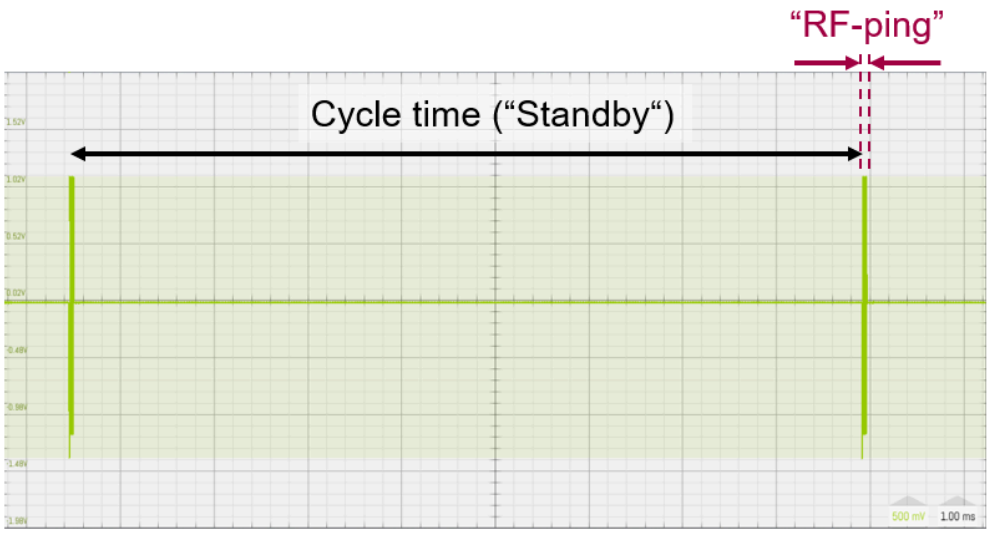


Figure 38. (U)LPCD principle

It is very obvious to keep the "RF ping" short and the cycle time long to reduce the average power consumption. However, the "RF ping" must be long enough to allow a proper detuning and avoid any false wake-up, and the cycle time defines the "reaction time". The longer the cycle time, the longer it might take, until the PN5190 detects the detuning. So these parameters depend on the use case, and need to be carefully defined to provide a proper user experience.

(U)LPCD principle: calibration and loop

The (U)LPCD always requires a calibration, which applies the (U)LPCD settings, takes the "status quo" of the antenna tuning (measures I/Q or RSSI value) and stores it as a "reference".

In a second phase, the (U)LPCD loop uses the same settings during the short RF ping and measures the I/Q or RSSI value. Then the measured values are compared to the reference, using a defined threshold window. The PN5190 wakes up, if the values exceed the threshold window.

The PN5190 offers three different LPCD modes:

1. **Semi autonomous LPCD**: This mode offers the function to calibrate and check the antenna detuning, without bringing the PN5190 into low-power mode. It allows to test LPCD parameters or check antenna detuning during normal operation. There is no power-down mode related to the Semi autonomous LPCD, i.e. there is no wake-up.
2. **LPCD**: The "normal" Low Power Card Detection (LPCD) uses the PN5190 internal µC to control the LPCD function. It provides a complete and very accurate I and Q channel measurement during the "RF ping". It allows to use the standard DPC operation including the DC-DC. The overall average LPCD current consumption can typically be in the range between 100 µA and 200 µA.
3. **ULPCD**: The Ultra Low Power Card Detection (ULPCD) uses a PN5190 hardware function to control the LPCD. The PN5190 internal µC is switched off, and there is no interface connection between host µC and the PN5190 during the ULPCD. This saves power and allows an average current consumption down to 20 µA or less. The ULPCD only works without DC-DC and uses the RSSI.

4.1 Semi autonomous LPCD

The Semi autonomous LPCD does not require any specific power configuration, since it works in normal operation. It does not use any standby or power-saving feature.

1. It does not use any low-power mode.
2. It uses the LPCD mechanism to calibrate the LPCD settings.
3. It uses the LPCD mechanism to read I and Q values.
4. It uses the LPCD EEPROM settings and read and write of registers only.

The Semi autonomous LPCD can be used to test LPCD parameters or analyze the loading and detuning of the antenna during the normal operation. After calibration, the reading of the I and Q values indicates, whether or not the antenna is detuned, and in which range it might be detuned. The values use the same mechanism as the LPCD itself, so the overall functionality can be tested without putting the PN5190 into (LPCD) standby. This might help to derive the optimum threshold settings for the LPCD itself.

The relevant EEPROM settings for the Semi autonomous LPCD are described in the section for the LPCD in Section 4.2:

1. LPCD_AVG_SAMPLES
2. LPCD_CONFIG
3. WAIT_RX_SETTLE
4. LPCD_VDDPA

These EEPROM settings are NOT relevant for the Semi autonomous LPCD:

1. LPCD_RSSI_TARGET: the RSSI_TARGET must be written into LPCD_CALIBRATE_CTRL register.
2. LPCD_RSSI_HYSTERESIS: the RSSI_HYSTERESIS must be written into LPCD_CALIBRATE_CTRL register.
3. LPCD_THRESHOLD_COARSE: There is no threshold applied in the Semi autonomous LPCD.

The semi-autonomous LPCD requires a calibration, before the I and Q values can be read.

The semi-autonomous LPCD is supported by the NFC Cockpit, as shown in Figure 39.

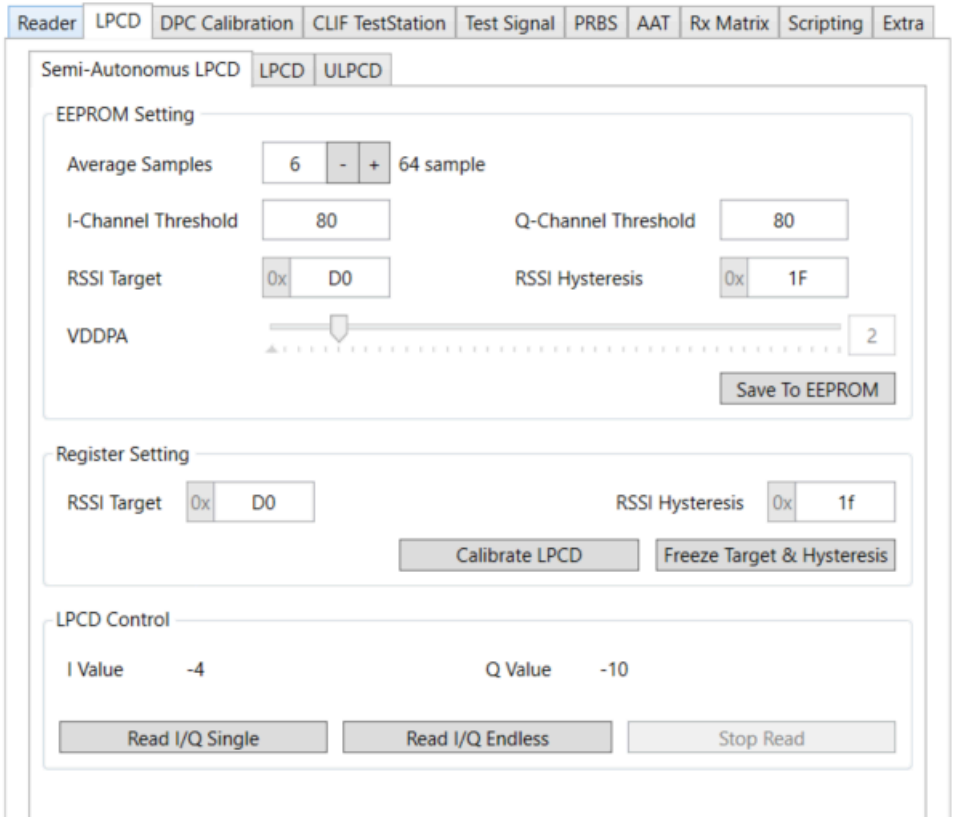


Figure 39. Semi-autonomous LPCD in NFC Cockpit

4.1.1 Semi autonomous LPCD calibration

The Semi autonomous LPCD calibration needs to be executed, before reading the I and Q values. Writing a valid RSSI_TARGET and RSSI_HYSTERESIS into the LPCD_CALIBRATE_CTRL register (0x50), while keeping the FREEZE_VALUE = 0, calibrates the LPCD.

Write RSSI_TARGET and RSSI_HYSTERESIS into LPCD_CALIBRATE_CTRL:

1. **FREEZE_VALUE = 0**: This calibrates the Semi autonomous LPCD. The calibration status can be checked in bit 31 of CALIBRATE_STATUS register (0x53).
2. **FREEZE_VALUE = 1**: This writes the RSSI_TARGET and RSSI_HYSTERESIS into the EEPROM LPCD_RSSI_TARGET (0x494) and LPCD_RSSI_HYSTERESIS (0x496).

Note: *The values of RSSI_TARGET and RSSI_HYSTERESIS in EEPROM (0x494, 0x496) are used for the LPCD, but not for the Semi autonomous LPCD.*

Values for RSSI_TARGET and RSSI_HYSTERESIS can be taken from the default protocol settings of type A106:

- RSSI_TARGET = 0x2A3
- RSSI_HYSTERESIS = 0x1F

4.1.2 Semi autonomous LPCD reading

After LPCD calibration, the I and Q values can be read. Reading the IQ_CHANNEL_VALS register (0x51), enables the RF carrier for a short "RF ping", using the EEPROM settings of WAIT_RX_SETTLE and LPCD_VDDPA, and returns a 16 bit signed integer for both I and Q value (Q_CHANNEL_VAL and I_CHANNEL_VAL). The value range depends on the LPCD_AVG_SAMPLES setting (see Table 3).

The calibration normally pulls at least one of the two values into the center of the range, i.e. to values relatively close to zero. Detuning or loading the antenna then is indicated via an increase or decrease of the I and / or Q value.

Note: *The "RF ping length" can be calculated with*

$$\text{length} \approx 100\mu\text{s} + \text{WAIT\_RX\_SETTLE}\,[\mu\text{s}] \quad (1)$$

4.2 Low-Power Card Detection

The Low-Power Card Detection (LPCD) uses the PN5190 internal µC to control the standby and card detection measurement. It keeps the host interface active, i.e. the LPCD can be stopped by a host command via SPI. The LPCD uses both the I and Q channel values, which provide a very good accuracy and detection range.

LPCD Calibration:

The LPCD uses the SWITCH_MODE_LPCD command to calibrate the LPCD settings. A calibration before starting the LPCD is required.

LPCD Loop:

The LPCD uses the SWITCH_MODE_LPCD command to start the LPCD.During the LPCD operation itself, the PN5190 stays in standby for the length of the cycle time. The timer value for the cycle time is part of the SWITCH_MODE_LPCD command, and can be up to 2690 ms. At the end of each cycle time, the PN5190 generates a short "RF ping" to measure the I and Q value and compare them with the threshold settings. As soon as the measured values exceed the threshold settings, the PN5190 stops the LPCD operation, wakes up and generates an interrupt to the host µC. The LPCD standby continues with another cycle time, if no card is being detected, i.e. if the measured values do not exceed the threshold settings.

The NFC Cockpit provides the best interface to test and adjust the LPCD settings. The Figure 40 shows an example of a typical LPCD configuration to start with. In a first step **<Calibrate LPCD>** calibrates the LPCD settings. The NFC Cockpit shows the I and Q values after calibration.

Starting the LPCD with **<Single LPCD>** starts the LPCD loop. As soon as a card is being detected, the LPCD stops. Restarting the LPCD typically requires a new calibration, otherwise the PN5190 might wake up immediately again.

The **<Endless LPCD>** restarts the LPCD after a wake-up automatically without recalibration. So placing a card and keeping it will always force a wake-up right after restarting the LPCD. Only removing the card (and bringing the tuning back to the original default) will stop the wake ups.

The **<Auto LPCD>** performs a new calibration after a wake-up, be automatically starting the LPCD loop again. Thus placing a card and keeping it, will cause a wake-up, but due to the recalibration (using the detuned antenna), the LPCD will stay in LPCD loop, if the card is not moved. Removing the card out of the detection range will change the antenna detuning again and cause a new wake-up.

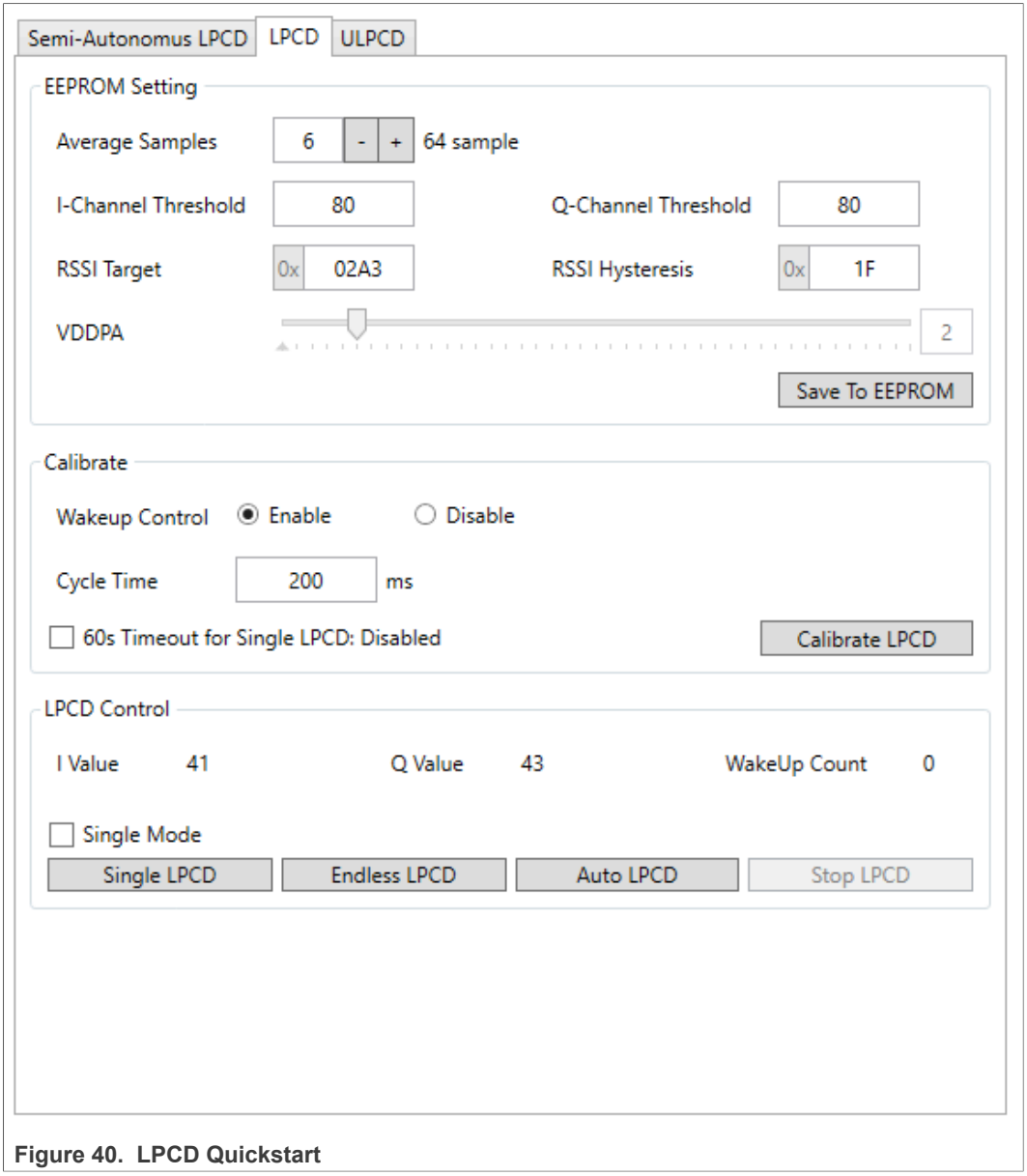


Figure 40. LPCD Quickstart

Modifying the I and Q channel Threshold value in the EEPROM normally is the best (and only) option to modify the LCPD sensitivity. However, the details of all the LPCD-related settings in the EEPROM are described in the following sections.

Note: *It might make sense to adjust and optimize the RF ping length (WAIT_RX_SETTLE) to reduce the average LPCD current consumption, even though that is not required to improve the detection range. Anyway, the ULPCD might be the better option, if the lowest average current consumption is the major requirement.*

4.2.1 DCDC_CONFIG

The DCDC_CONFIG is a single-byte EEPROM setting in address 0x01. It defines the use of the DC-DC during the LPCD.

1. Bit 3 = 0: The DC-DC is not used during the LPCD RF Ping.
2. Bit 3 = 1: The DC-DC is used during the LPCD RF Ping.

Note: *Normally it recommended to NOT use the DC-DC for the RF ping, since the recommended setting for the LPCD_VDDPA sets the VDDPA below the VBATPWR anyhow. There is hardly any benefit in using the DC-DC during the LPCD RF ping.*

4.2.2 LPCD_AVG_SAMPLES

The LPCD_AVG_SAMPLES is a single-byte EEPROM setting in address 0x492. It defines the number of samples, allowing values from 0 to 6. The Table 3 shows the value range and a typical related threshold.

Note: *The impact of this value in a real application is minor, as long as the threshold setting is applied accordingly.*

Table 3. LPCD_AVG_SAMPLES

Value	Value range	typical threshold
0	≈-30,000 … +30,000	3200
1	≈-16,000 … +16,000	1600
2	≈-8,000 … +8,000	800
3	≈-4,000 … +4,000	400
4	≈-2,000 … +2,000	200
5	≈-1,000 … +1,000	100
6	≈-500 … +500	50

4.2.3 LPCD_RSSI_TARGET

The LPCD_RSSI_TARGET is a single-byte EEPROM setting in address 0x494. It defines the target for the RSSI control of the PN5190 RX for the LPCD operation. Since this value is not specially described, it is recommended to use and not change the default settings (= RSSI target value for RM type A).

4.2.4 LPCD_RSSI_HYST

The LPCD_RSSI_HYST is a single-byte EEPROM setting in address 0x496. It defines the hysteresis for the RSSI control of the PN5190 RX for the LPCD operation. Since this value is not specially described, it is recommended to use and not change the default settings (= RSSI hysteresis value for RM type A).

4.2.5 LPCD_CONFIG

The LPCD_CONFIG is a two byte EEPROM setting in address 0x497 and 0x498. It defines some basic settings, which *normally do not need to be changed*.

The least significant 3 bit (bit 0 to bit 2) define the use of the magnitude and / or I and Q signal for the LPCD measurement. The default setting uses I and Q channel, which typically gives the best results.

Bit 3 enables both drivers (default for the normal balanced antenna design). Disabling Bit 3 only uses TX1 during the LPCD RF ping.

Bit 4 enables the VDDPA fast discharge after the RF ping. This is the normal way to switch off the RF carrier, and applies, when the VDDPA > VBATPWR, i.e. the DC-DC is active. In such case, the VDDPA is switched to 1.5 V before shutting down. This

discharges the supply and is used to stabilize the starting condition for the next RF ping. With lower VDDPA setting (LPCD_VDDPA) the discharge is not required. However, discharging the VDDPA (enabling Bit 4) reduces the average power consumption.

Bit 5 enables the RF carrier to be shut down, before the TXLDO is disabled after the RF ping. Enabling Bit 5 reduces the average power consumption. Disabling the Bit 5 might cause the VDDPA to stay high after RF off.

4.2.6 LPCD_THRESHOLD_COARSE

The LPCD_THRESHOLD_COARSE is a four byte EEPROM setting in address 0x49A … 0x49D. It defines the threshold, allowing two byte values for both the I and Q channel. As soon as the measured value exceeds the threshold (positive or negative), the PN5190 wakes up. This value defines the balance between "sensitivity" (low threshold) and "robustness" (high threshold), so it makes sense to test and adjust this value carefully.

Note: *The LPCD_THRESHOLD_COARSE must be defined in combination with the LPCD_AVG_SAMPLES. See Table 3.*

4.2.7 WAIT_RX_SETTLE

The WAIT_RX_SETTLE is a single-byte EEPROM setting in address 0x4AB. It defines the time after enabling the RF carrier (during the "RF ping"), before the sampling starts in steps of µs. The sampling takes another 20 µs. So in total the RF ping length can be defined with

$$\text{length} \approx 20\mu\text{s} + \text{WAIT\_RX\_SETTLE}\ [\mu\text{s}] \quad (2)$$

Note: *The overall ping length must be long enough to allow a proper and stable RF carrier level, before the sampling starts. Otherwise the measurement (and therefore the card detection) might not be accurate enough. The optimum RF ping length might depend on the power supply concept as well as the overall antenna tuning.*

4.2.8 LPCD_VDDPA

The LPCD_VDDPA is a single-byte EEPROM setting in address 0x4AF. It defines the VDDPA, using the known range from 0x00 = 1.5 V up to 0x2A = 5.7 V, which is applied during the LPCD RF ping. Normally this can be the lowest VDDPA to save power, since the detection range does not depend much on the power level. The antenna detuning can typically be detected with a low RF power level as well as with a normal or high-power level. However, a low-power level helps to reduce the overall average power consumption, and even a very strong detuning might not exceed any power limit.

4.3 ULPCD

The Ultra Low Power Card Detection (ULPCD) is a hardware function of the PN5190, which puts the PN5190 into a standby (low power) mode. The internal PN5190 µC is deactivated, and there is no SPI interface function available during the ULPCD. An internal low frequency oscillator (LFO) is used to trigger a very short RF carrier pulse ("RF ping"), after a defined cycle time has executed. This "RF ping" is used to check any antenna loading or detuning. The ULPCD continues applying another cycle time in standby, if no loading or detuning has been detected, or it wakes up the PN5190 µC, in case the antenna had been detuned or loaded.

The ULPCD requires a specific power configuration: the use of the internal DC-DC excludes the the use of ULPCD.

Note: *The use of the DC-DC requires an overall power management (timing), which does not make much sense in combination with ultra low-power design, since the DC-DC is designed to drive a very high output power level. For typical battery powered, low-power reader devices, the use of the DC-DC does not make sense, but the ULPCD is available,*

Note: *Be aware that disabling the DC-DC requires a hardware modification (to supply VUP) as well as related EEPROM settings different from the default settings (DCDC_POWER_CONFIG, 0x00). Wrong settings in combination with certain power configurations* ***might destroy the PN5190****!*

ULPCD Calibration:

The ULPCD uses the SWITCH_MODE_LPCD command to calibration the ULPCD settings. The ULPCD requires a calibration before starting the ULPCD loop itself. The calibration requires the HF Attenuator value as a parameter of the command, which must be derived and fixed upfront.

ULPCD Loop:

The ULPCD uses the SWITCH_MODE_LPCD command to start the ULPCD loop. The ULPCD requires a calibration before starting the ULPCD itself. The ULPCD loop requires the same HF Attenuator value, as used for the ULPCD calibration, and the cycle time value value as parameters of the command. The cycle time can be up to 4095 ms. Starting the ULPCD loop switches the PN5190 into the ULPCD standby mode, which is only been left, if

1. a card (detuning / loading) is being detected, or
2. the PN5190 is being reset by the host µC, or
3. the GPIO3 is being toggled.

PNEV5190BP Hardware preparation:

The PNEV5190BP is already prepared to support the operation without DC-DC. Normally only a single jumper at J13 is required to supply the VUP externally with VBATPWR. The Figure 41 shows the older version of the PNEV5190BP, which required some soldering to prepare the board for the ULPCD. The release version of the board only requires the Jumper J13 to be closed (no DC-DC for ULPCD) or reopened again ("normal" DC-DC usage).

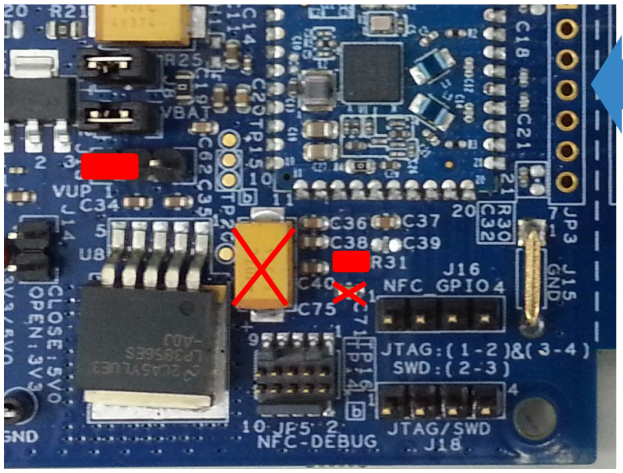


Figure 41. PNEV5190BP HW preparation for ULPCD

Before switching on the RF field, the DC-DC must be disabled! This is done with the DCDC_CONFIG (see Section 4.2.1):

- DCDC_CONFIG = 0xE4: Default usage of the DC-DC, ULPCD not allowed.
- DCDC_CONFIG = 0x21: DC-DC disabled, VUP = VBATPWR, ULPCD allowed.

NFC Cockpit ULPCD:

The NFC Cockpit provides a useful interface to test and optimize the ULPCD settings, as shown in Figure 42.

The <Read HF Attenuator> stores the DPC setting (DPC_CONFIG), disables the DPC, forcing the VDDPA to the set value, enables the RF field and reads the HF Attenuator value from the RX_CTRL_STATUS register. Afterwards the original DPC setting is recovered to guarantee the proper normal operation. The HF Attenuator value is shown in the GUI and stored for the ULPCD function.

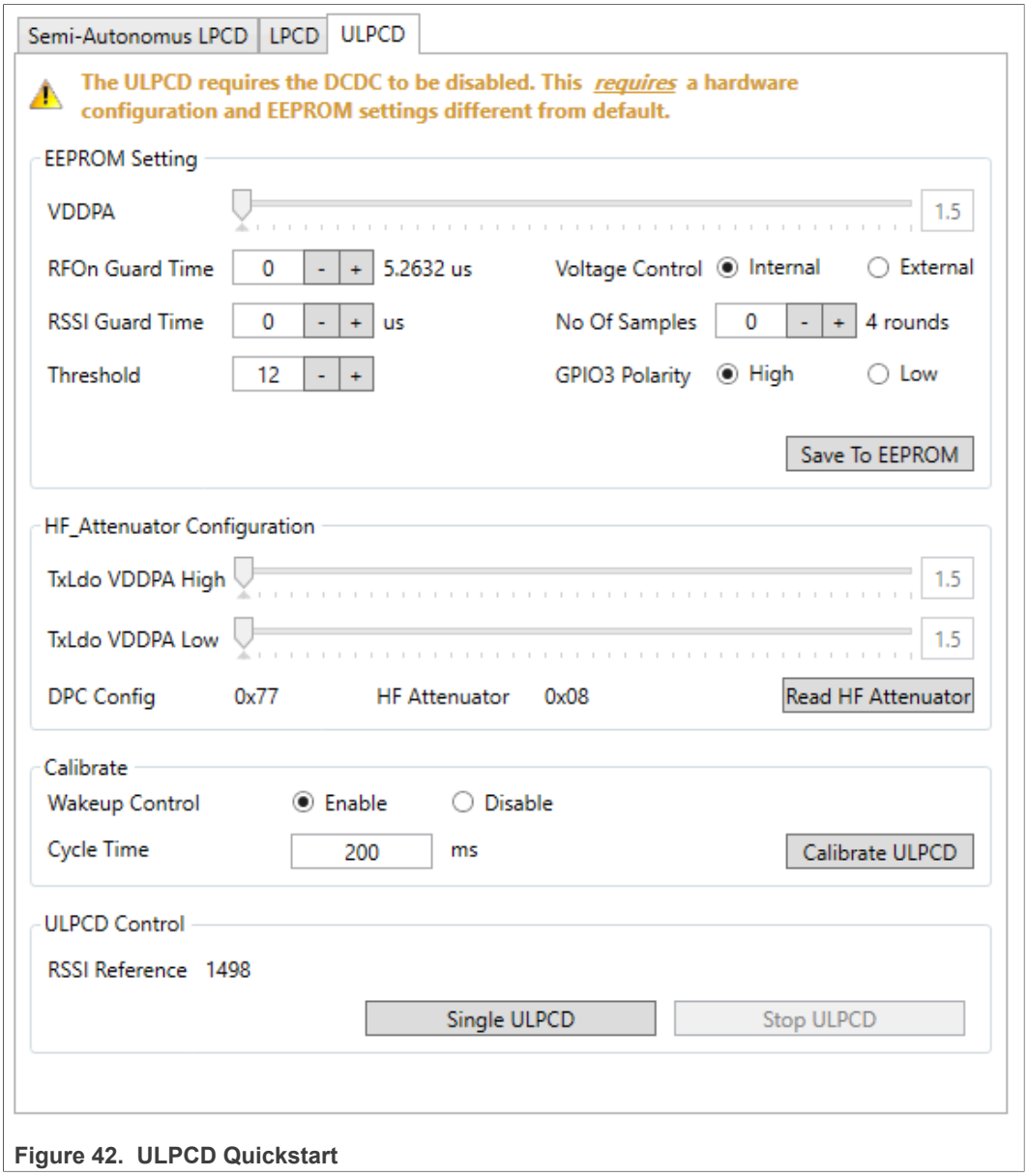


Figure 42. ULPCD Quickstart

<Calibrate ULPCD> executes the SWITCH_MODE_LPCD, performing an ULPCD calibration based on the given settings (including the HF Attenuator). As a result the returned RSSI reference value is shown in the NFC Cockpit (only for info).

<Single ULPCD> executes the SWITCH_MODE_LPCD, starting the ULPCD loop.

Note: *Be aware that the ULPCD loop execution stops the SPI communication, since the PN5190 µC is disabled. So only a card detection (antenna detuning / loading) can properly stop the NFC Cockpit ULPCD.*

4.3.1 HF Attenuator value

The HF Attenuator value is required for both the ULPCD calibration and the ULPCD loop as part of the SWITHC_MODE_LPCD command. The HF Attenuator value guarantees the optimum RSSI range, even though the VDDPA might be reduced. So the best way to derive the optimum HF Attenuator value is to enable the RF carrier

once (using the Type A 106 settings), but forcing the VDDPA into the same value, as set in ULPCD_VDDPA_CTRL. Then reading the bit 3 to bit 8 of the RX_CTRL_STATUS (address 0x28h) returns the HF Attenuator value, which should be stored in the Host FW and then always be used for the ULPCD.

4.3.2 ULPCD_VDDPA_CTRL

The ULPCD_VDDPA_CTRL is a two byte EEPROM setting in address 0x4BF and 0x4C0. It defines the VDDPA within the bits 3 to 8, using the known range from 0x00 = 1.5 V up to 0x2A = 5.7 V, which is applied during the ULPCD RF ping. Normally this can be the lowest VDDPA to save power, since the detection range does not depend much on the power level. The antenna detuning can typically be detected with a low RF power level as well as with a normal or high-power level. However, a low-power level helps to reduce the overall average power consumption, and even a very strong detuning might not exceed any power limit.

Note: *Keep the other bits = RFU → do not change!*

4.3.3 ULPCD_TIMING_CTRL

The ULPCD_TIMING_CTRL is a single-byte EEPROM setting in address 0x4C2. It defines the RF On guard time, i.e. the time between the RF on and the first sampling (measurement). This time is required to ensure a stable RF carrier for a reliable sampling. However, even a value of 0 might work properly.

The RF On guard time can be estimated with

$$RF\_ON\_Guard\_time \approx 6\mu s + ULPCD\_TIMING\_CTRL \cdot 3\ [\mu s] \quad (3)$$

4.3.4 ULPCD_VOLTAGE_CTRL

The ULPCD_VOLTAGE_CTRL is a single-byte EEPROM setting in address 0x4C6. It defines whether the VUP is externally supplied by an extra supply voltage (bit 1 = 0) or the VUP is directly connected to VBAT/VBATPWR (bit 1 = 1).

Note: *Keep the other bits = RFU → do not change!*

4.3.5 ULPCD_RSSI_GUARD_TIME

The ULPCD_RSSI_GUARD_TIME is a two byte EEPROM setting in address 0x4C9 and 0x4CA, which defines the time between 2 consecutive samples. So this time defines the RF ping length in combination with the number of RSSI samples in addition to the ULPCD_RFON_Guard_Time (see Section 4.3.6).

Note: *Keep the other bits = RFU → do not change!*

4.3.6 ULPCD_RSSI_SAMPLE_CFG

The ULPCD_RSSI_SAMPLE_CFG is a single-byte EEPROM setting in address 0x4CA, which defines the number of samples:

- 0: 4 samples
- 1: 8 samples
- 2: 16 samples
- 3: 32 samples

The averaging of more samples reduces the level of noise, but increases the overall RF ping length, which increases the average power consumption. However, in most cases even the capture of 4 samples give sufficient results.

The RF ping length can be estimated with

$$RFPingLength \approx 6\mu s + RFON \cdot 3\mu s + \frac{RSSIGUARD \cdot \#OfSamples}{27}\mu s + \frac{\#OfSamples \cdot 136}{100}\mu s$$

- RFPinglength = length of the RF carrier pulse
- RFON = value of ULPCD_TIMING_CTRL
- RSSIGUARD = value of ULPCD_RSSI_GUARD_TIME
- #OfSamples = number of samples (4 ...32) according to ULPCD_RSSI_SAMPLE_CFG

4.3.7 ULPCD_THRESH_LVL

The ULPCD_RSSI_SAMPLE_CFG is a single-byte EEPROM setting in address 0x4CB, which defines the RSSI threshold value in the range from 0 to 31. During the ULPCD calibration, the PN5190 stores an RSSI level as reference. During the ULPCD loop, the PN5190 captures the RSSI at every RF ping. As soon as the captured value is higher than the reference + ULPCD_THRESH_LVL or lower than reference - ULPCD_THRESH_LVL, the PN5190 wakes up.

It makes sense to test and adjust this value carefully: With a lower ULPCD_THRESH_LVL, the card detection becomes more sensitive but less robust. Any ULPCD_THRESH_LVL of 10...20 normally is a good starting point.

4.3.8 ULPCD_GPIO3

During the ULPCD loop, the SPI interface is disabled (like the PN5190 internal µC), so the host µC cannot control the PN5190, while the ULPCD loop is running. The ULPCD can only be stopped

1. if a card (detuning / loading) is being detected,
2. if the PN5190 is being RESET (using the RESET_N pin), or
3. if the GPIO3 is being toggled.

The ULPCD_GPIO3 is a single-byte EEPROM setting in address 0x4CC, which defines the polarity of GPIO3. With bit 0 = 0, a high level at GPIO3 stops the ULPCD, with bit 0 = 1, a high stops the ULPCD.

4.3.9 Average power consumption

The average power can be estimated, using the following formula:

$$I_{Average} \approx \frac{4500 + \left(\frac{5 \cdot Cycletime}{[ms]}\right) + \left(\frac{ITVDD}{[mA]} \cdot \frac{RFPingLength}{[\mu s]}\right)}{\frac{Cycletime}{[ms]}}\ \mu A$$

- $I_{Average}$ = overall average current in µA
- Cycletime = cycle time, as defined in the ULPCD command call in ms
- RFPingLength = length of the RF carrier on time in µs (as estimated in Section 4.3.6)

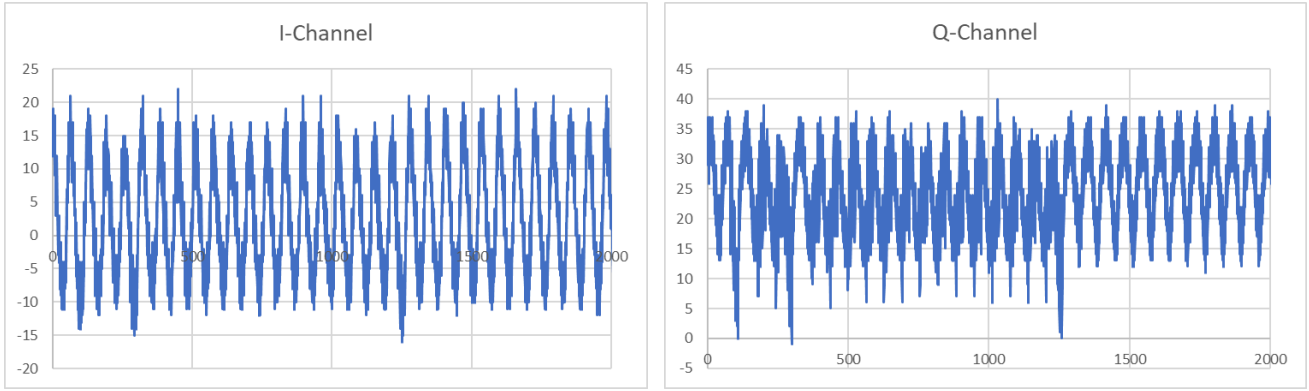
Note: *This estimation for the average current consumption cannot be an accurate calculation, but only an estimation. Be aware that leakage currents of several 10 µA or 100 µA can easily occur, if the hardware design has not been carefully checked.*

5 Appendix

5.1 Noise robustness

Noise can easily cause serious decrease of RX performance. However, the matched filter concept of the PN5190 receiver is very strong against noise. Especially the BPSK is very robust.

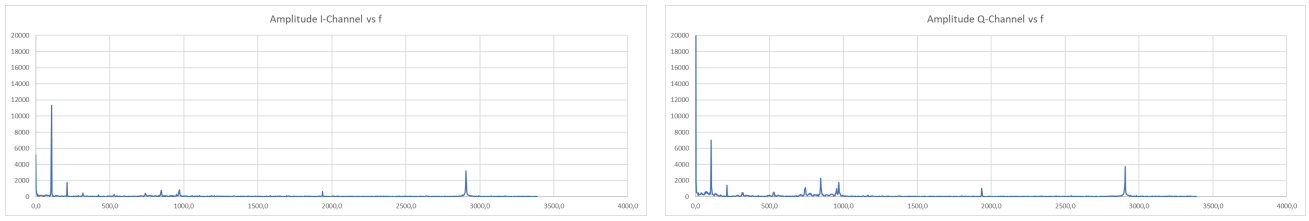
The Figure 43 and Figure 44 show the ADC_I and ADC_Q signal of an ATQA response including a strong 106 kHz noise signal. The type A card is placed at the maximum reading distance, so the ATQA is weak. The noise signal of 106 kHz is 10 dB stronger than the ATQA signal itself, but still there is no error in decoding the ATQA.



ADC_I with 106 kHz noise

ADC_Q with 106 kHz noise

Figure 43. 106 kHz noise and ATQA in time domain



ADC_I with 106 kHz noise

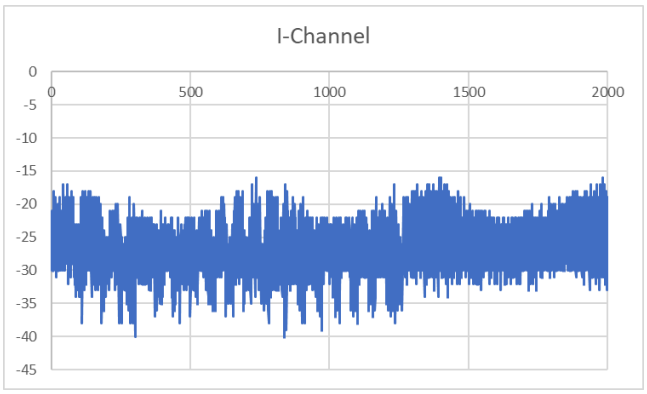
ADC_Q with 106 kHz noise

Figure 44. 106 kHz noise and ATQA in frequency domain

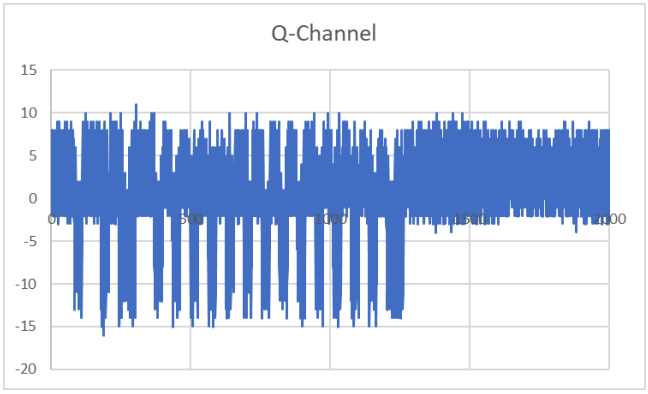
The Figure 45 and Figure 46 show the ADC_I and ADC_Q signals in time and frequency domain of an ATQA including a 12.602 MHz noise coupled into the antenna. The noise frequency is quite “bad” for the decoding of ISO 14443 type A signals, so the level of noise is in the same magnitude as the ATQA signal itself. At this limit combination, the ATQA decoding starts to fail in this case.

The Figure 47 and Figure 48 show the ADC_I and ADC_Q signals in time and frequency domain of an ATQB including a 12.602 MHz noise coupled into the antenna. In this case, the ATQB signal is 6 dB stronger than the ATQA from the previous example. However, the noise in this case is 12 dB stronger than in the previous example (which means the

noise is 6 dB bigger than the ATQB), and still the BPSK decoding works properly without error.

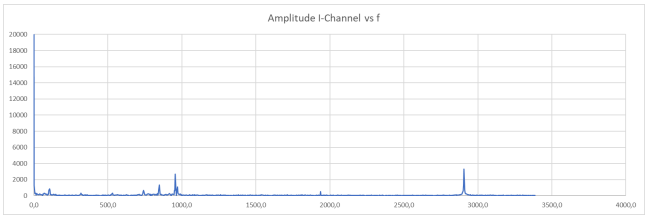


ADC_I with 12602 kHz noise

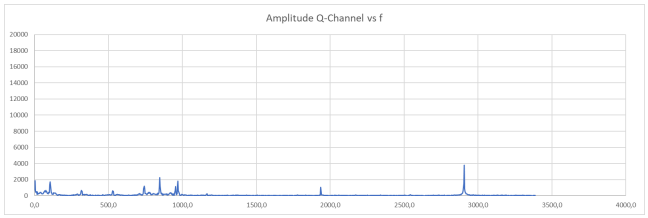


ADC_Q with 12602 kHz noise

Figure 45. 12602 kHz noise and ATQA in time domain

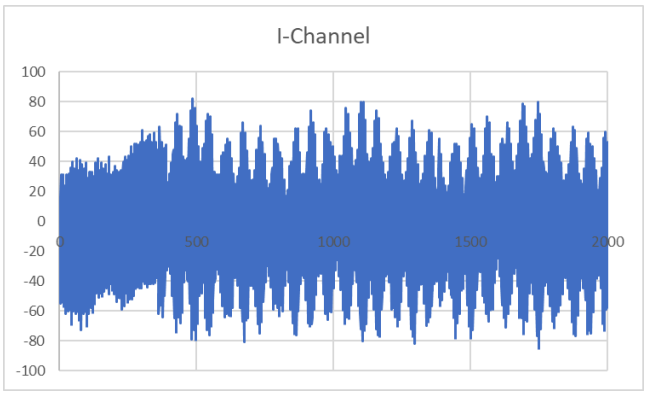


ADC_I with 12602 kHz noise

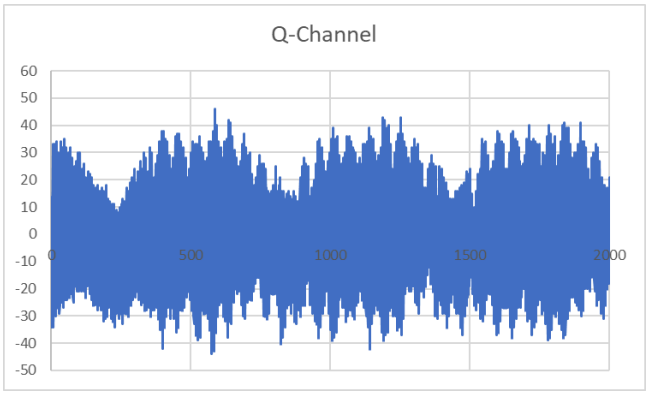


ADC_Q with 12602 kHz noise

Figure 46. 12602 kHz noise and ATQA in frequency domain

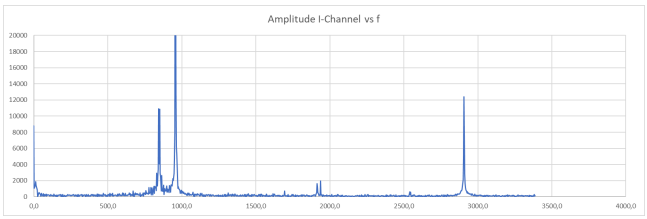


ADC_I with 12.602 kHz noise



ADC_Q with 12.602 kHz noise

Figure 47. 12.602 kHz noise and ATQB in time domain



ADC_I with 12.602 kHz noise

ADC_Q with 12.602 kHz noise

Figure 48. 12.602 kHz noise and ATQB in frequency domain

5.2 CTS example for A106 debugging

This example uses the sample rate of 3390 kHz and captures the SyncOut and ADC_I, using a double size UID MIFARE Classic with 1K memory card with its ATQA = 0x4400 in a reasonable distance. Of course, this capture does not indicate any error. The configuration is set up in the NFC Cockpit as shown in Figure 49.

This is related configuration data:

```
0x00 0x00 0x1F 0x02 0x04 0x00 0x71 0x70 0x78 0x79 0x9B 0x00 0x00
0x00 0x00 0x10 0x00 0x10 0x00 0x10 0x00 0x00 0x00 0x77 0x77 0x07
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x77 0x77 0xB7 0x3C
```

which can be found in the CTS Configuration Output (see Figure 49 as well as in the save TXT file, when saving the configuration.

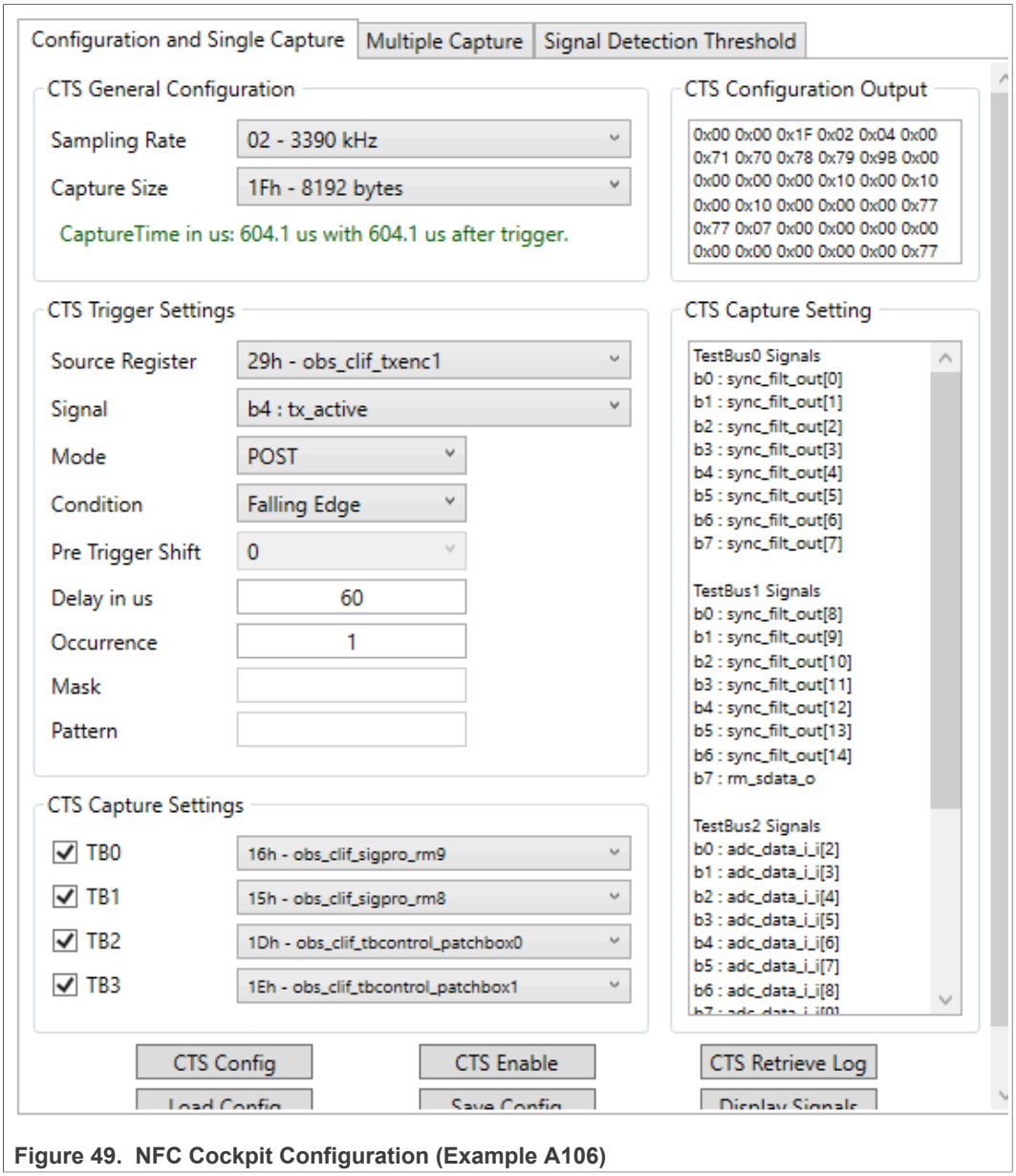


Figure 49. NFC Cockpit Configuration (Example A106)

The captured data is shown in Figure 50. It can be seen that with a sample rate of 3390 kHz, all 4 test buses and the full capture size used, the CTS can capture 600 µs.

The Figure 51 shows the same capture, except the fact that the sample rate is changed to 1695 kHz and the ADC_I data is disabled (i.e. only the SyncOut is captured): in that case the trace can show 2400 µs.

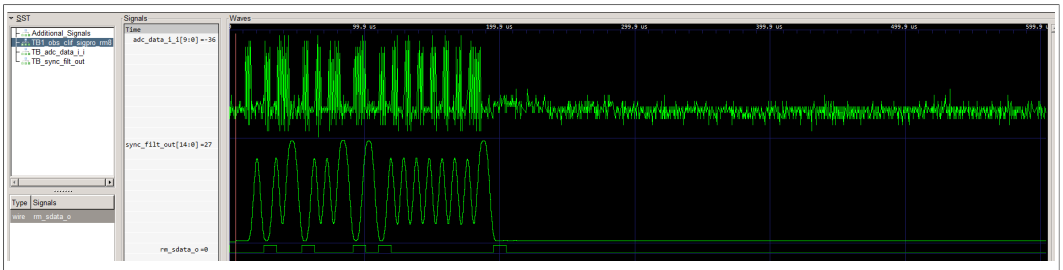


Figure 50. A106 example capture (ATQA = 0x4400)

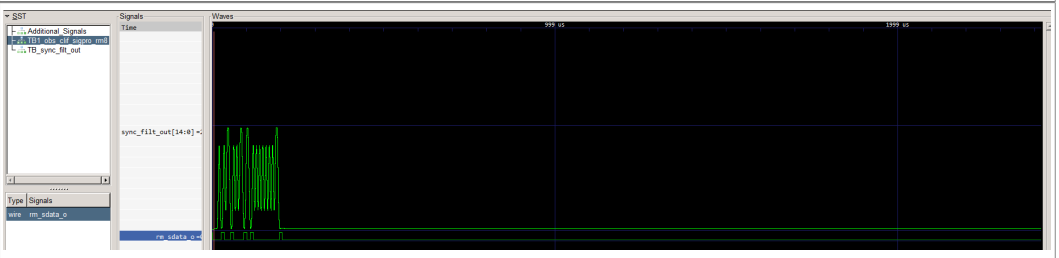


Figure 51. A106 example capture 2 (ATQA = 0x4400)

5.3 CTS example for B106 debugging

This example uses the sample rate of 3390 kHz and captures the SyncOut and ADC_I, using a type B card with an ATQB = 0x50 10 18 4C E6 11 00 00 11 77 81 C3 in a reasonable distance. Of course, this capture does not indicate any error. The configuration is set up in the NFC Cockpit as shown in Figure 52.

This is the related configuration data:

```
0x00 0x00 0x1F 0x02 0x04 0x00 0x71 0x70 0x78 0x79 0x9B 0x00 0x00
0x00 0x00 0x10 0x00 0x10 0x00 0x10 0x00 0x00 0x00 0x77 0x77 0x07
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x77 0x77 0x57 0x5F
```

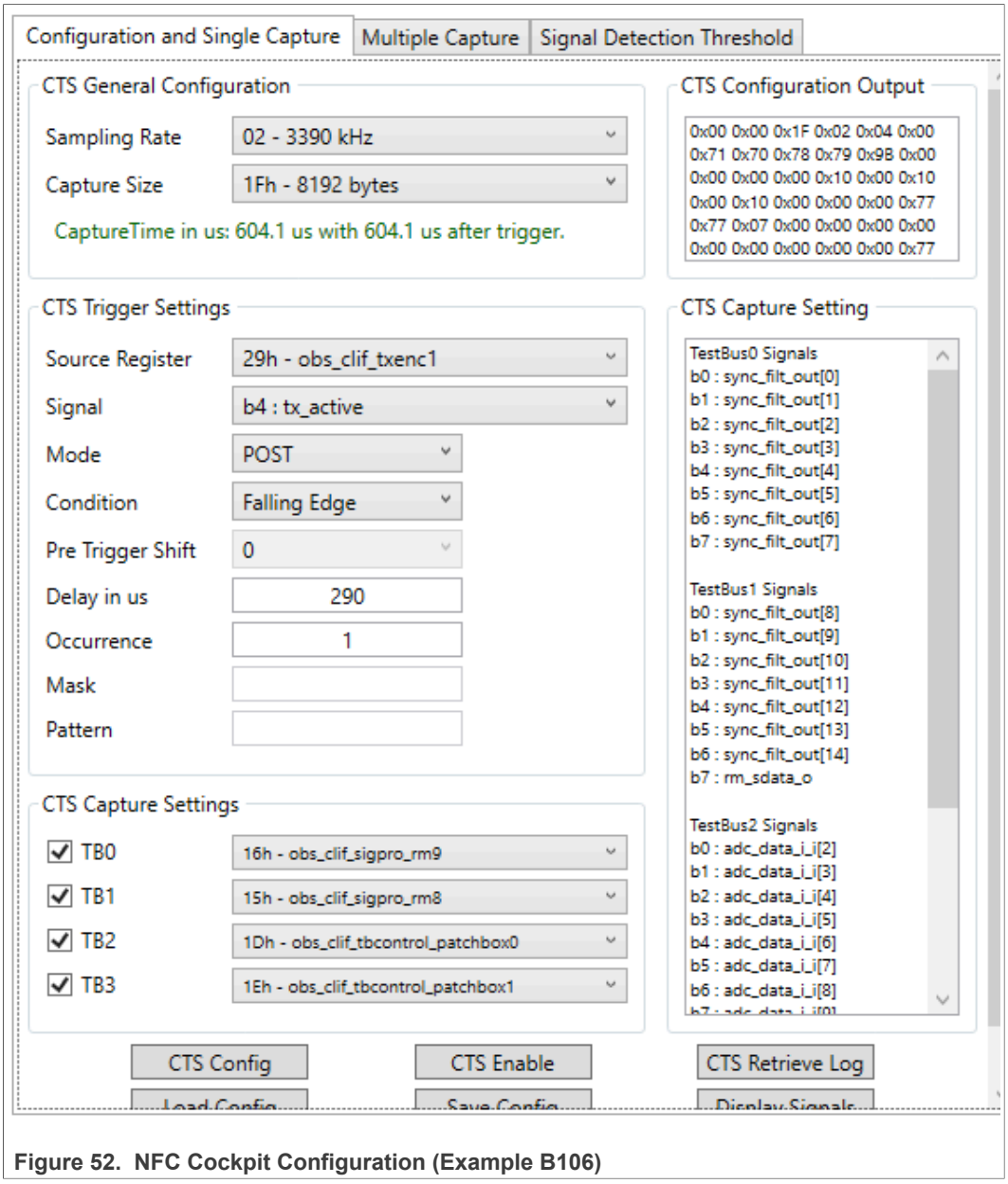


Figure 52. NFC Cockpit Configuration (Example B106)

The captured data is shown in Figure 53. It can be seen that with a sample rate of 3390 kHz, all 4 test buses and the full capture size used, the CTS can capture 600 µs.

Note: *Be aware that in this example the delay has been increased to 290 µs to just capture the SOF. Since in type B there is no fixed bit grid and a different timing compared to type A, a different card can respond at a different point of time. The amount of data inside the ATQB is much more than in the ATQA, so the complete response cannot be captured within 600 µs.*

The Figure 54 shows the complete ATQB. The sample rate has been changed to 1695 kHz and the ADC_I test signal has been disabled. So in total the capture takes 2400 µs, which includes the full ATQB.

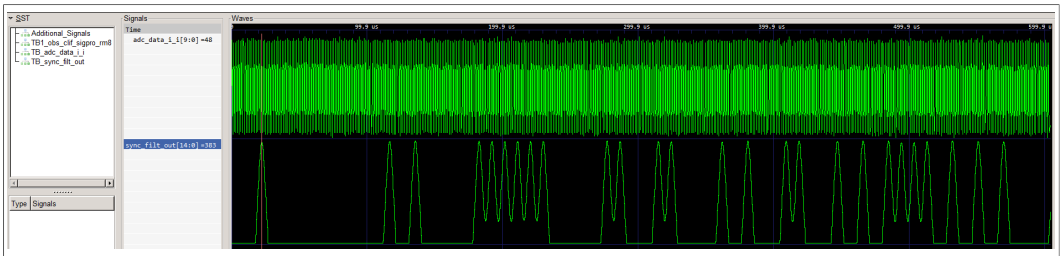


Figure 53. B106 example capture

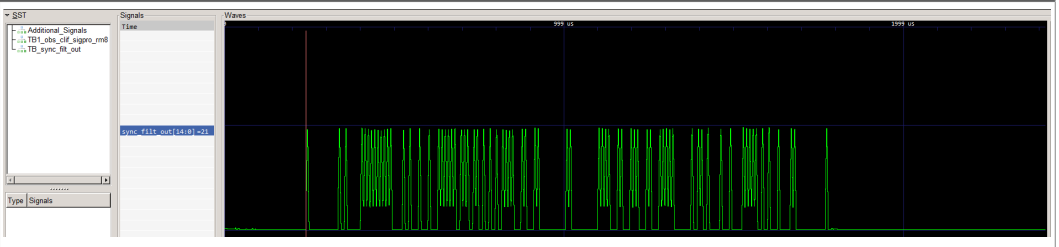


Figure 54. B106 example capture 2

6 References

[1] AN12549 PN5190 antenna design guide
[2] PN5190 data sheet
[3] OT6824.zip

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

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

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

7.3 Licenses

Purchase of NXP ICs with NFC technology — Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

7.4 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

MIFARE — is a trademark of NXP B.V.

MIFARE Classic — is a trademark of NXP B.V.

Tables

Figures

Contents

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

Date of release: 25 October 2022
Document identifier: AN12551