# UG10025

## NXP NFC Cockpit

**Rev. 1.0 — 10 May 2023** **User guide**

**Document Information**

| Information | Content |
|---|---|
| Keywords | NxpNfcCockpit |
| Abstract | This document contains the user guidance for NXP NFC Cockpit. |

# Revision history

**Revision history**

| Rev | Date | Description |
|---|---|---|
| v.1.0 | 20230510 | First official released version |



© 2023 NXP B.V. All rights reserved.

# 1 Introduction to NxpNfcCockpit

This document describes what is *NxpNfcCockpit* and how to use the features provided by this tool.

# 2 Installation of NxpNfcCockpit

This section describes how to install *NxpNfcCockpit*. As a part of the installation the following items are installed:

- The main executable and its supported DLLs
- Binaries required for PN7462AU
- Binaries required for PN7640
- Binaries required for PN7642
- Binaries required for LPC1769 when connected to CLRC663
- Binaries required for LPC1769 when connected to PN5180
- Binaries required for LPC1769 when connected to PN5190
- Binaries required for K8x when connected to PN5190
- VCOM Drivers for PC
- Reference XMLs for EEPROM layout mapping used by the application
- Configuration file to be modified by user on need basis
- Reference files for RxMatrix
- Reference scripts for scripting
- Test application to check porting of secondary firmware on users platform

## 2.1 Installation steps

***Attention:*** *Since NxpNfcCockpit installer also installs drivers for VCOM Connection to PN7462/PN7640/RC663/PN5180/PN5190/PN7642, administrative privileges are needed for the installation.*

- Run the installer as an administrator.
- Follow the steps as shown below.
- Read through the license agreement.



© 2023 NXP B.V. All rights reserved.

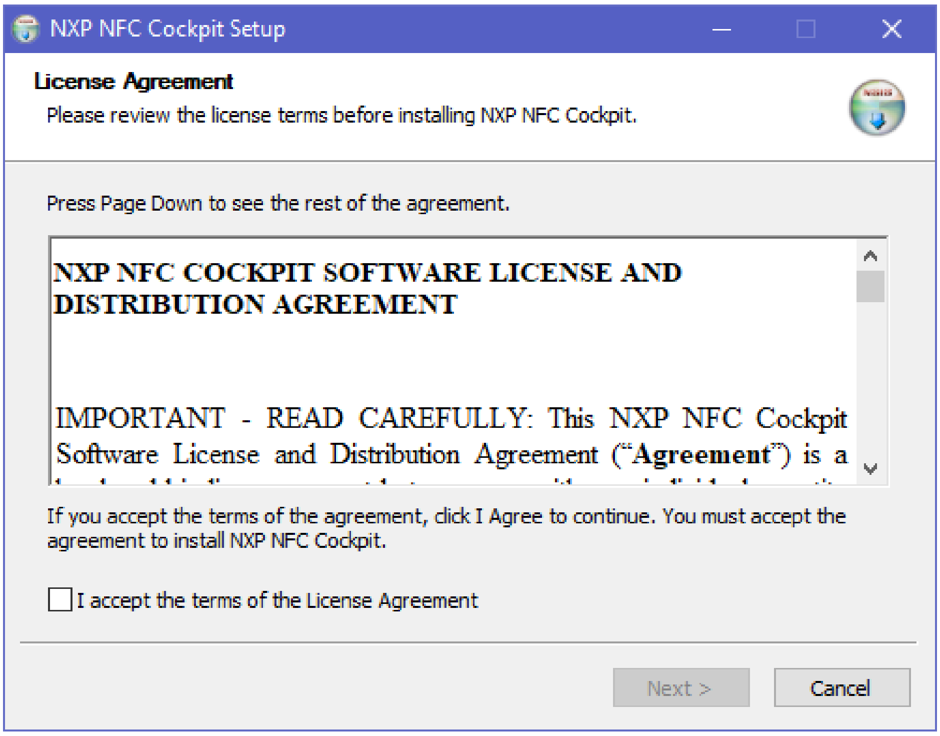



Figure 1. License Agreement part 1

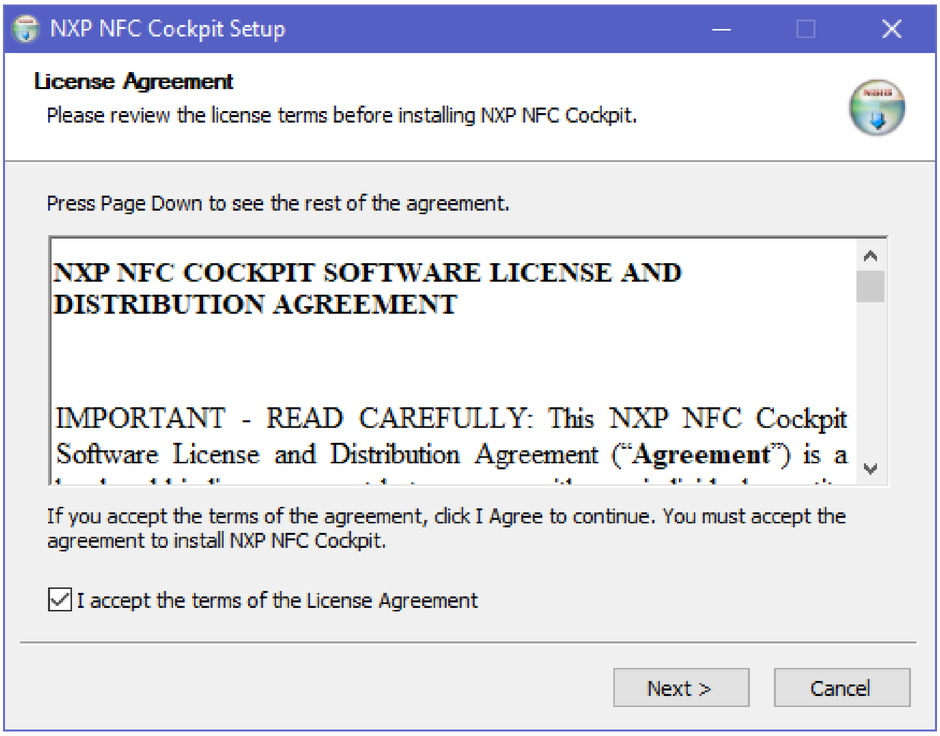



Figure 2. Licence Agreement part 2



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

- Select the optional components available as part of installer. Automatic Waveform Generator (AWG), which is used to control NI Devices. Enable option if NI Tools are installed. To install NI tools, refer to [PN5180 Evaluation board Quick Start Guide](#) .

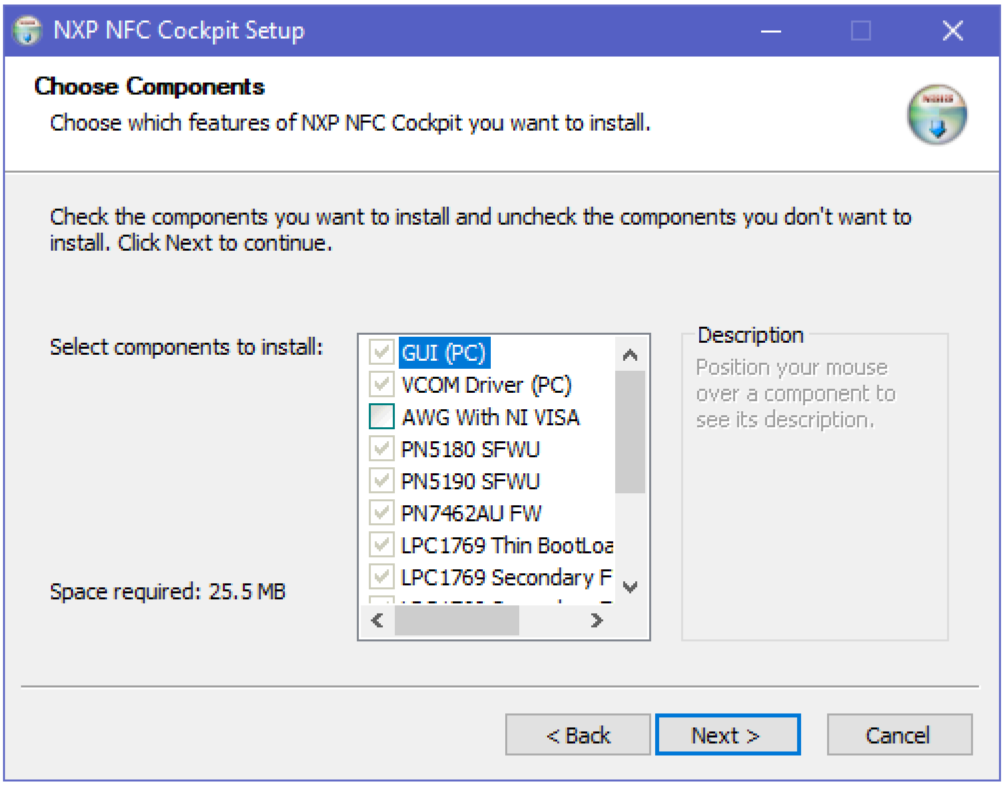



**Figure 3. Component Selection**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

- Select the installation folder.

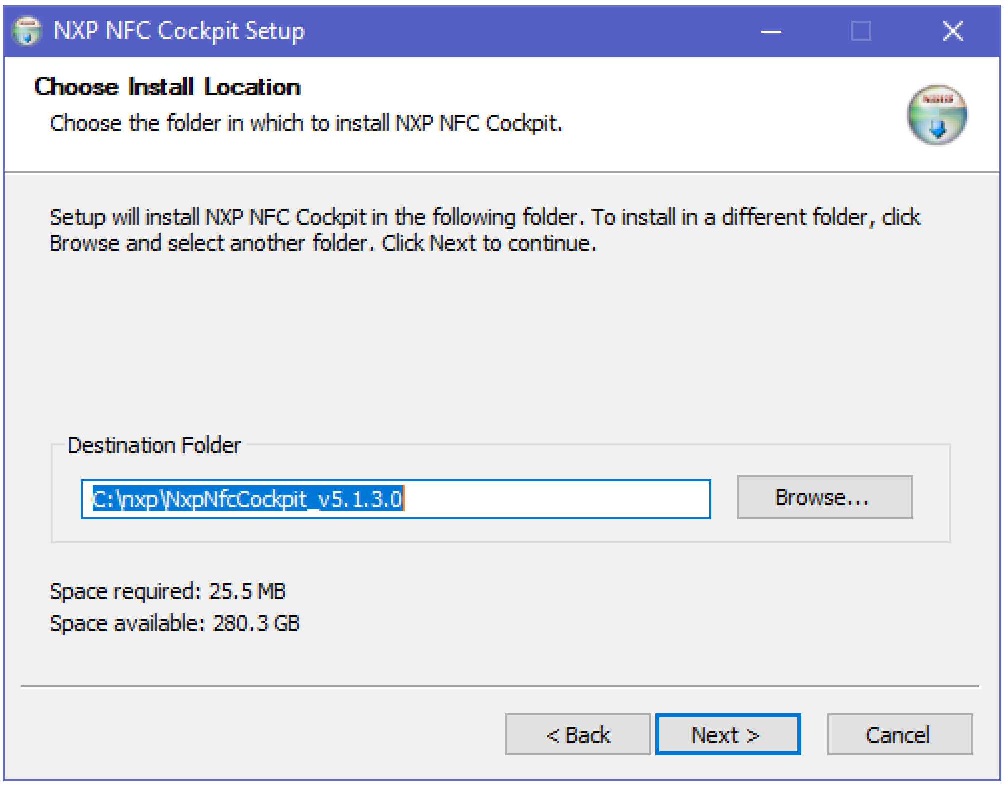



Figure 4. Installation Location


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

- Create shortcuts on start menu. As part of installation, the installer creates shortcut to Datasheets, User Manuals of the supported products.

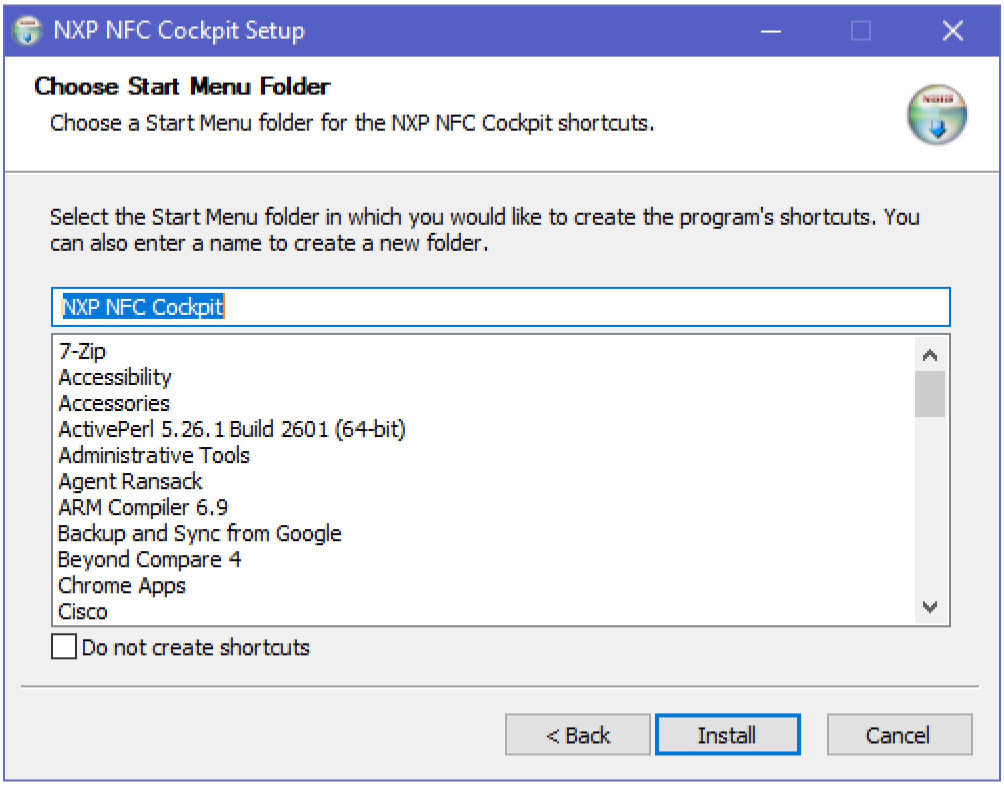



Figure 5. Create shortcut

- Driver Installation pop-up. Check Always trust software and Click Install.

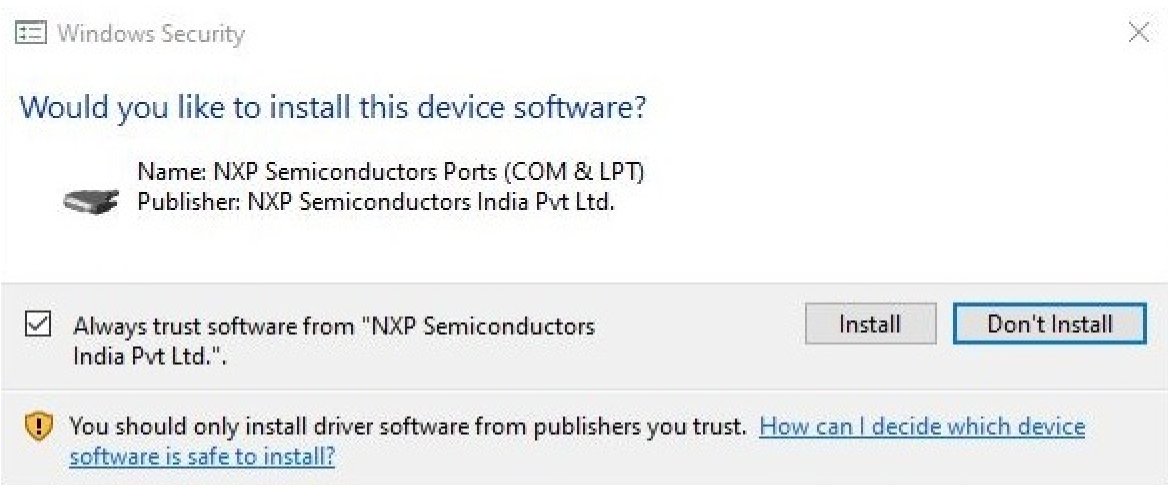



Figure 6. Required trusted software



© 2023 NXP B.V. All rights reserved.

- Close the installer on completion of the process.

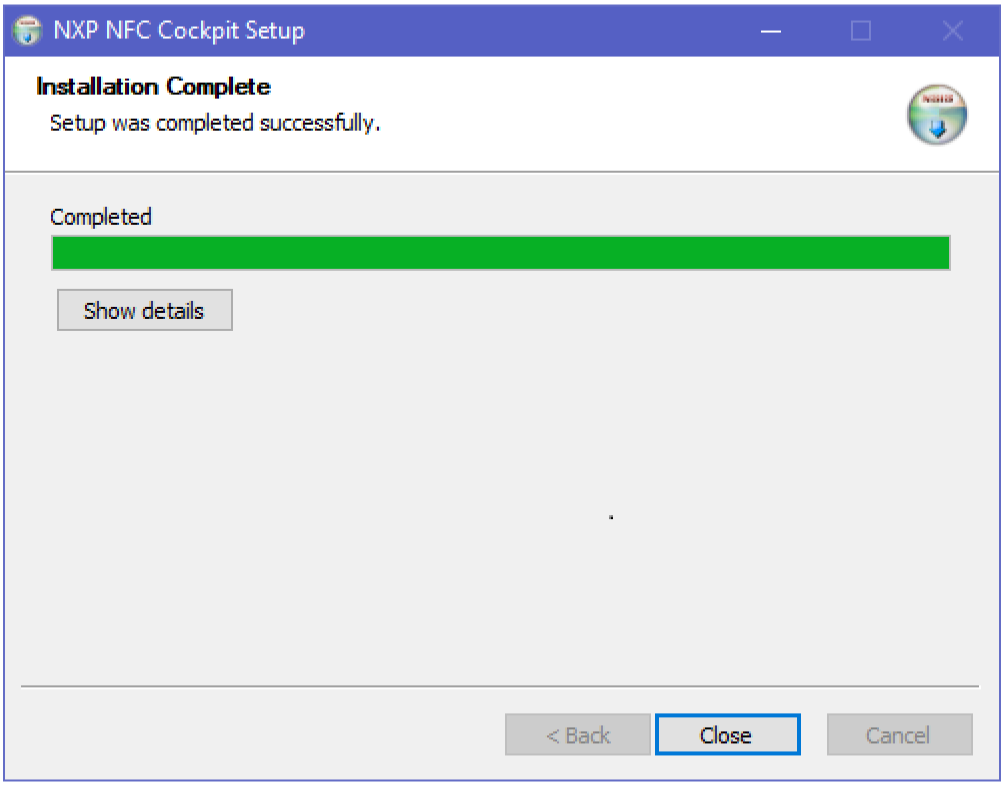



**Figure 7. Installation successful**

## 2.2 NfcCockpit Driver Installation

This section describes the installation of driver that is required by the *NxpNfcCockpit* tool to communicate with the Microcontroller-Host. This procedure is helpful to install the drivers manually if it was not auto installed during setup installation or something went wrong while installation of drivers during setup installation.

***Note:***

- *The installation procedure is same for all the drivers.*
- *Drivers are located at*
  - ***NxpNfcCockpit Installation dir/VCOM/install_PN7462AU_vcom.bat****. For PN7462 Boards*
  - ***NxpNfcCockpit Installation dir/VCOM/install_PN76XX_vcom.bat****. For PN7640 and PN7642 Boards*
  - ***NxpNfcCockpit Installation dir/VCOM/install_vcom.bat****. For RC663, PN5180, PN5190 boards with LPC1769 as Host-Controller*
  - ***NxpNfcCockpit Installation dir/VCOM/install_vcom_k8x.bat****. For Pn5190 boards with K82 as Host-Controller*

**Steps for driver installation**.

- Navigate to VCOM directory.
- Right Click on the required batch file and Click Run as administrator.
- Click Yes for the elevated permissions.
- Check the Trust Software message and Click Install.
- Wait untill ***Press any key to continue...*** message is seen.



© 2023 NXP B.V. All rights reserved.

- If driver installed successfully ***Successfully installed the driver*** message will appear. Also ***Number successfully imported*** will be 1.

  If Success message is not seen, ***Failure message will be seen and Number Successfully installed will be 0***. Close the current terminal and re-run the steps.

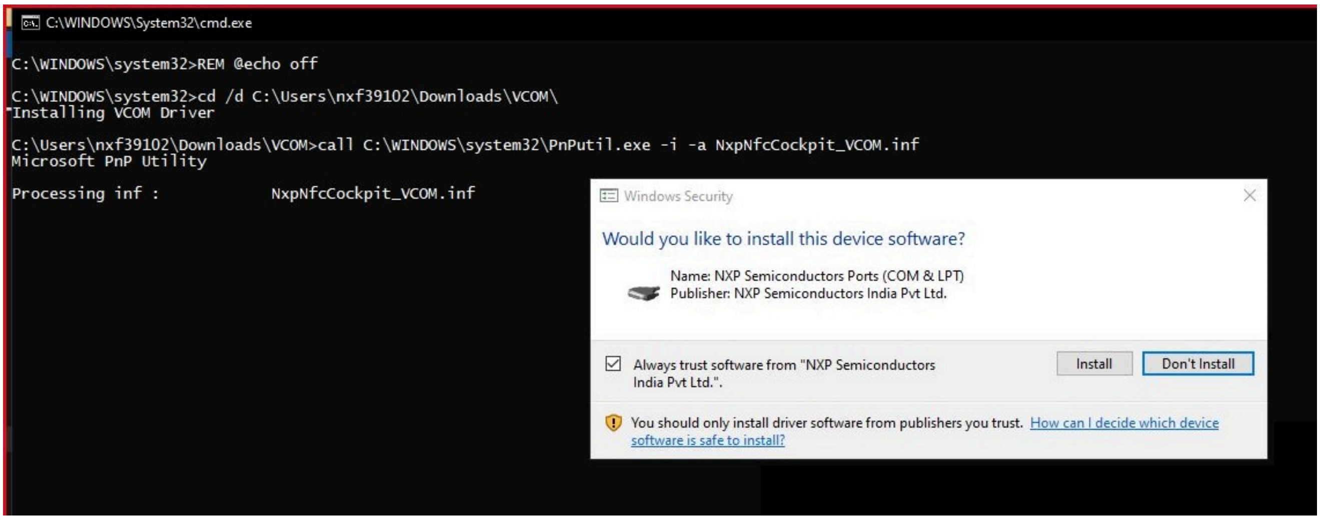



**Figure 8. Driver Installation Request**

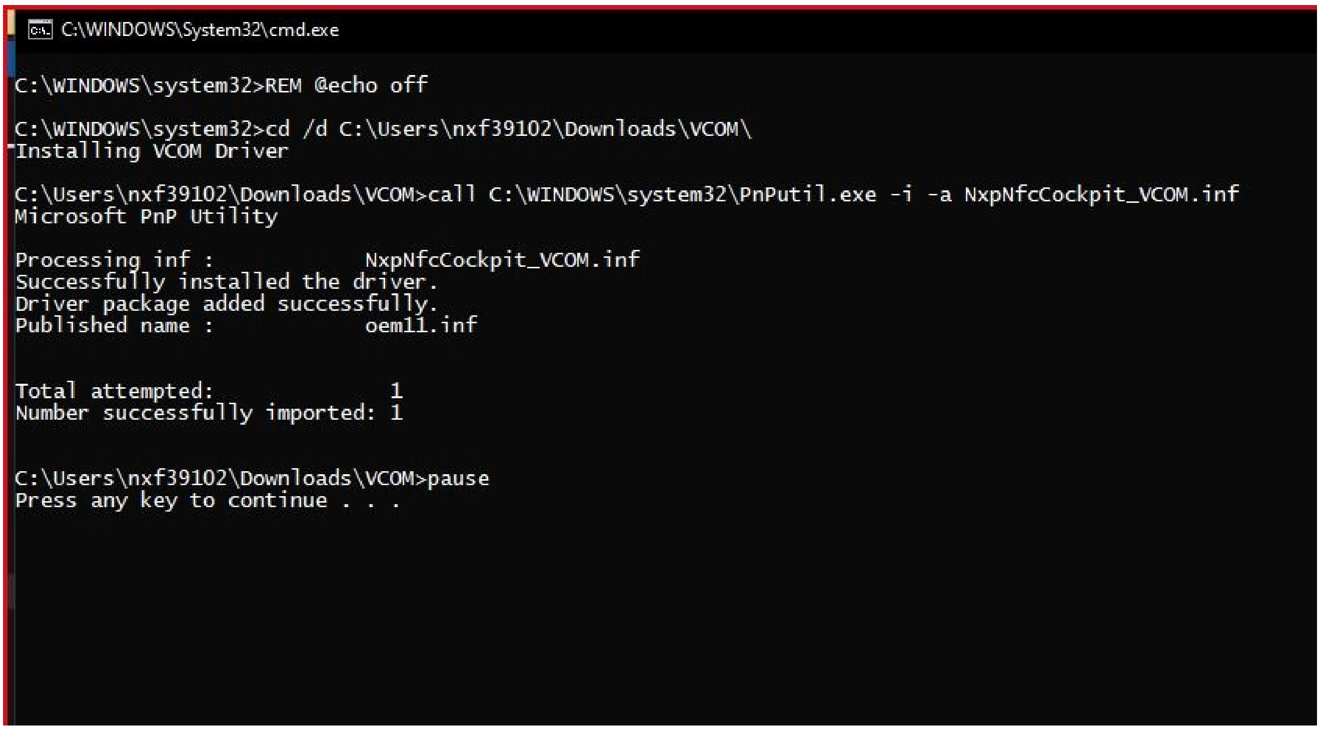



**Figure 9. Driver Completion**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

# 3 NfcCockpit firmware programming

This section describes the programming of firmware that is required by the NfcCockpit tool to communicate.

***Note:*** *This programming of firmware is not for reader IC. For programming reader IC, refer Secure firmware download.*

## 3.1 Setup: PN7462

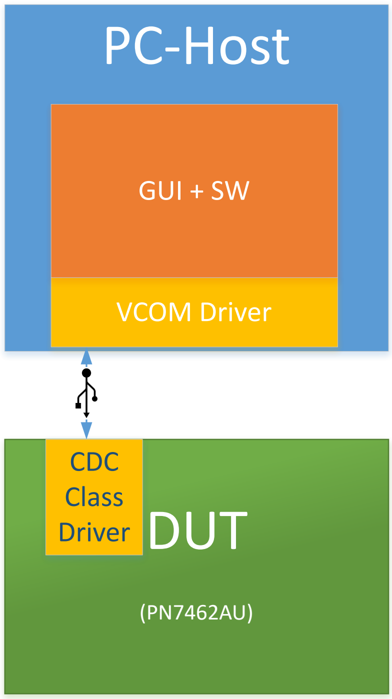



**Figure 10. PN7462**

Firmware can be updated by using primary downloader functionality (see [UM10883] ). Firmware binary is available in the NFC Cockpit installation folder: *"Select NfcCockpit Installation dir\firmware\PN7462AU"*.

***Note:*** *USB drivers needed for NFC Cockpit are part of the installation package and are automatically installed.*



© 2023 NXP B.V. All rights reserved.

## 3.2 Setup: PN7640 and PN7642

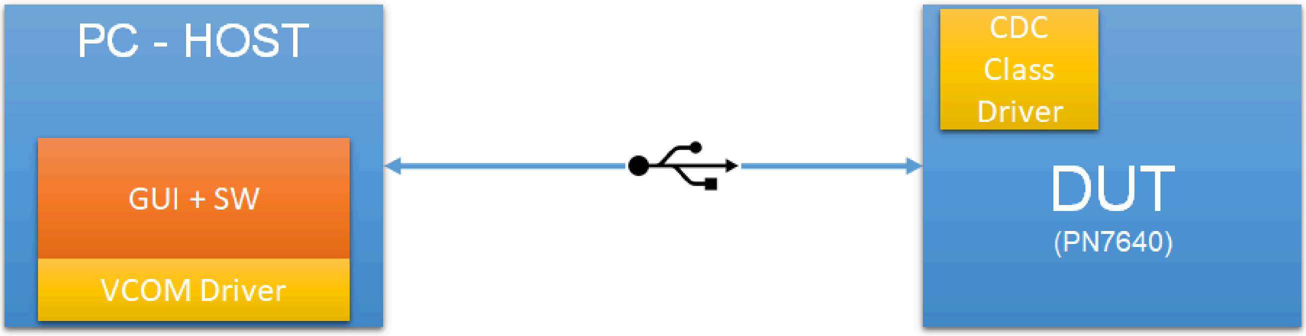
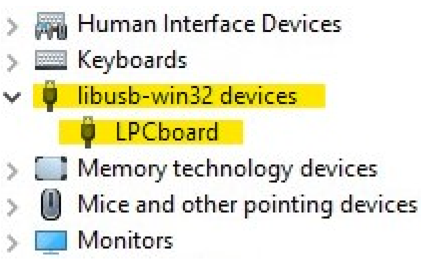



**Figure 11. PN7640**

Firmware can be updated by using primary downloader functionality (i.e using mass storage device). Firmware binary is available in the NFC Cockpit installation folder: *"Select NfcCockpit Installation dir\firmware\PN76XX"*.

***Note:*** *USB drivers needed for NFC Cockpit are part of the installation package and are automatically installed. If there is some issue with the driver installation refer to [NfcCockpit Driver Installation]*

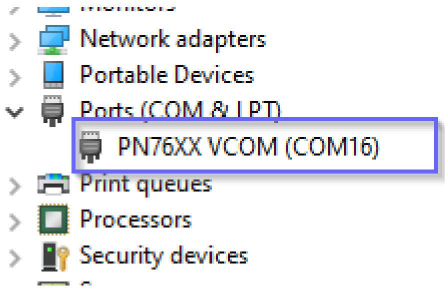
1. **Programming the Application**
   a. Remove all the connections (USB, Power) from PNEV76FAMA and LPC55s16 board
      i. With External Powered (Power from external Jack J9)
         - Connect USB TypeC to J5 (regular VCOM port).
         - Next connect External power to PNEV76FAMA Board.
         - Mass storage comes up with driver label as PN76XX_DL
      ii. With System Powered (Power from VCOM port, J5)
         - Connect USB TypeC to J5 (regular VCOM port).
         - Mass storage comes up with driver label as PN76XX_DL
   b. Replaces CRP_00.bin with the latest (*) Flash.bin.
   c. Once copied, the explorer closes automatically.
   d. Open the mass storage drive again and check for CRPSTA_0.BIN file. If this file is available, the nonsecure firmware has flashed properly.
2. **Verifying the program**
   a. Remove all the connections (USB, Power) from PNEV76FAMA and LPC55s16 board. Follow the next steps.
   b. Connect LPC55s16 J4 with micro-USB. If LPC55s16 application and driver are proper, then below item should be listed in device manager.




   c. Connect External Power supply to PNEV76FAMA board if board is configured to work with external power supply else proceed to next step.
   d. You should see connected device PN76XX VCOM in connected devices.



© 2023 NXP B.V. All rights reserved.




e. If COM-Port is not listed, then press the button SW3 (NFC_VEN).

***Note:***

1. *In either of the above cases, J6 (USB_VBUS) should be connected and J25 (DWL_REQ) should be removed.*
2. *LP55s16 connection while programming and execution.*
   a. *No connection should be made to LPC55s16 board while programming the application.*
   b. *LPC55s16 J4 should be connected first before executing the application.*

## 3.3 Setup: LPC1769 + (CLRC663 / PN5180 / PN5190), K82 + (PN5190)

NFC Cockpit requires a dedicated firmware running on the LPC1769. This firmware application implements CDC USB class device (VCOM). The NFC Cockpit directs commands to the VCOM port and dedicated firmware executes commands on the hardware level.

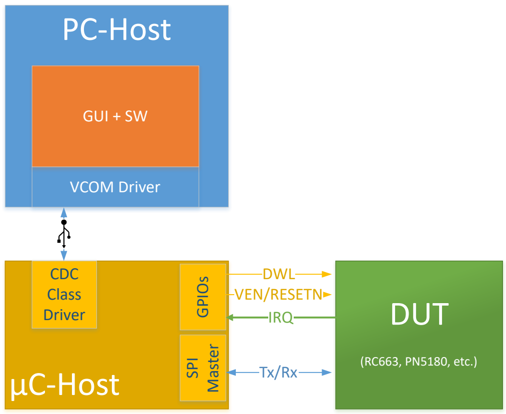



**Figure 12. LPC1769 + CLRC663, PN5180, and PN5190**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

### 3.3.1 MCUXpresso and LPCLink2 / JLink

***Firmware programming using MCUXpresso and LPCLink2 / JLink.***

- Select an LPC1769 Project in MCUXpresso IDE. Press the program button on MCUXpresso IDE.

**Figure 13. IC program button**

- MCUXpresso should start searching for a connected probe

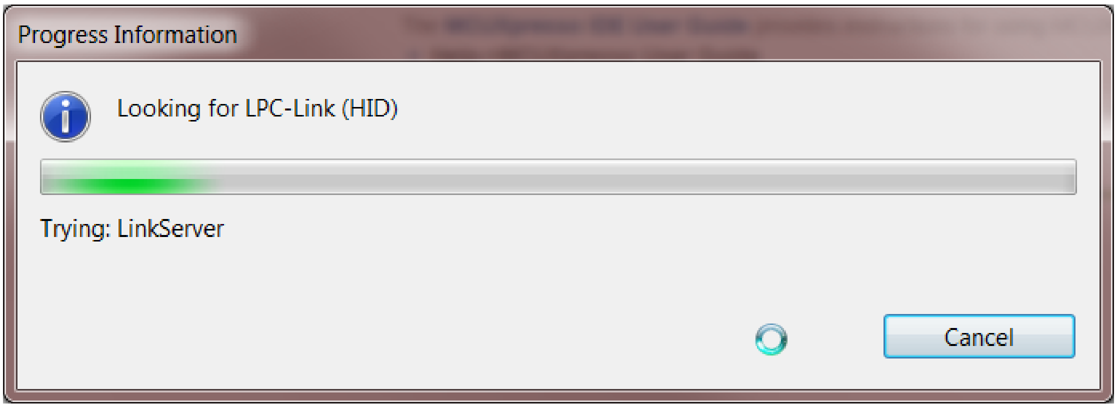



**Figure 14. Detection link probes**



© 2023 NXP B.V. All rights reserved.

- Select the connected LPCLink2 / JLink Probe

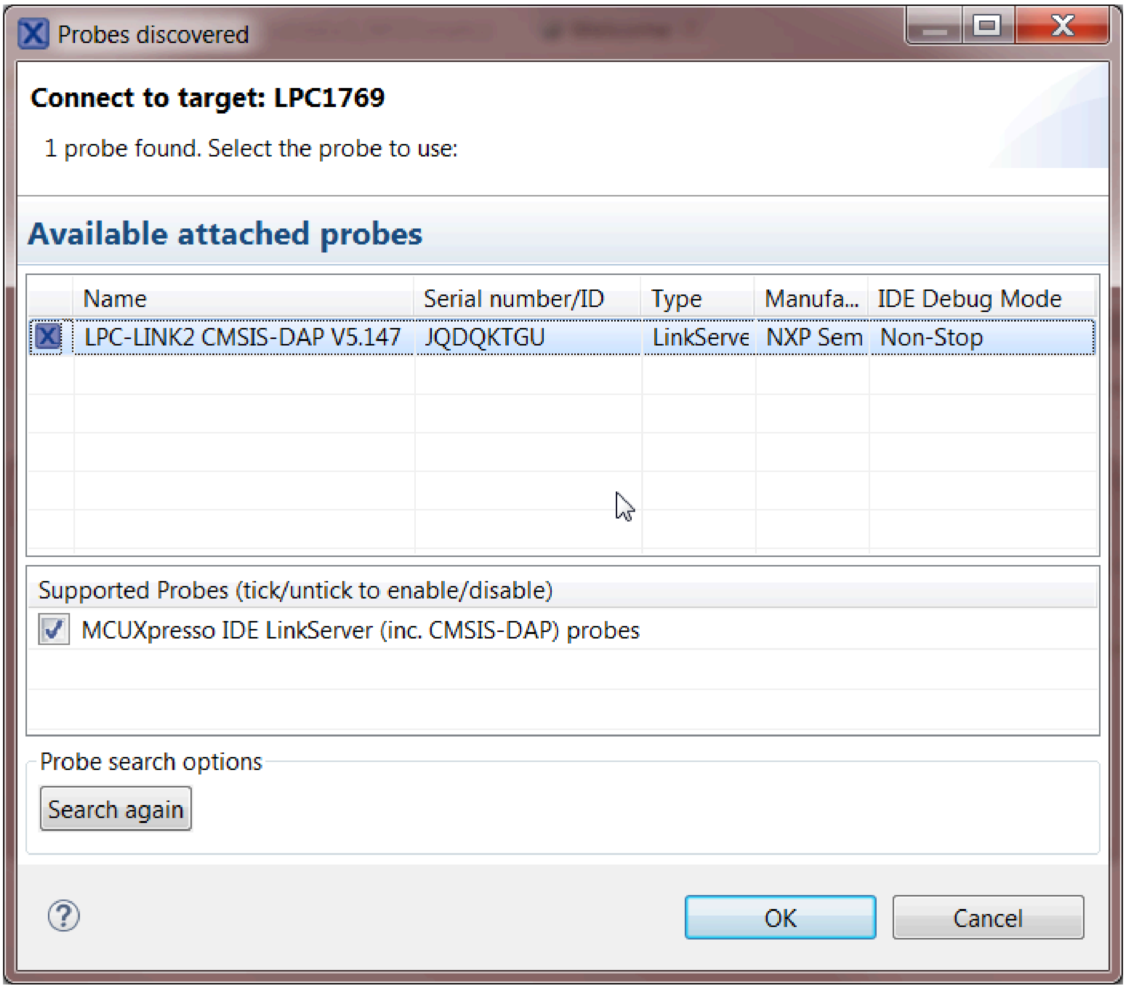



**Figure 15. Select link 2**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

- Select the binary file to be loaded.
  - For CLRC663 IC: Select *NxpNfcCockpit* Installation dir\firmware\Secondary_RC663\BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary.bin
  - For PN5180 IC: Select *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5180\BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary.bin
  - For PN5190 IC: Select
    - *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5190**LPC1769**\ BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary_FSDI_*.bin
    - *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5190**K8x**\ BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary_FSDI_*.bin

  where * is FSDI value. Current FSDI values are 8 and 10.

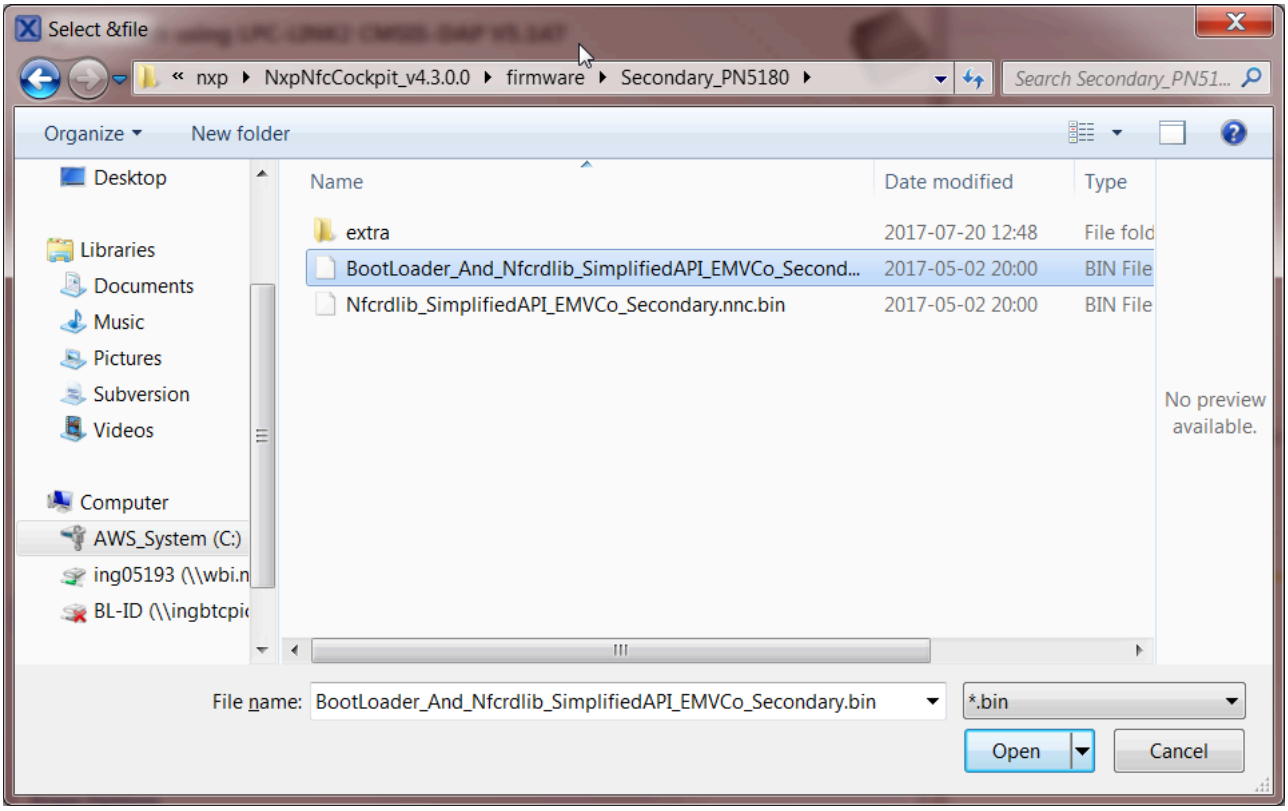



**Figure 16. Select binary**



© 2023 NXP B.V. All rights reserved.

- Set Base Address as 0x0

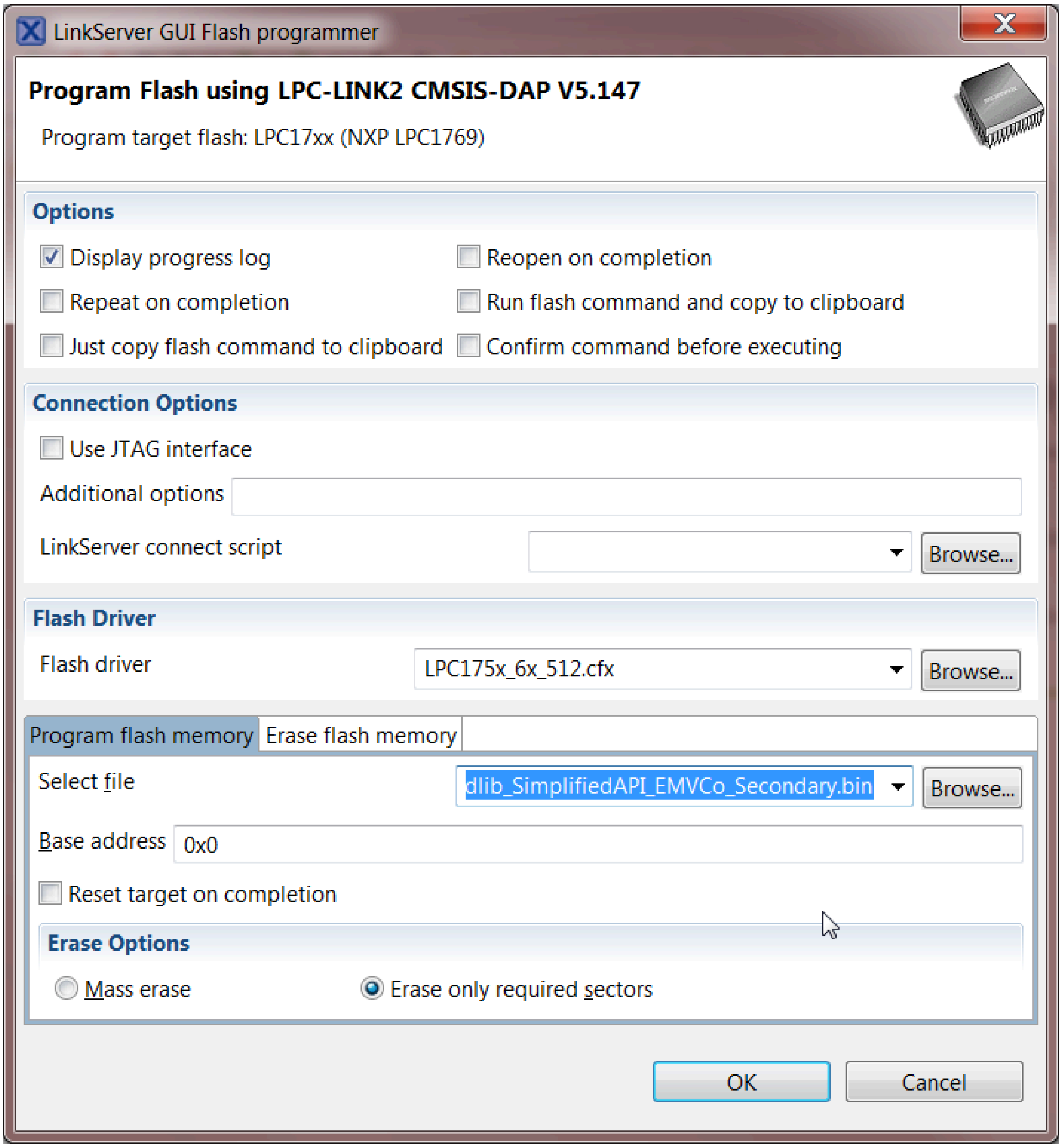



Figure 17.  Set base address



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

- Flash programming should start now

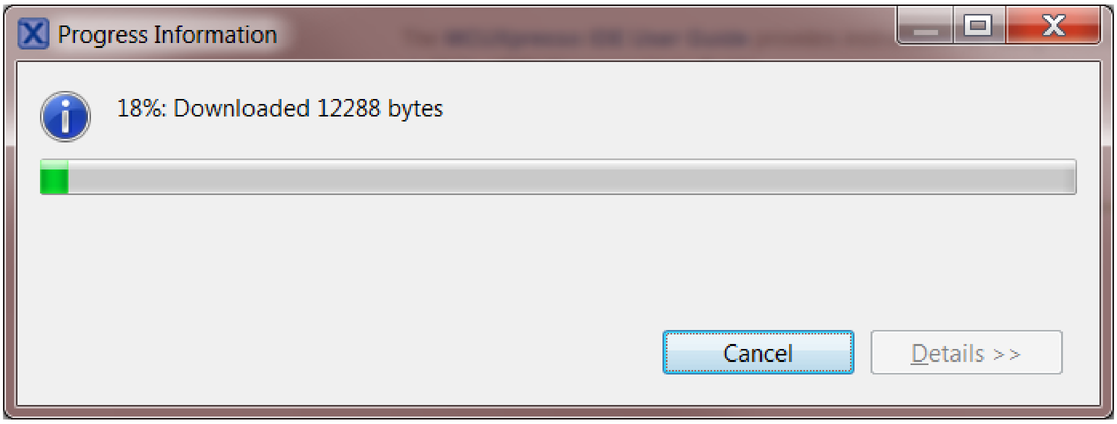



**Figure 18. Start programming**

- Once flash programming is successfully completed, reboot the board.

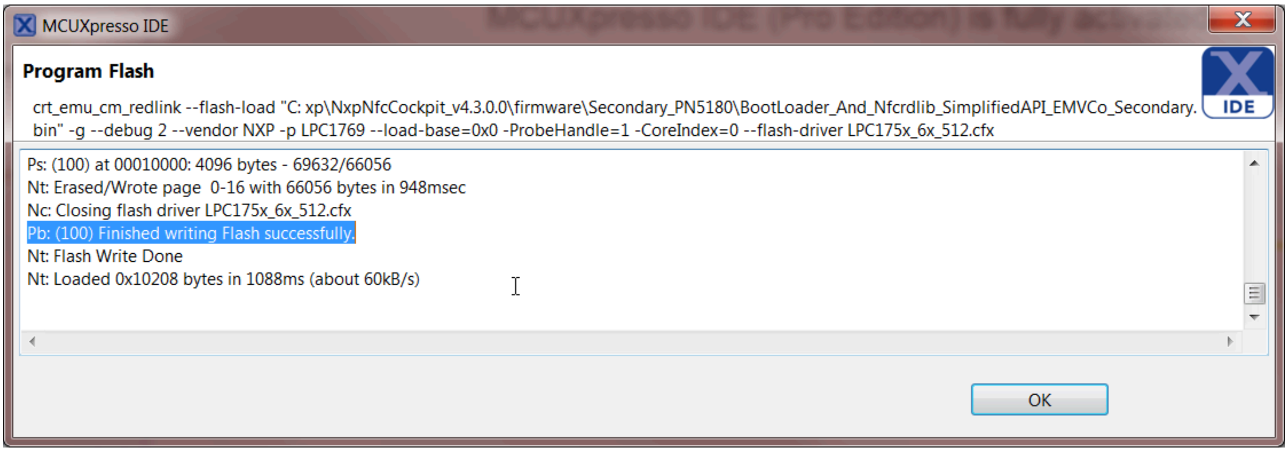



**Figure 19. Programmed successfully**



© 2023 NXP B.V. All rights reserved.

- You should see connected device NxpNfcCockpit VCOM in connected devices.

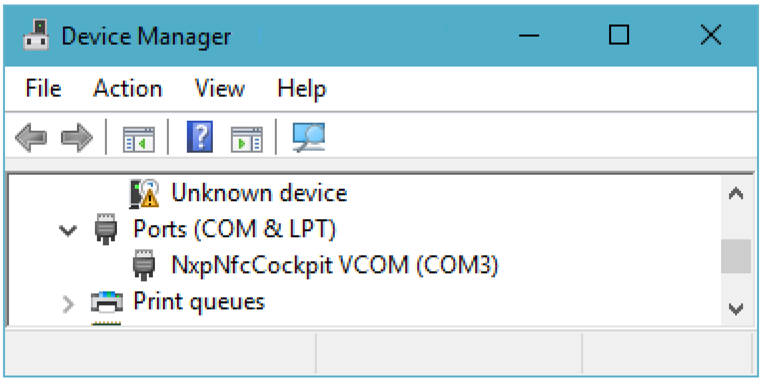



**Figure 20. Device manager view**

### 3.3.2 SEGGER's J-Flash Lite and JLink

***Firmware Programming using SEGGER's J-Flash Lite and JLink.***

- Open J-Flash Lite from SEGGER installation directory. Mostly it is in the below path. "C:\Program Files (x86)\SEGGER\JLink\JFlashLite.exe"
- Select LPC1769 / K82 as Device, Interface as SWD and Speed as 4000 KHz. Click **OK** after updating the required settings.
  To select the device click the button with three dots (...).
  – Device Selection

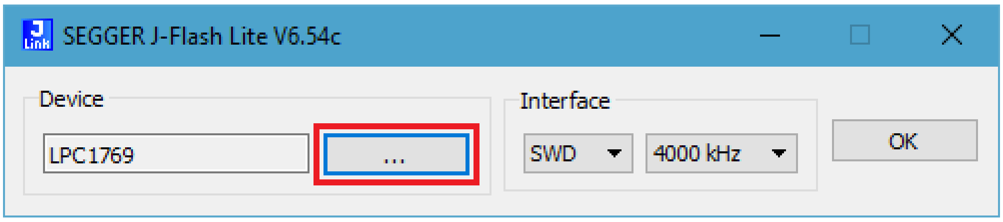



**Figure 21. select device**

  – Device Selection: LPC1769

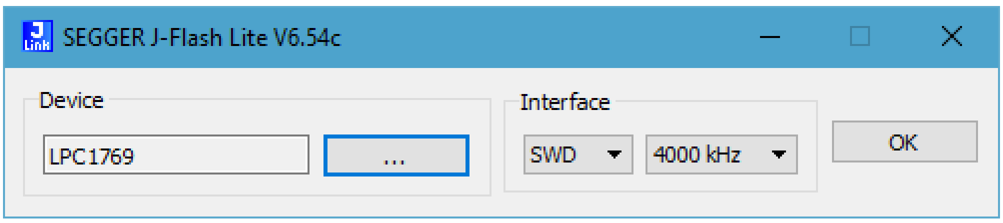



**Figure 22. LPC1769 and SWD speed**

  – Device Selection: K82



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

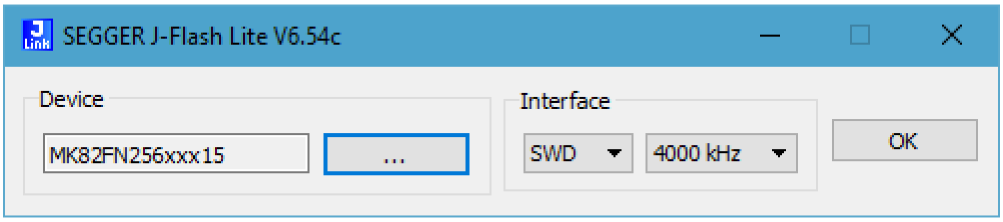



**Figure 23. MK82 and SWD speed**



© 2023 NXP B.V. All rights reserved.

- Set Prog. addr. (bin file only) as 0x00000000
- Select the binary file to be loaded.
  - For RC663 IC: Select *NxpNfcCockpit* Installation dir\firmware\Secondary_RC663\BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary.bin
  - For PN5180 IC: Select *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5180\BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary.bin
  - For PN5190 IC: Select
    - *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5190**LPC1769**\ BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary_FSDI_*.bin
    - *NxpNfcCockpit* Installation dir\firmware\Secondary_PN5190**K8x**\ BootLoader_And_Nfcrdlib_SimplifiedAPI_EMVCo_Secondary_FSDI_*.bin

  where * is FSDI value. Current FSDI values are 8 and 10.

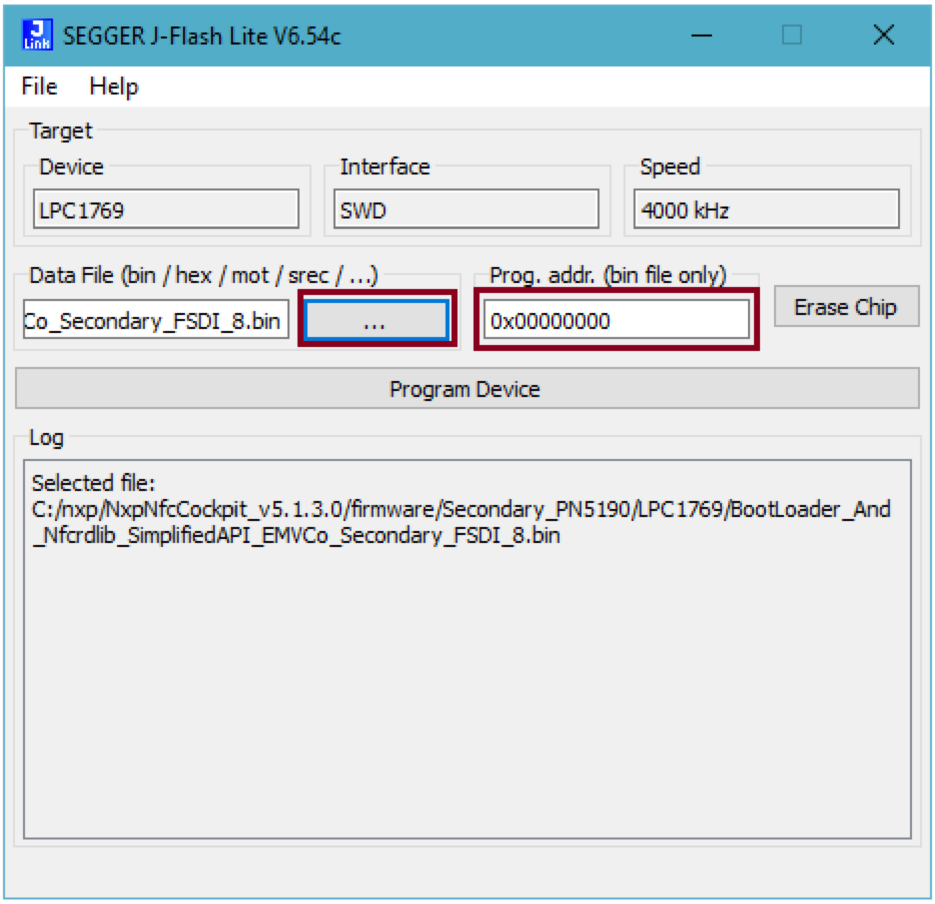



**Figure 24. Select binary to flash**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

- Click Program Device. Once flash programming is successfully completed, reboot the board.

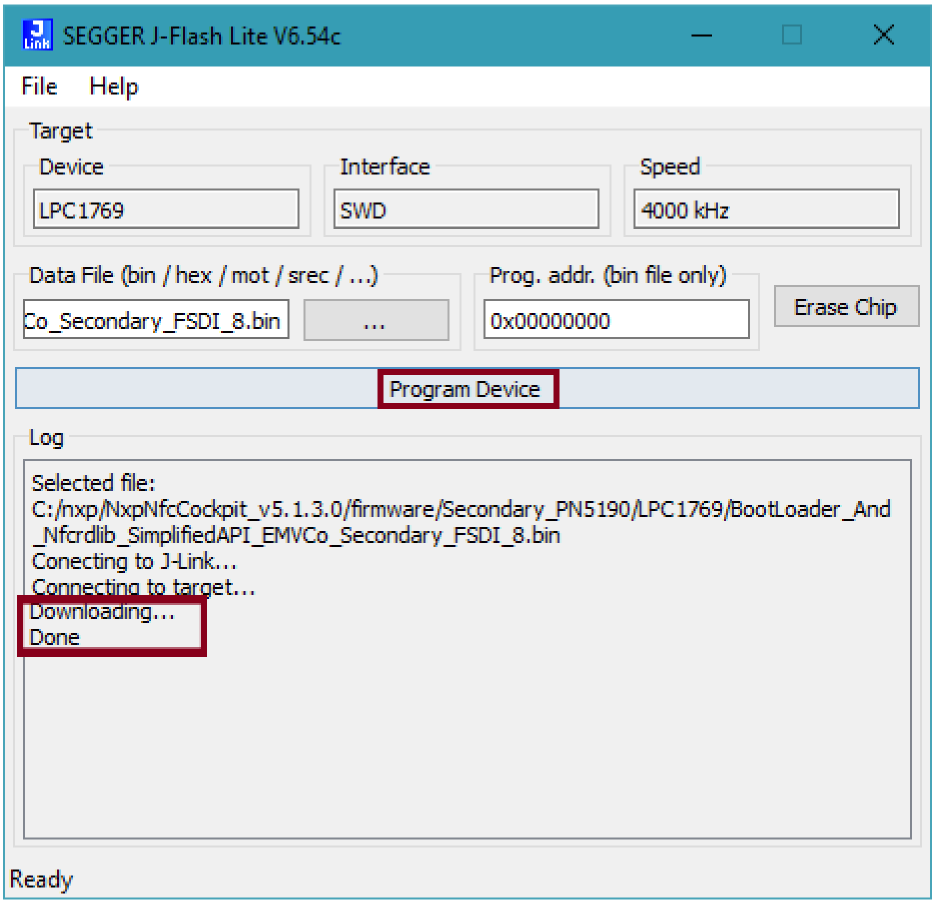



Figure 25. Programmed successfully

- You should see connected device NxpNfcCockpit VCOM in connected devices.

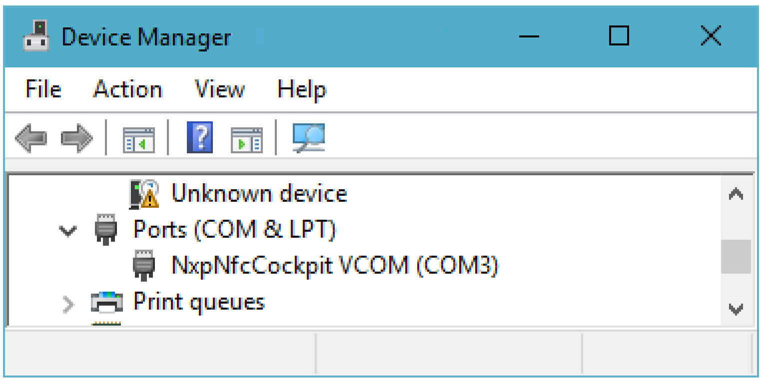



Figure 26. Device manager view



© 2023 NXP B.V. All rights reserved.

# 4 Features of NxpNfcCockpit

## 4.1 Registers manipulation

The NFC Cockpit allows the reading and writing of registers of the IC. Selecting a register reads and shows the hexadecimal value as well as the corresponding bit values. The input allows to change each bit separately as well as writing hexadecimal values. Writing back the value changes the register of the IC. On *mouse over*, the application displays a short description of the register parts. Accessing the registers using *NxpNfcCockpit* is same for all the supported Reader ICs.

***Note:*** *Some register content cannot be changed manually (read only) and some content might be overwritten by the firmware.*

**Registers** allow user to modify the value of registers in multiple ways. It can be modified at bit level using masking or as a complete 32bit/8bit value.

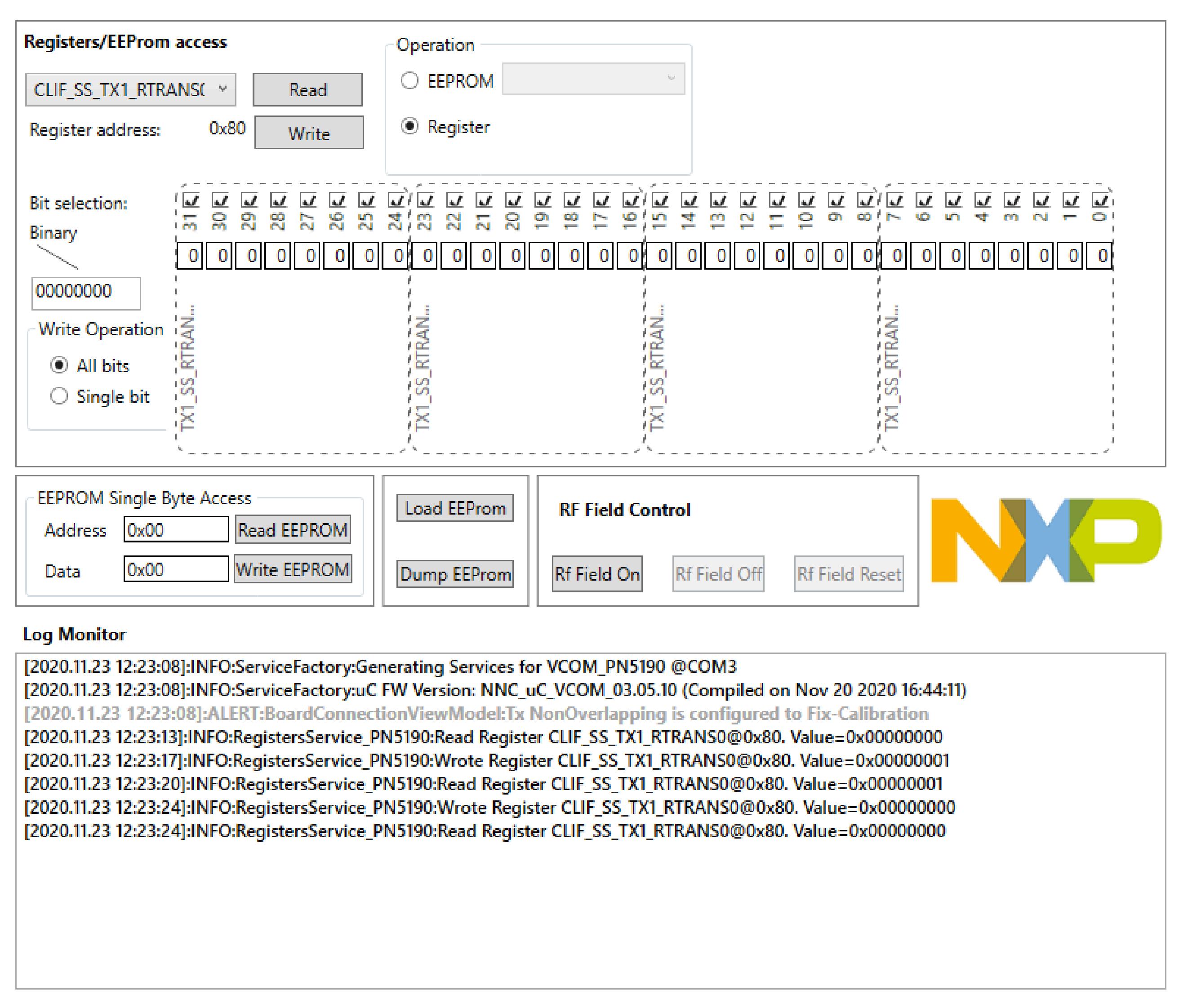



**Figure 27. Write and Read register**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

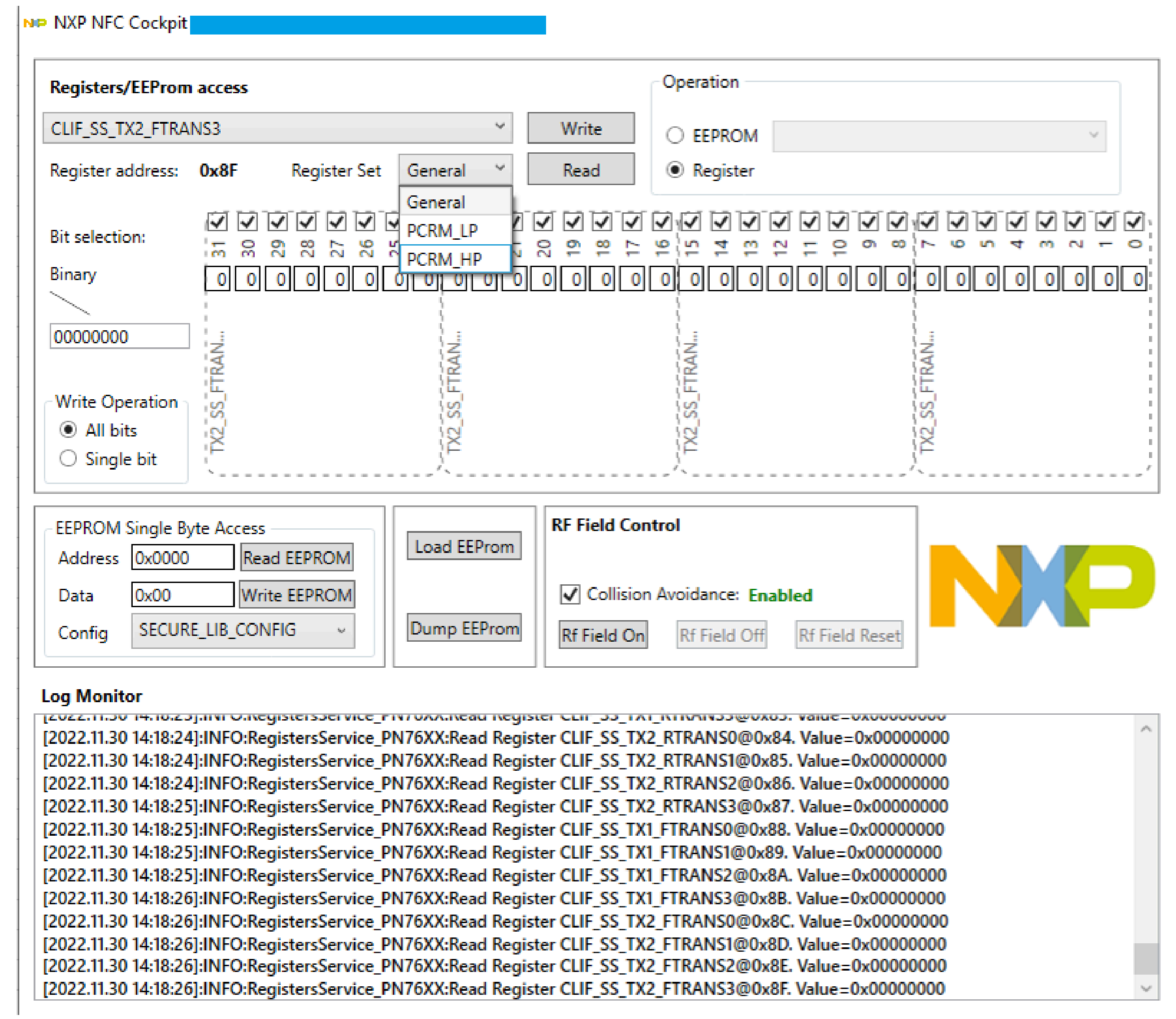


Figure 28. Write and Read register RegisterSet

## 4.2 EEPROM manipulation and management

The *NxpNfcCockpit* allows four options for accessing EEPROM

- Read EEPROM
- Write EEPROM
- Dump EEPROM
- Load EEPROM

***Note:*** *All exported format cannot be imported back by the NxpNfcCockpit.*

### 4.2.1 Read

Reads a single byte from EEPROM using address. The address varies based on the reader IC. The address is 32 bits long data while value will be on 8 bits.



© 2023 NXP B.V. All rights reserved.

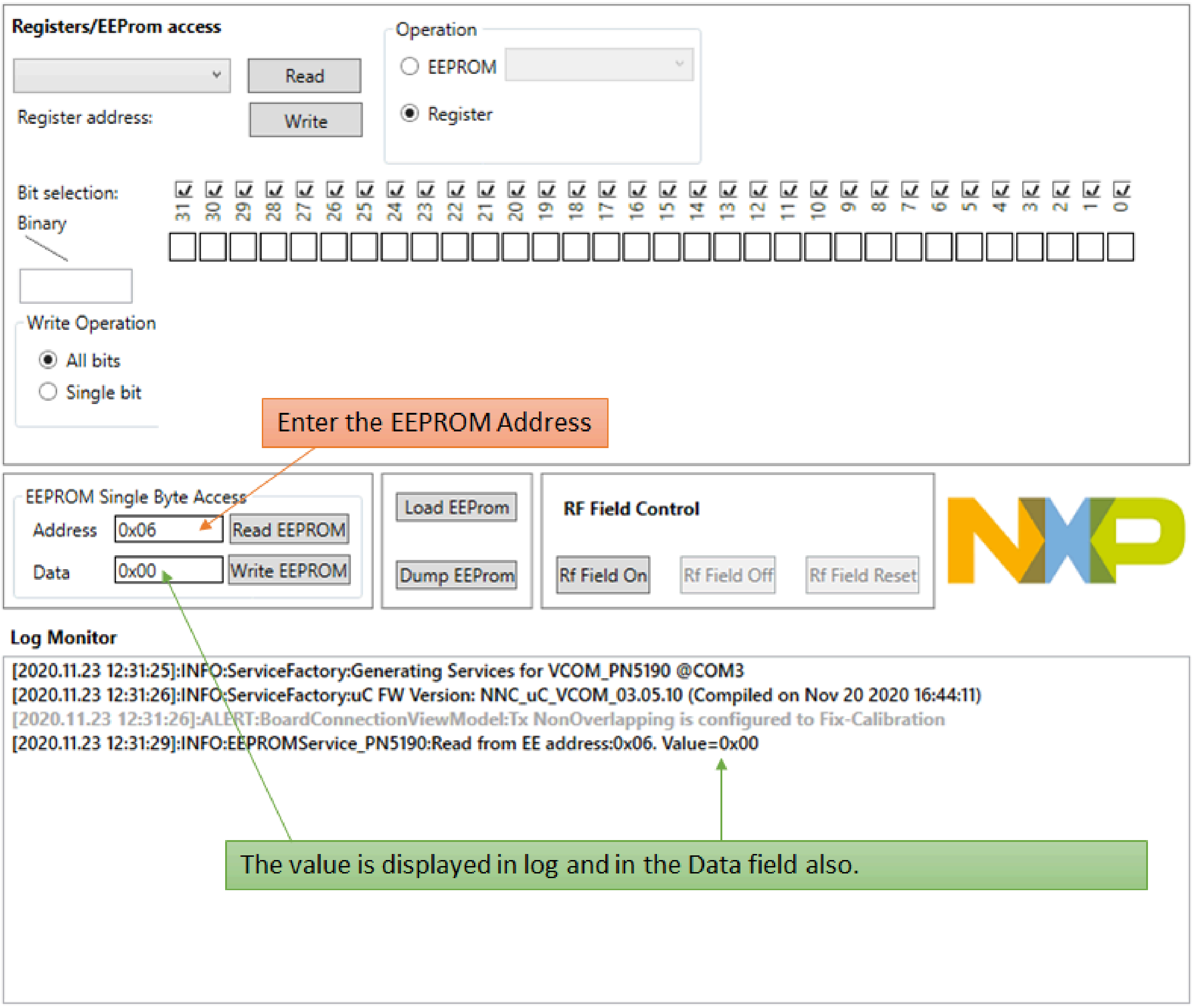



**Figure 29. EEPROM Read**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

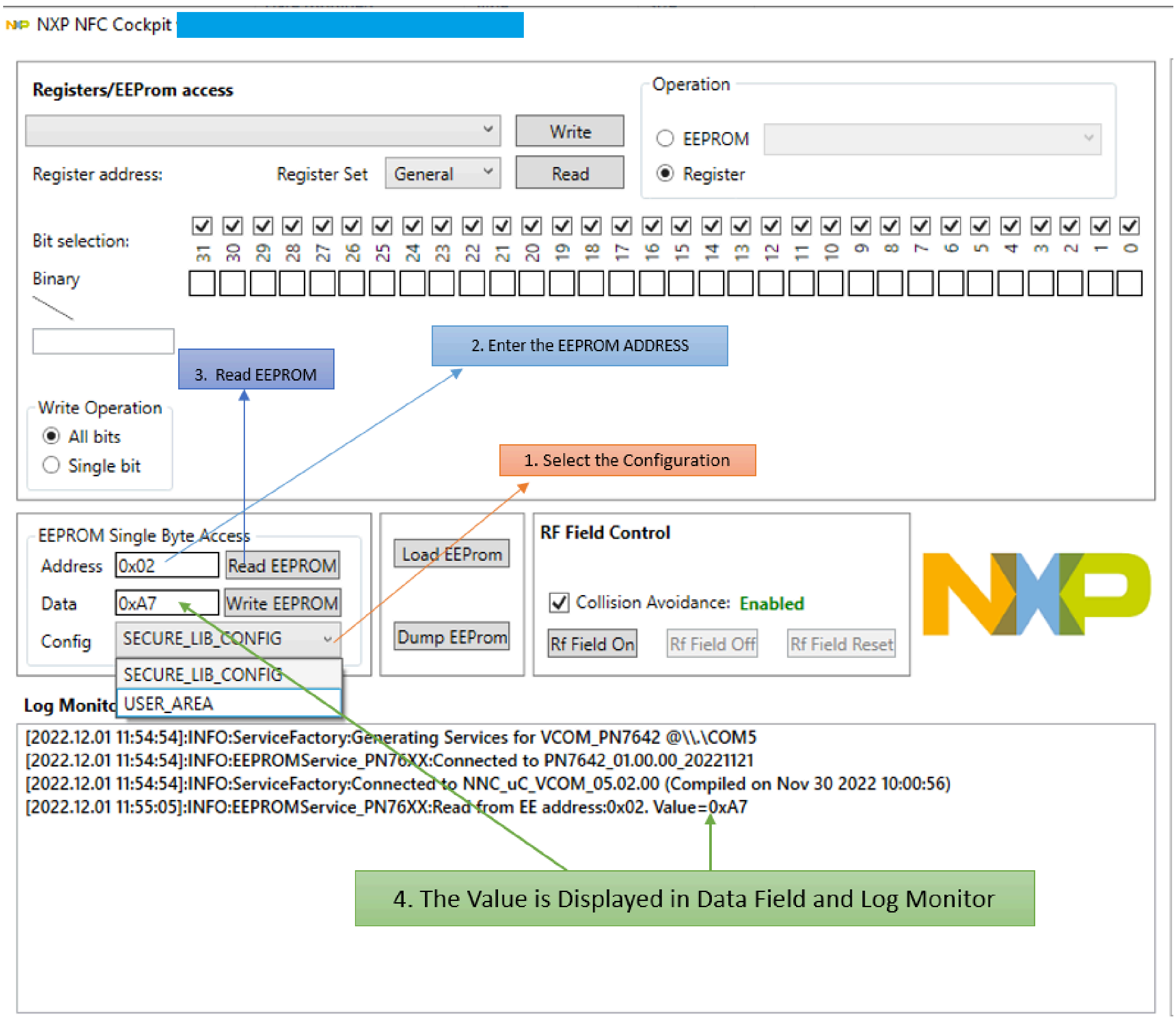



**Figure 30. EEPROM Read Config**



© 2023 NXP B.V. All rights reserved.

### 4.2.2 Write

Writes a single byte to EEPROM using address. The address varies based on the reader IC. The address is 32 bits long data while value will be on 8 bits.

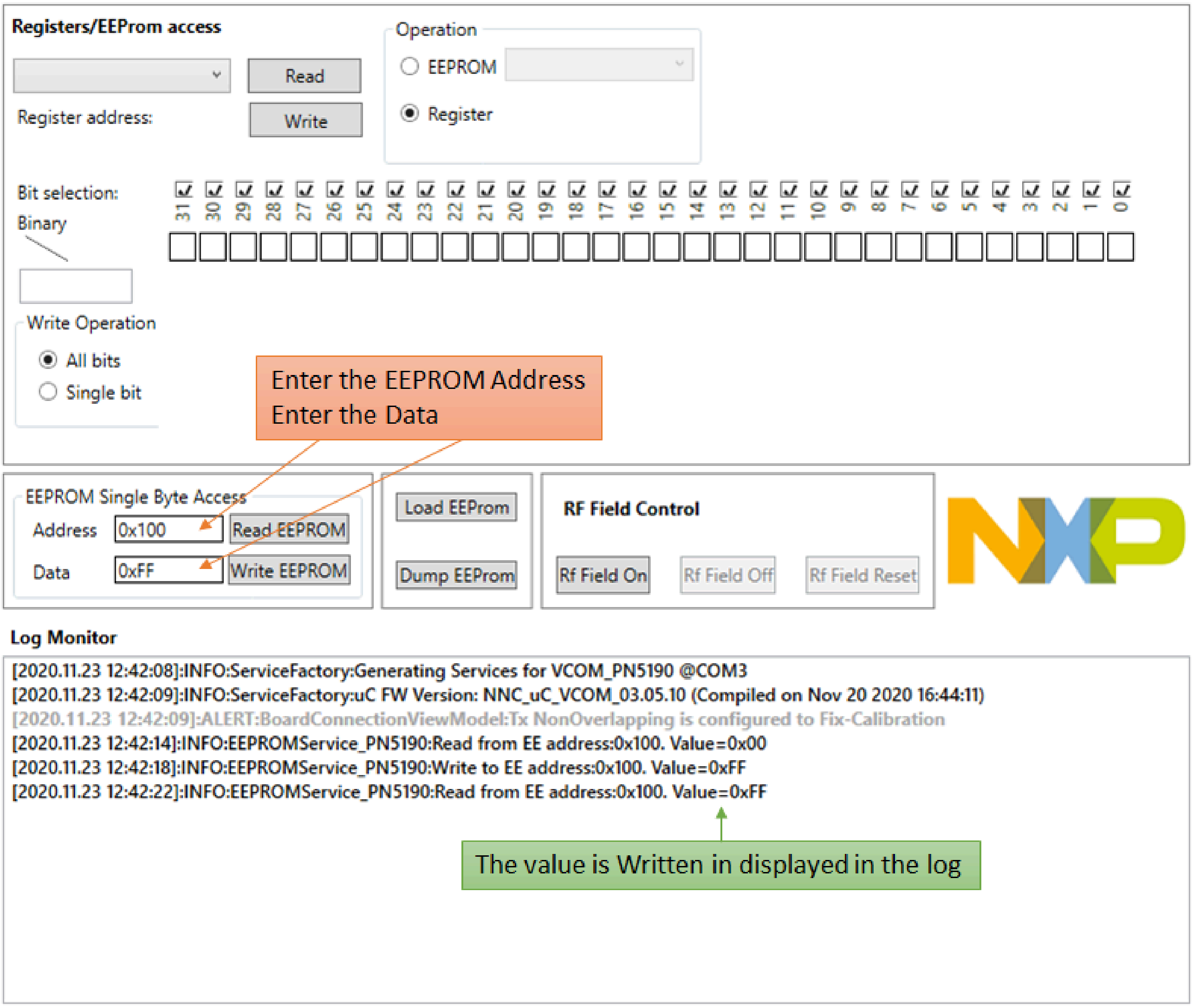



**Figure 31. EEPROM Write**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

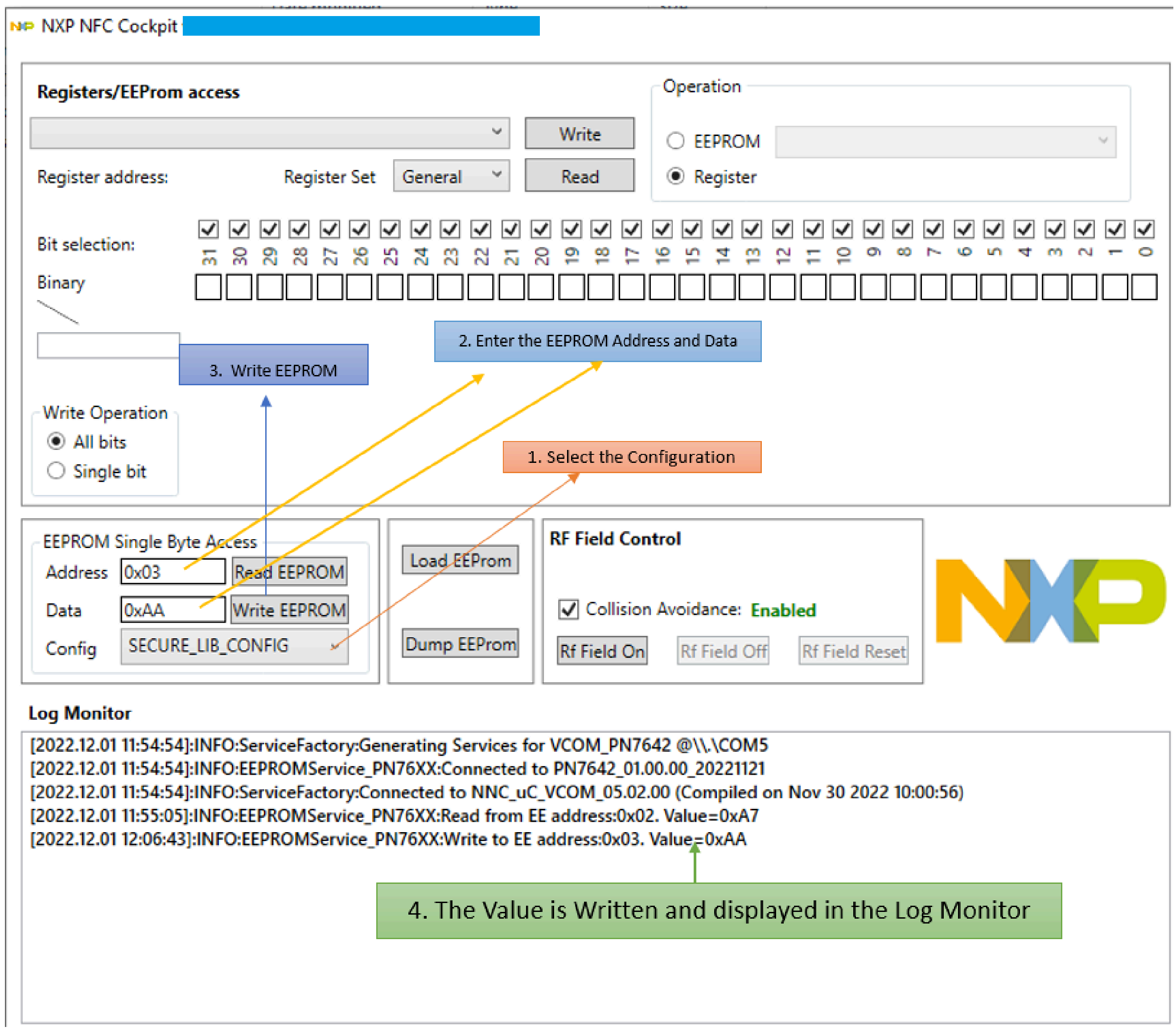



**Figure 32. EEPROM Write Config**



© 2023 NXP B.V. All rights reserved.

### 4.2.3 Dump

Stores the complete user area of the EEPROM into a file that can be stored on PC. This can be used to generate a backup of all settings or to transfer optimized settings onto another board or into own software. The EEPROM contents can be dumped to either .xml file for .h file.

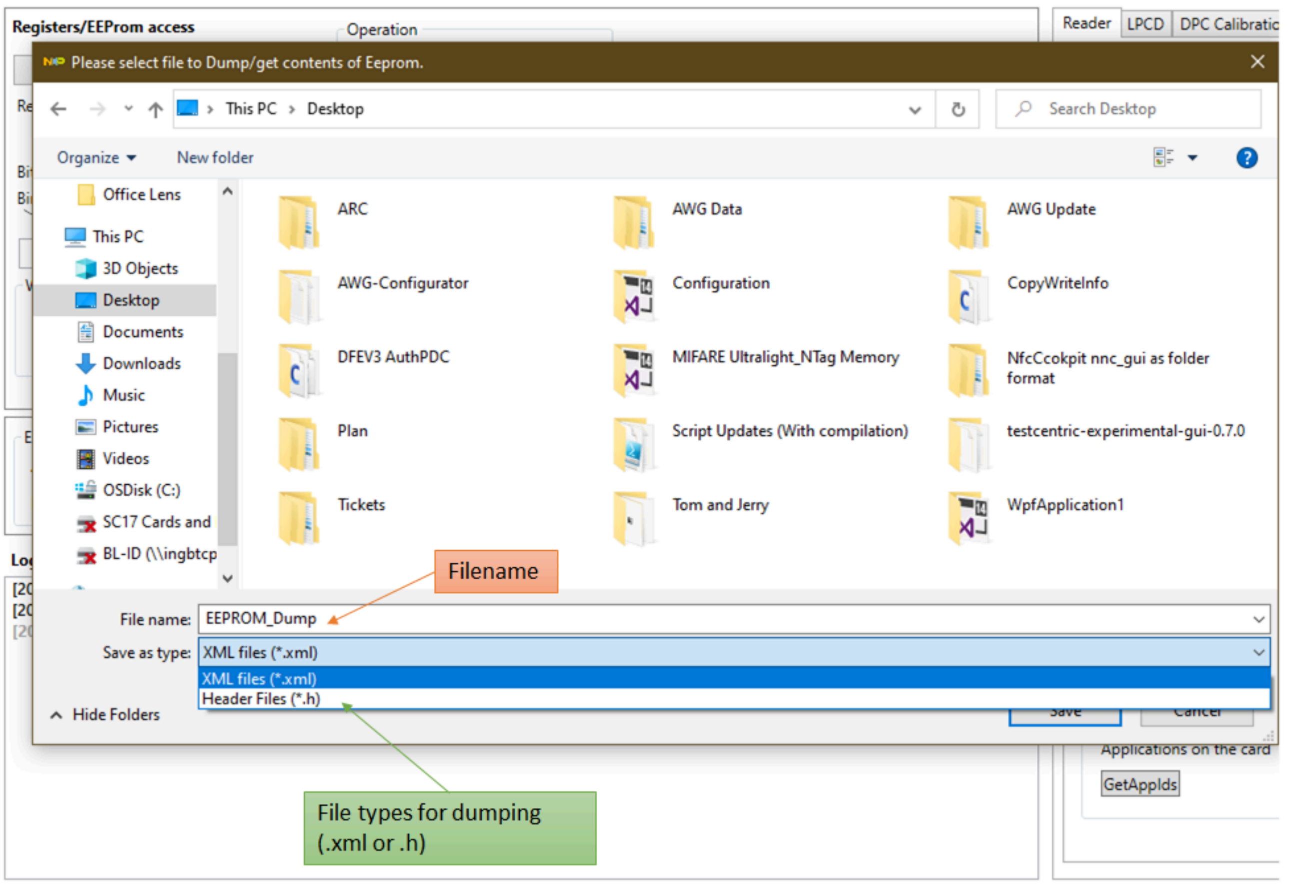



Figure 33. EEPROM Dump File Naming



© 2023 NXP B.V. All rights reserved.

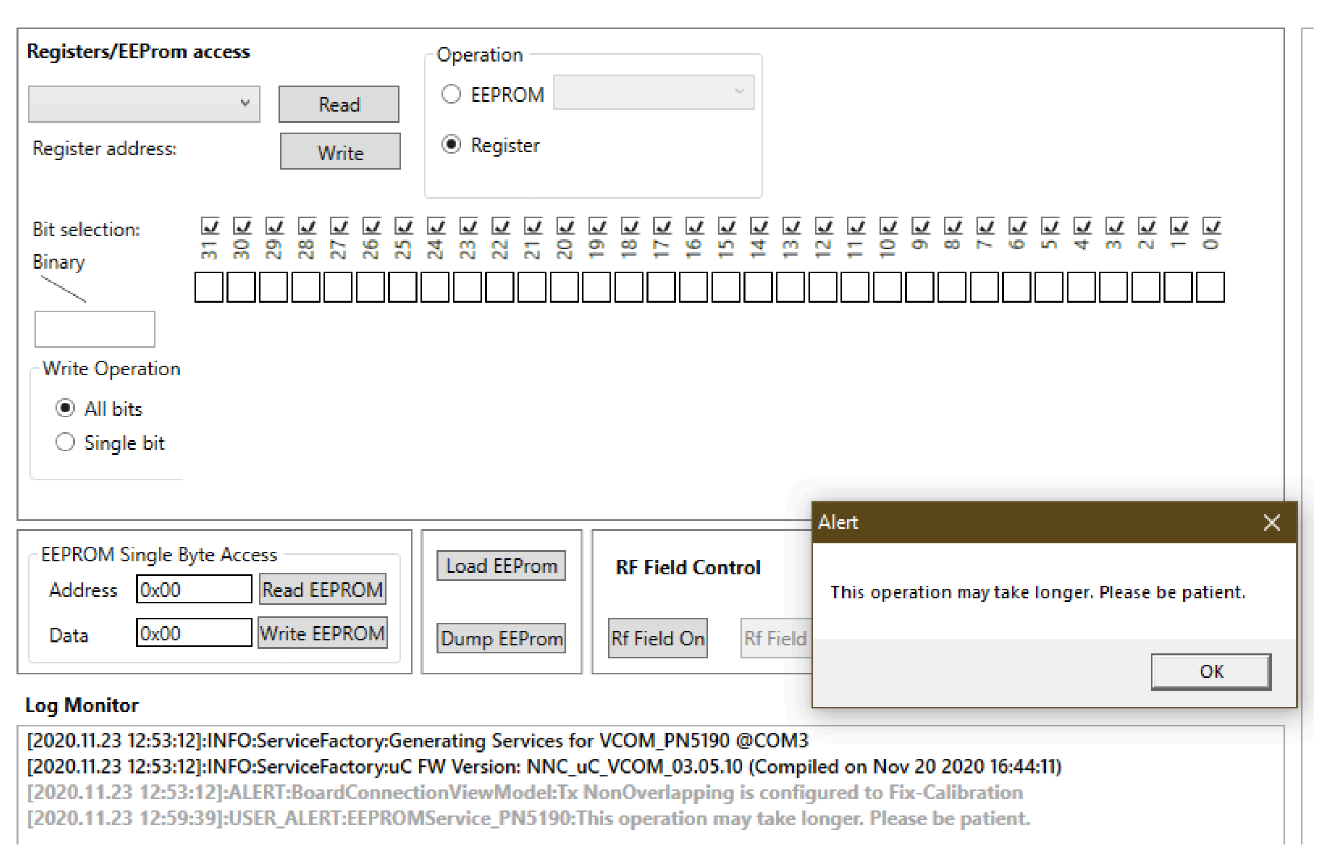



**Figure 34. EEPROM OK Confirmation**



© 2023 NXP B.V. All rights reserved.

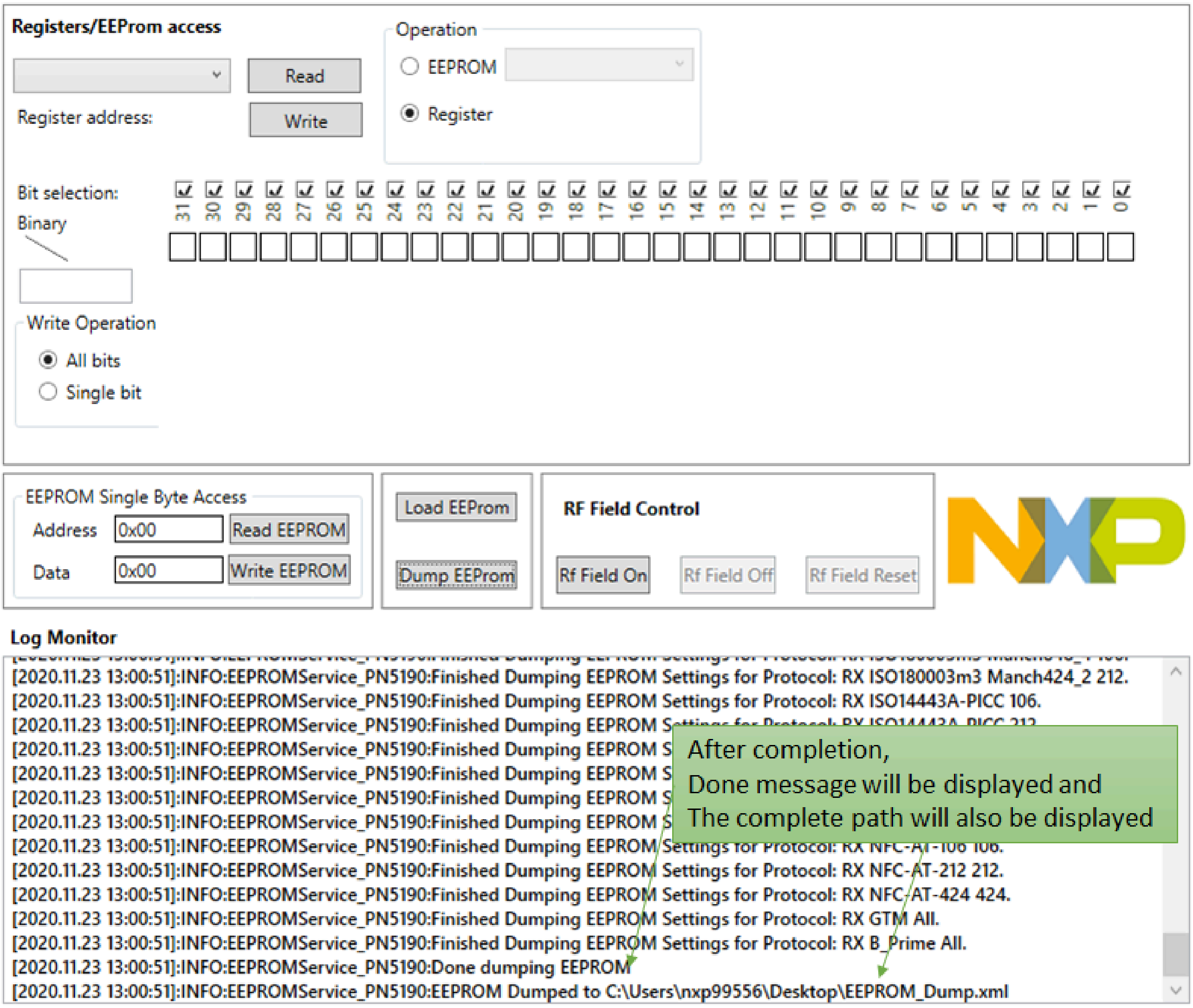



**Figure 35. EEPROM Dump Successful**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

```
▼<E2PROMmapping>
 ▶<Region regionName="SystemConfiguration" regionOffset="0x00" regionLength="0x76" regionAccess="RW" regionType="DATA">...</Region>
 ▶<Region regionName="DPC" regionOffset="0x076" regionLength="0xC1" regionAccess="RW" regionType="DATA">...</Region>
  <Region regionName="ProtocolLookup" regionOffset="0x00" regionLength="0x37" regionAccess="NOT_ACCESSIBLE" regionType="DATA"/>
 ▼<Region regionOffset="0x70" regionName="RegisterValuePair" regionAccess="INDIRECT" regionType="PROTOCOL">
  ▼<Protocol protocolName="TX ISO14443A 106" protocolIndex="0x00" protocolOffset="0x70">
     <Register registerName="CLIF_SS_TX1_CMCFG" registerLogicalAddress="0x3B" registerValue="0x000900FF"/>
     <Register registerName="CLIF_SS_TX2_CMCFG" registerLogicalAddress="0x3C" registerValue="0x000F00FF"/>
     <Register registerName="CLIF_TX_UNDERSHOOT_CONFIG" registerLogicalAddress="0x13" registerValue="0x00000000"/>
     <Register registerName="CLIF_TX_OVERSHOOT_CONFIG" registerLogicalAddress="0x14" registerValue="0x00000000"/>
     <Register registerName="CLIF_TRANSCEIVE_CONTROL" registerLogicalAddress="0x8" registerValue="0x00003D41"/>
     <Register registerName="CLIF_SS_TX_CFG" registerLogicalAddress="0x15" registerValue="0x00002289"/>
   </Protocol>
  ▼<Protocol protocolName="TX ISO14443A 212" protocolIndex="0x01" protocolOffset="0x8E">
     <Register registerName="CLIF_SS_TX1_CMCFG" registerLogicalAddress="0x3B" registerValue="0x000900FF"/>
     <Register registerName="CLIF_SS_TX2_CMCFG" registerLogicalAddress="0x3C" registerValue="0x000900FF"/>
     <Register registerName="CLIF_TX_UNDERSHOOT_CONFIG" registerLogicalAddress="0x13" registerValue="0x00000000"/>
     <Register registerName="CLIF_TX_OVERSHOOT_CONFIG" registerLogicalAddress="0x14" registerValue="0x00000000"/>
     <Register registerName="CLIF_TRANSCEIVE_CONTROL" registerLogicalAddress="0x8" registerValue="0x00000001"/>
     <Register registerName="CLIF_SS_TX_CFG" registerLogicalAddress="0x15" registerValue="0x00002289"/>
   </Protocol>
  ▼<Protocol protocolName="TX ISO14443A 424" protocolIndex="0x02" protocolOffset="0xAC">
     <Register registerName="CLIF_SS_TX1_CMCFG" registerLogicalAddress="0x3B" registerValue="0x000900FF"/>
     <Register registerName="CLIF_SS_TX2_CMCFG" registerLogicalAddress="0x3C" registerValue="0x000900FF"/>
     <Register registerName="CLIF_TX_UNDERSHOOT_CONFIG" registerLogicalAddress="0x13" registerValue="0x00000000"/>
     <Register registerName="CLIF_TX_OVERSHOOT_CONFIG" registerLogicalAddress="0x14" registerValue="0x00000000"/>
     <Register registerName="CLIF_TRANSCEIVE_CONTROL" registerLogicalAddress="0x8" registerValue="0x00000001"/>
     <Register registerName="CLIF_SS_TX_CFG" registerLogicalAddress="0x15" registerValue="0x00002289"/>
   </Protocol>
  ▼<Protocol protocolName="TX ISO14443A 848" protocolIndex="0x03" protocolOffset="0xCA">
     <Register registerName="CLIF_SS_TX1_CMCFG" registerLogicalAddress="0x3B" registerValue="0x000900FF"/>
     <Register registerName="CLIF_SS_TX2_CMCFG" registerLogicalAddress="0x3C" registerValue="0x000900FF"/>
     <Register registerName="CLIF_TX_UNDERSHOOT_CONFIG" registerLogicalAddress="0x13" registerValue="0x00000000"/>
     <Register registerName="CLIF_TX_OVERSHOOT_CONFIG" registerLogicalAddress="0x14" registerValue="0x00000000"/>
     <Register registerName="CLIF_TRANSCEIVE_CONTROL" registerLogicalAddress="0x8" registerValue="0x00000001"/>
     <Register registerName="CLIF_SS_TX_CFG" registerLogicalAddress="0x15" registerValue="0x00002289"/>
   </Protocol>
  ▼<Protocol protocolName="TX ISO14443B 106" protocolIndex="0x04" protocolOffset="0xE8">
     <Register registerName="CLIF_SS_TX1_CMCFG" registerLogicalAddress="0x3B" registerValue="0x000900FF"/>
     <Register registerName="CLIF_SS_TX2_CMCFG" registerLogicalAddress="0x3C" registerValue="0x000900FF"/>
     <Register registerName="CLIF_TX_UNDERSHOOT_CONFIG" registerLogicalAddress="0x13" registerValue="0x00000000"/>
     <Register registerName="CLIF_TX_OVERSHOOT_CONFIG" registerLogicalAddress="0x14" registerValue="0x00000000"/>
     <Register registerName="CLIF_TRANSCEIVE_CONTROL" registerLogicalAddress="0x8" registerValue="0x00000001"/>
     <Register registerName="CLIF_SS_TX_CFG" registerLogicalAddress="0x15" registerValue="0x00002289"/>
   </Protocol>
```


**Figure 36.  EEPROM saved to XML**



© 2023 NXP B.V. All rights reserved.

### 4.2.4 Load

Loads a previous saved file and stores it into the user area of EEPROM.

***Note:*** *Loading of EEPROM from .xml file is only supported. To know the .xml format perform Dump EEPROM once.*

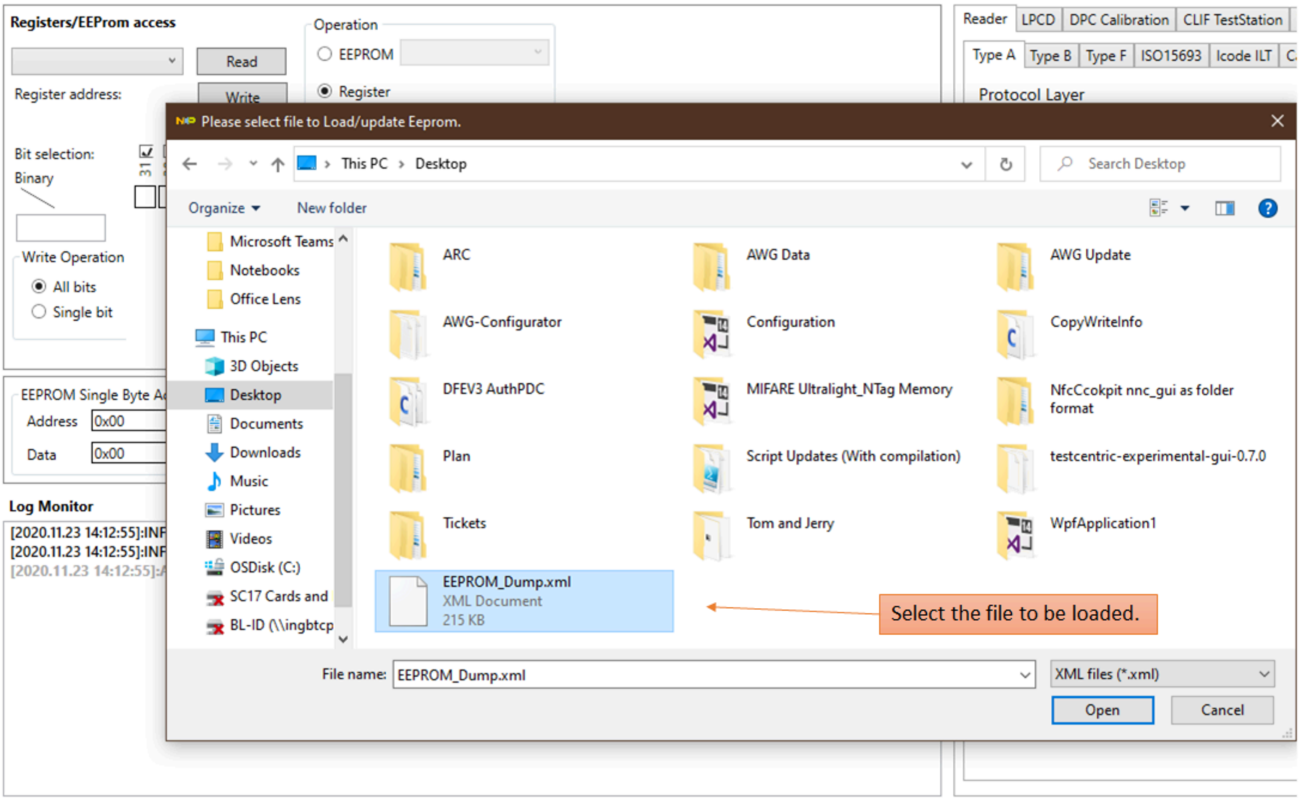



**Figure 37. EEPROM Load File Selection**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

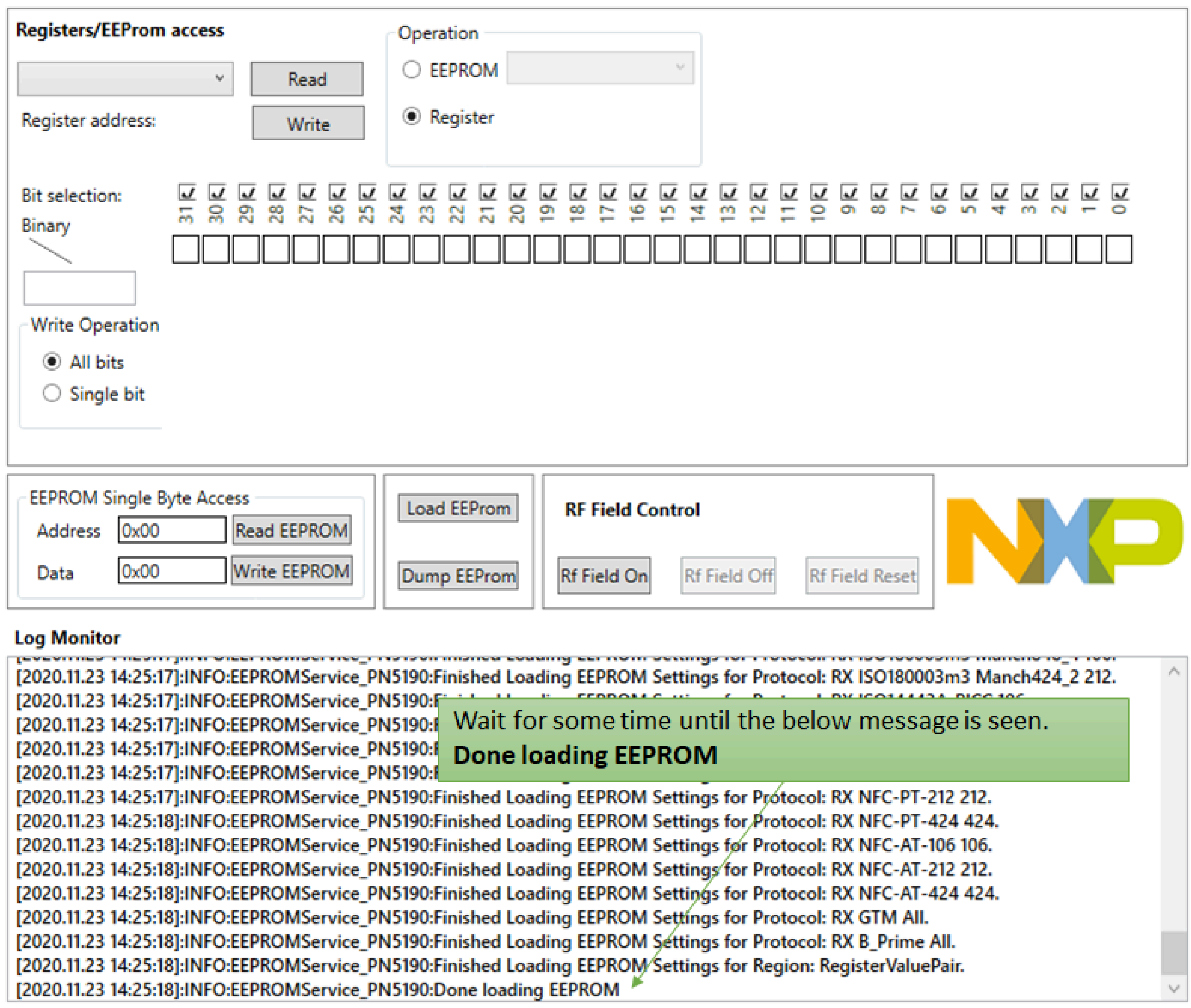



**Figure 38. EEPROM Load Successful**

## 4.3 Reader Mode (PCD)

**ReaderMode** offers all protocol operations. Below mentioned are the supported protocol list:

- ISO14443 - A
- ISO14443 - B
- Type F
- ISO15693
- ISO18000P3M3

### 4.3.1 ISO14443 - A

**Type A** feature allows the user to do operations such as

| | |
|---|---|
| **Layer 14443-3a commands** | Layer3 commands which respond with ATQA until SAK. |
| **Layer 14443-4a commands** | Layer4 commands which respond with ATS. |



© 2023 NXP B.V. All rights reserved.

| | |
|---|---|
| **Protocol Tuning** | Protocol Tuning with Single/Endless REQA. |
| **Data Exchange** | Layer4 data exchange with/without CRC. |
| **Get AppIds** | Read AppIds for MIFARE DESFire after Layer4 exchange. |

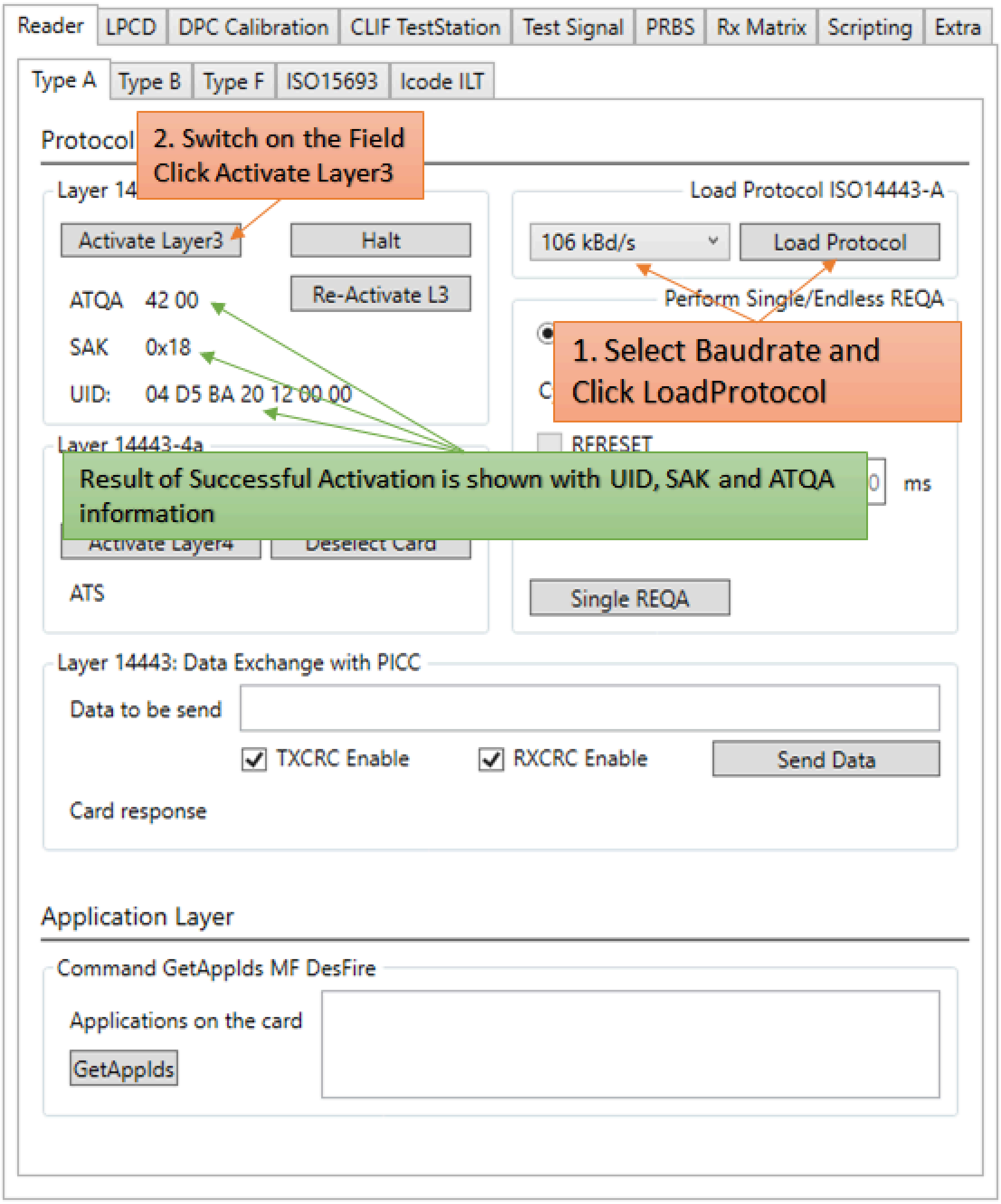



**Figure 39. Level 3 Activation of MIFARE 1K**



© 2023 NXP B.V. All rights reserved.

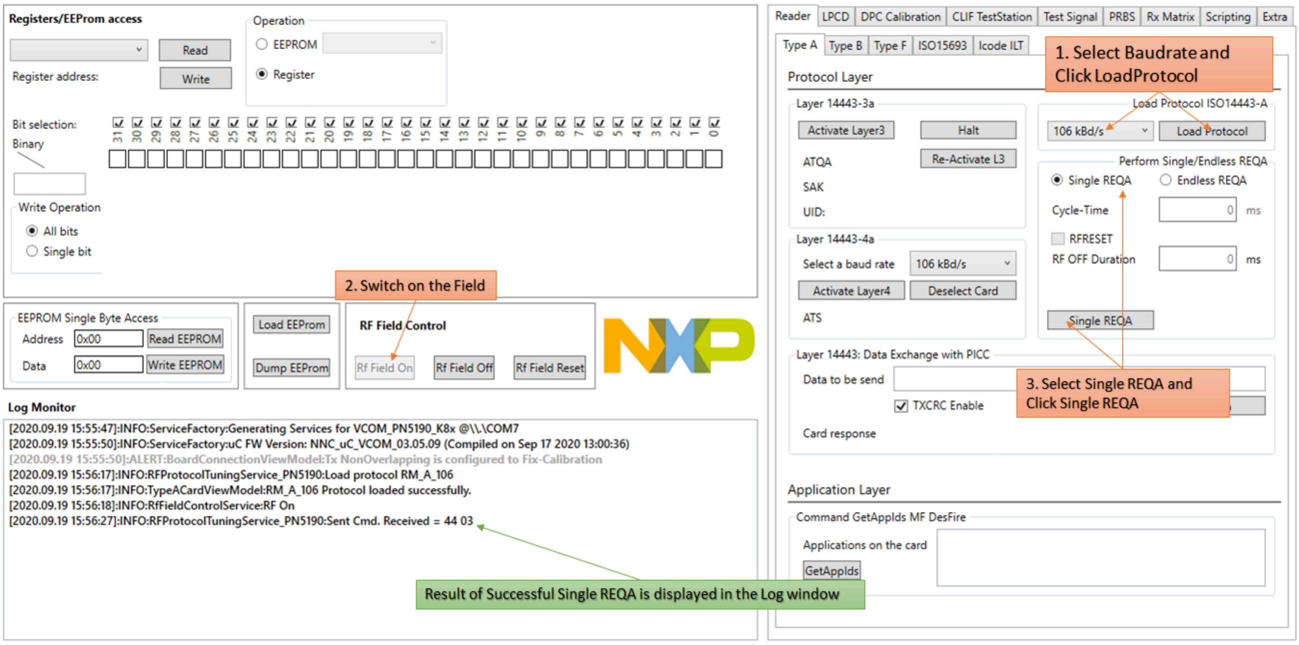



**Figure 40. Single REQUEST A**

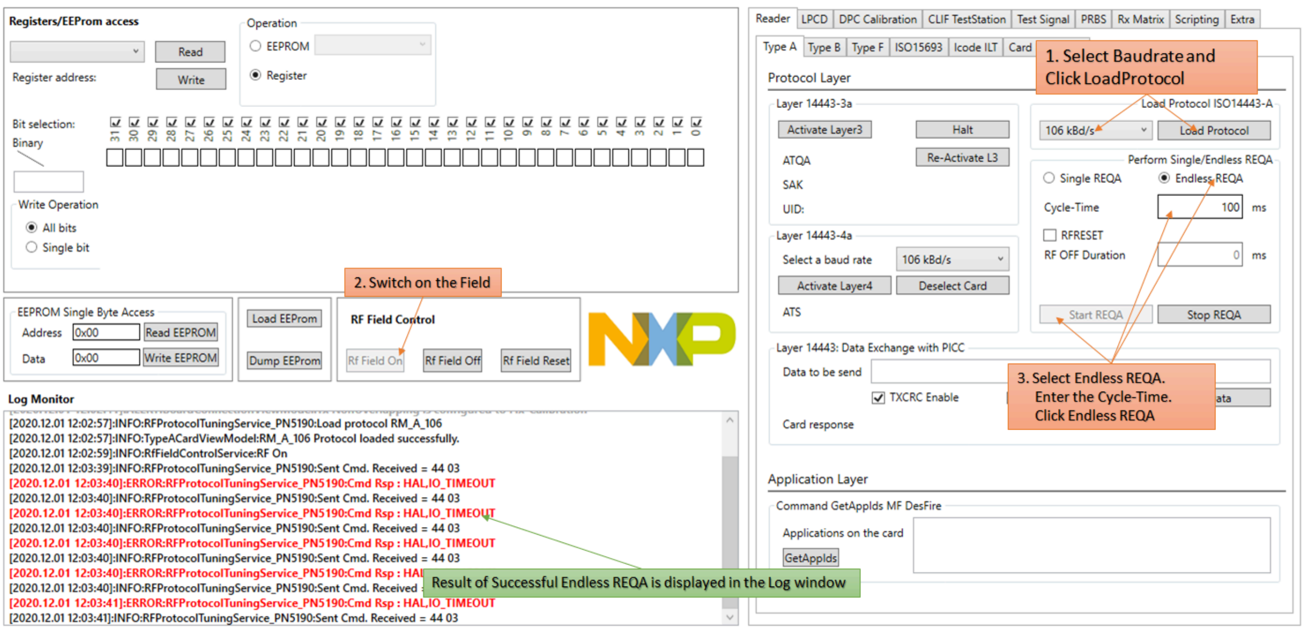



**Figure 41. Endless REQA without RF Reset**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

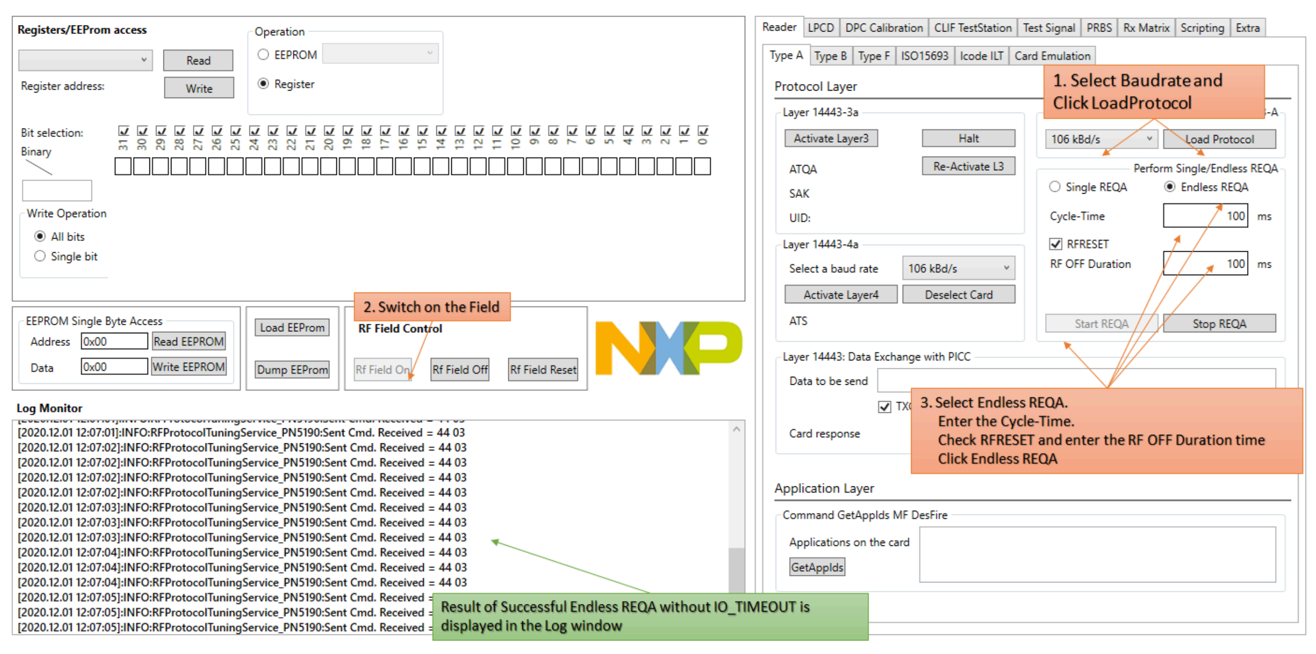



**Figure 42. Endless REQA with RF Reset**

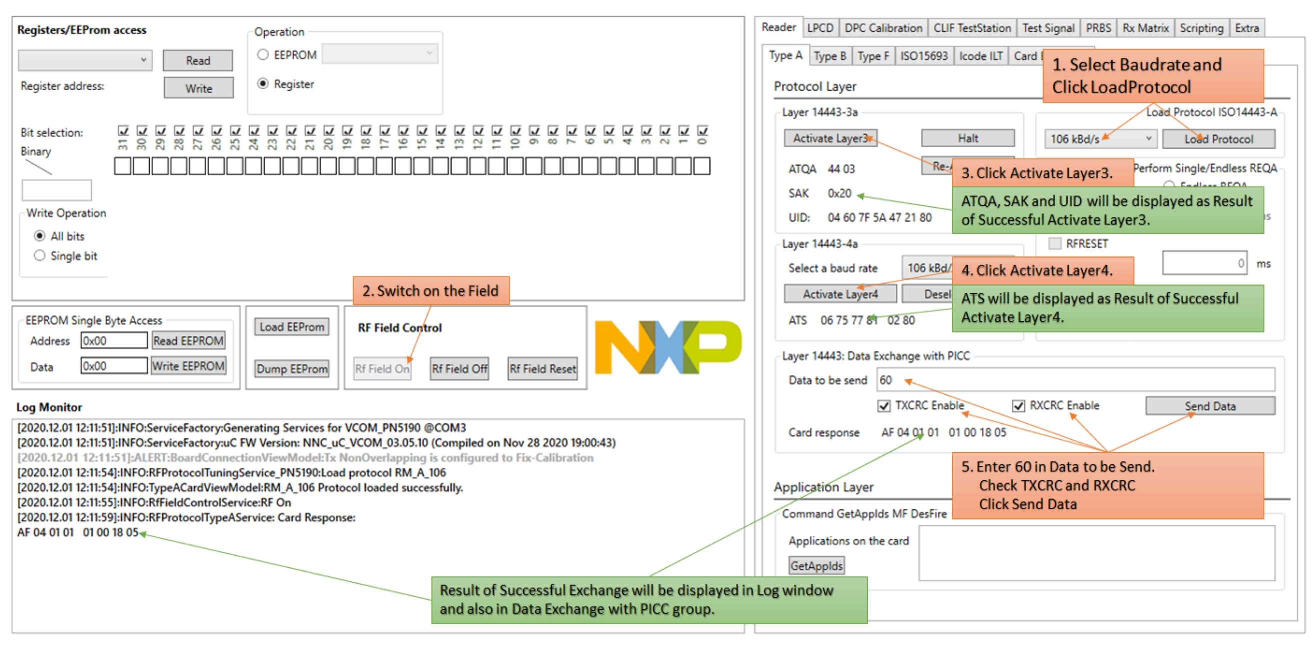



**Figure 43. MIFARE DESFire L4 Exchange**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

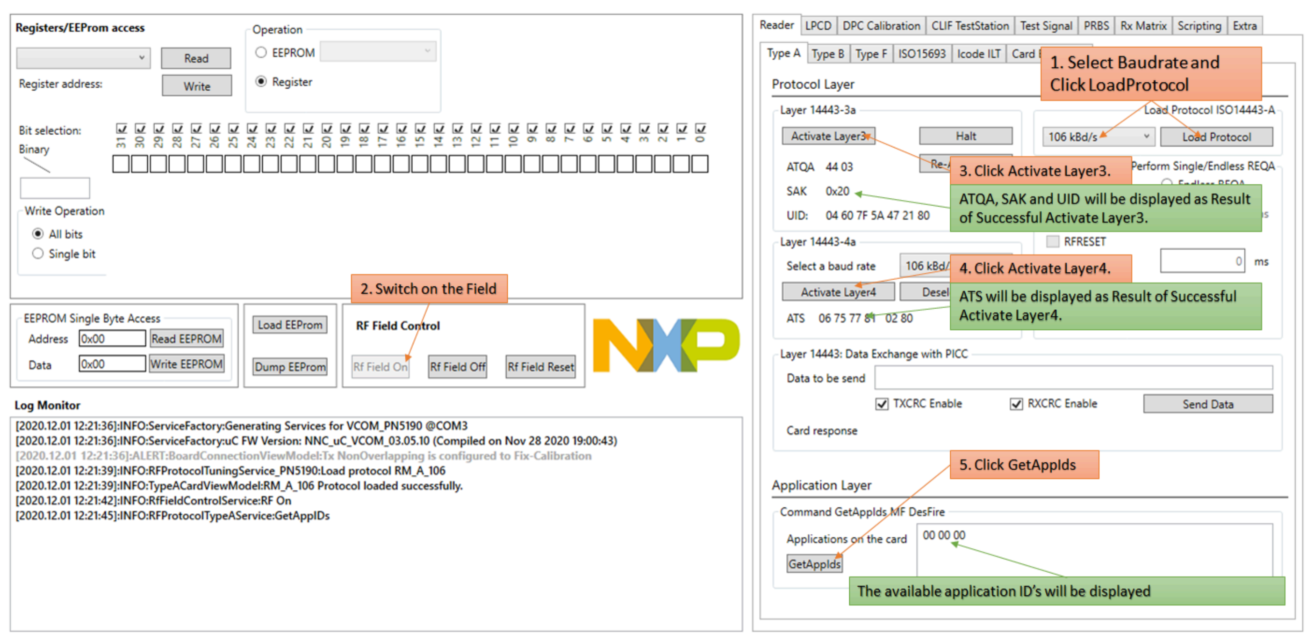



**Figure 44. MIFARE DESFire GetAppID's**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.3.2 ISO14443 - B

**Type B** feature allows the user to do operations such as

| | |
|---|---|
| **Activation commands** | Activate commands which respond with PUPI |
| **Protocol Tuning** | Protocol Tuning with Single/Endless REQB |
| **Data Exchange** | Layer4 data exchange with/without CRC |

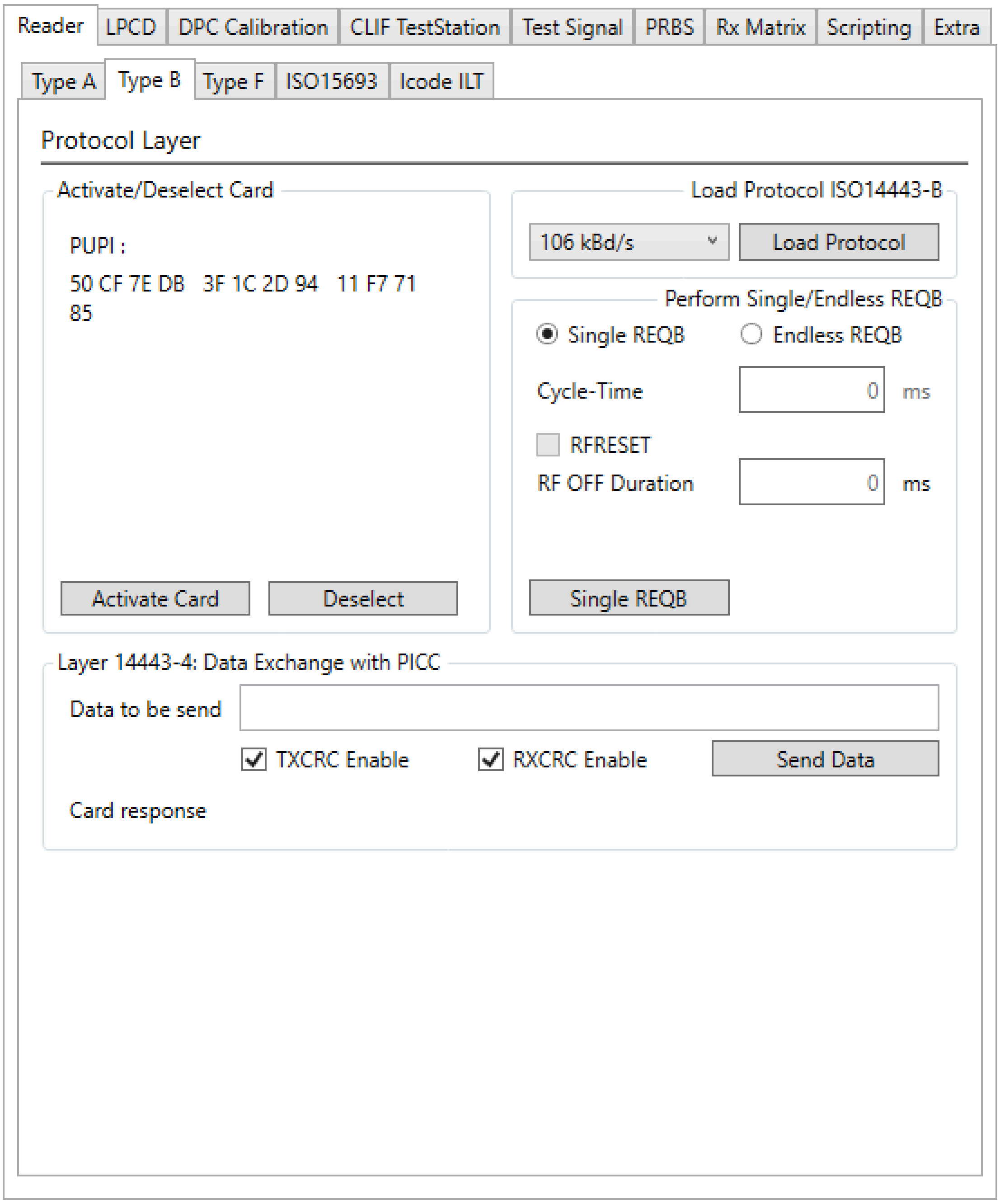



**Figure 45. Activation**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

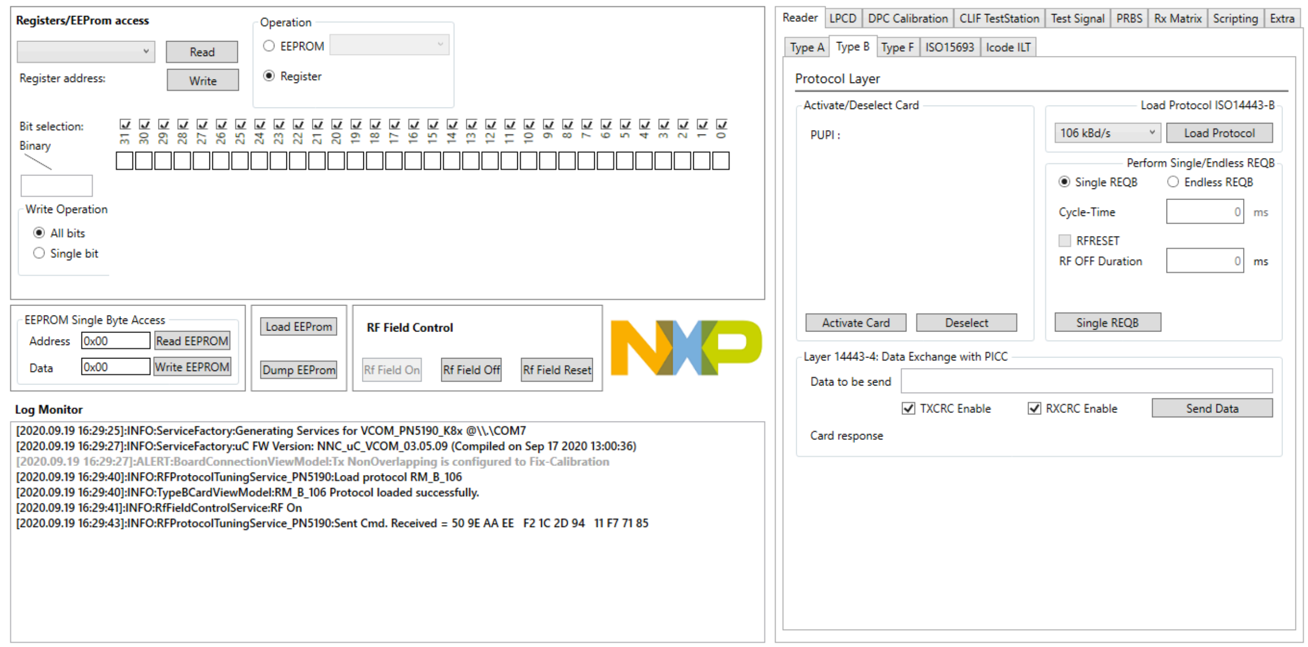


**Figure 46. Single REQUEST B**

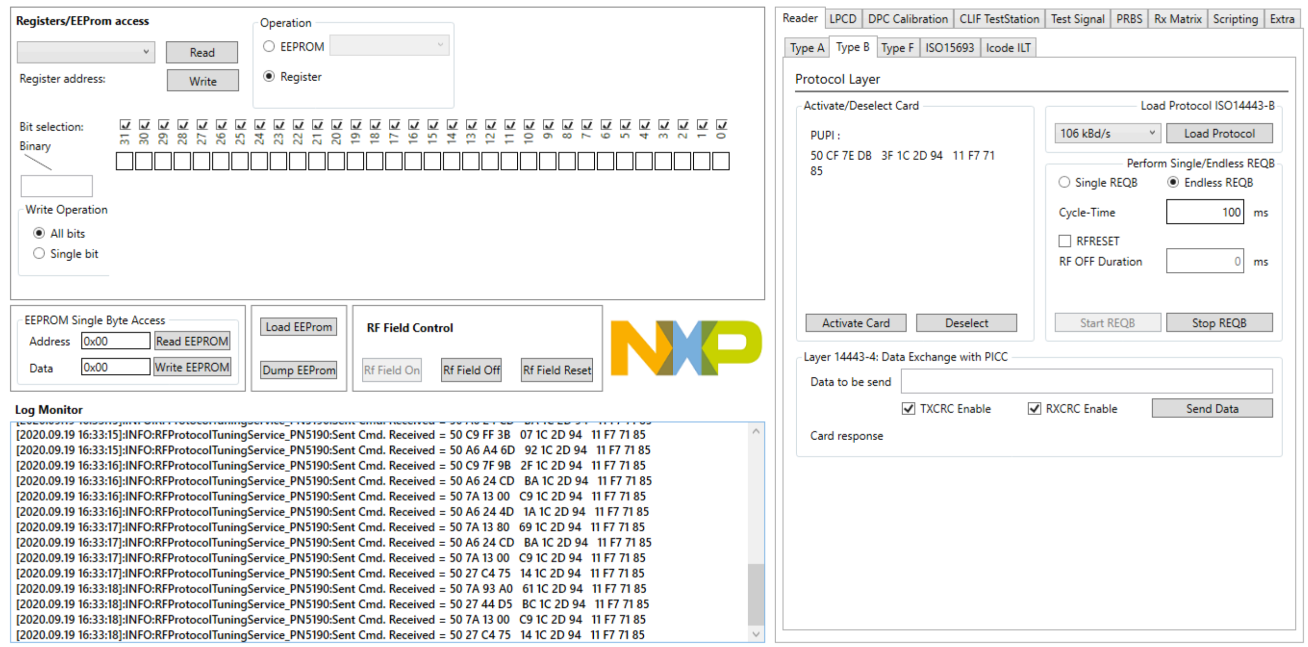


**Figure 47. Endless REQB without RF Reset**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

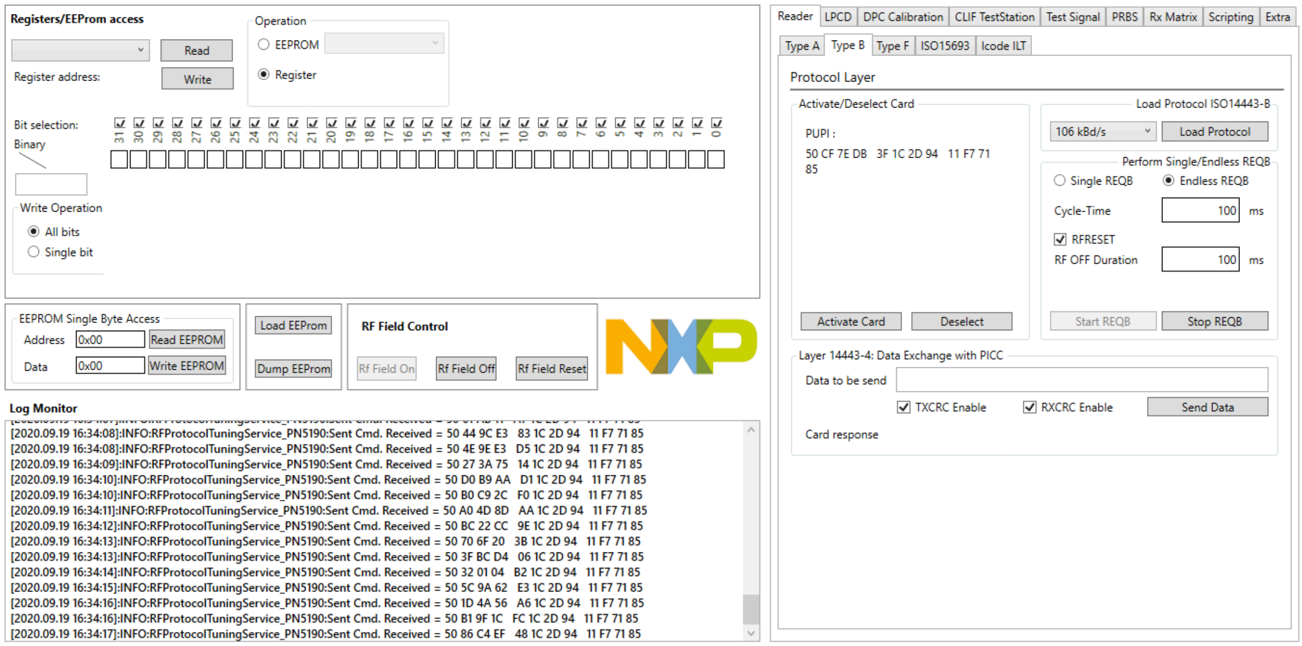


Figure 48. Endless REQB with RF Reset

## 4.3.3 Type F

**Type F** feature allows the user to do operations such as

| | |
|---|---|
| **REQC commands** | REQC commands which respond with IDmPMm |
| **Protocol Tuning** | Protocol Tuning with Single/Endless REQC |
| | Protocol Tuning with Single/Endless CardDetect |

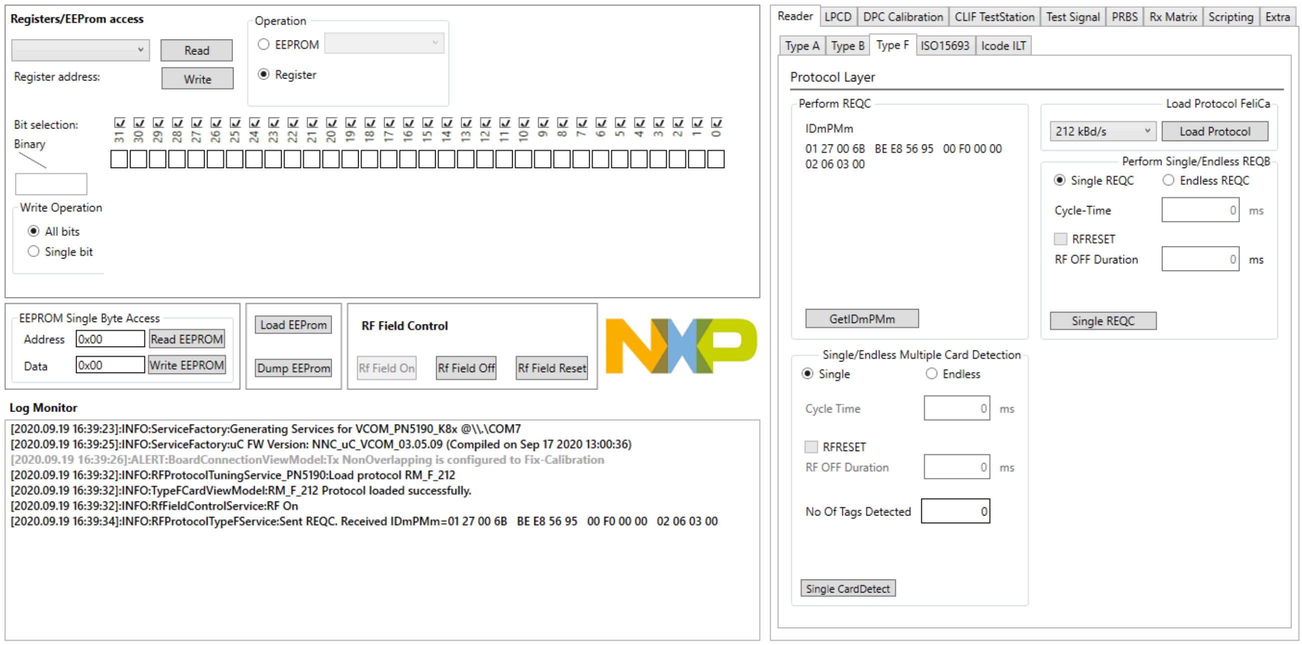


Figure 49. GetIDM/PDM



© 2023 NXP B.V. All rights reserved.

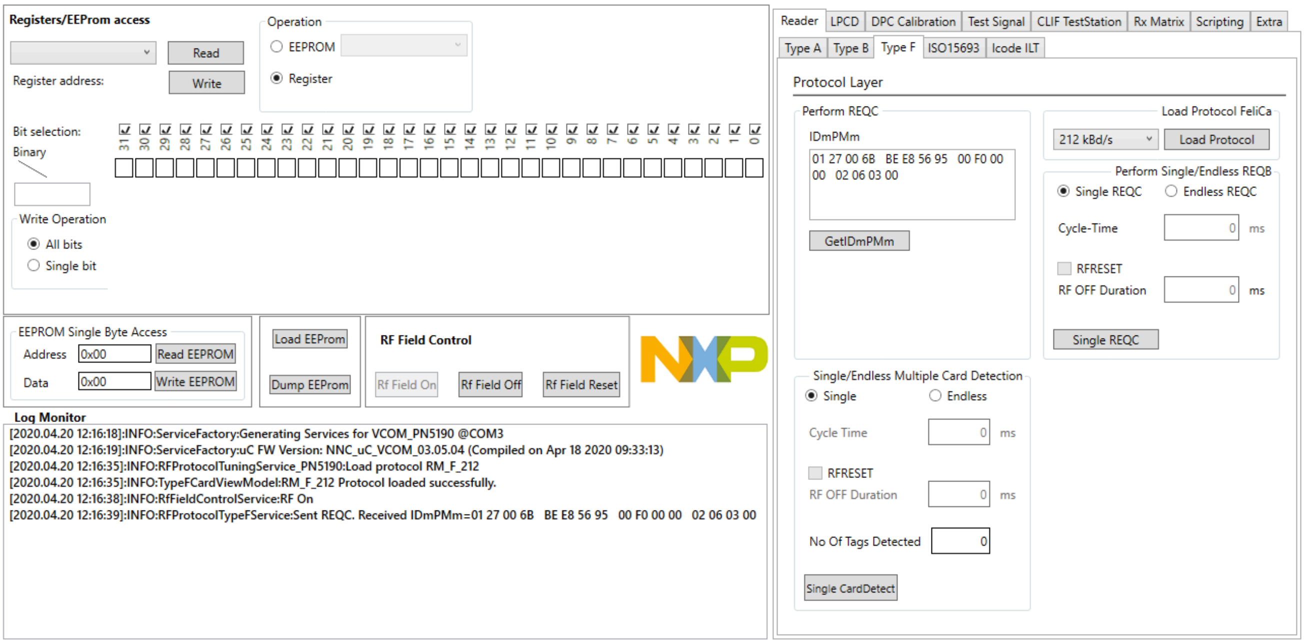



**Figure 50. Single REQUEST C**

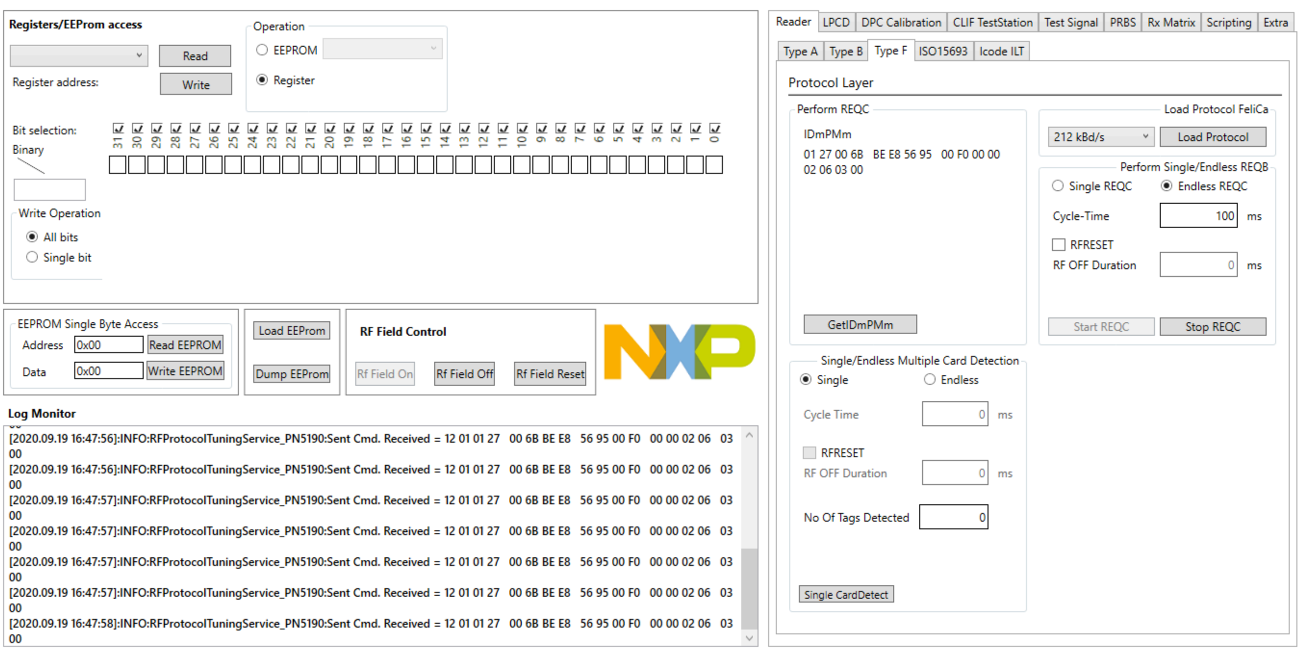



**Figure 51. Endless REQC without RF Reset**



All information provided in this document is subject to legal disclaimers.

Rev. 1.0 — 10 May 2023

© 2023 NXP B.V. All rights reserved.

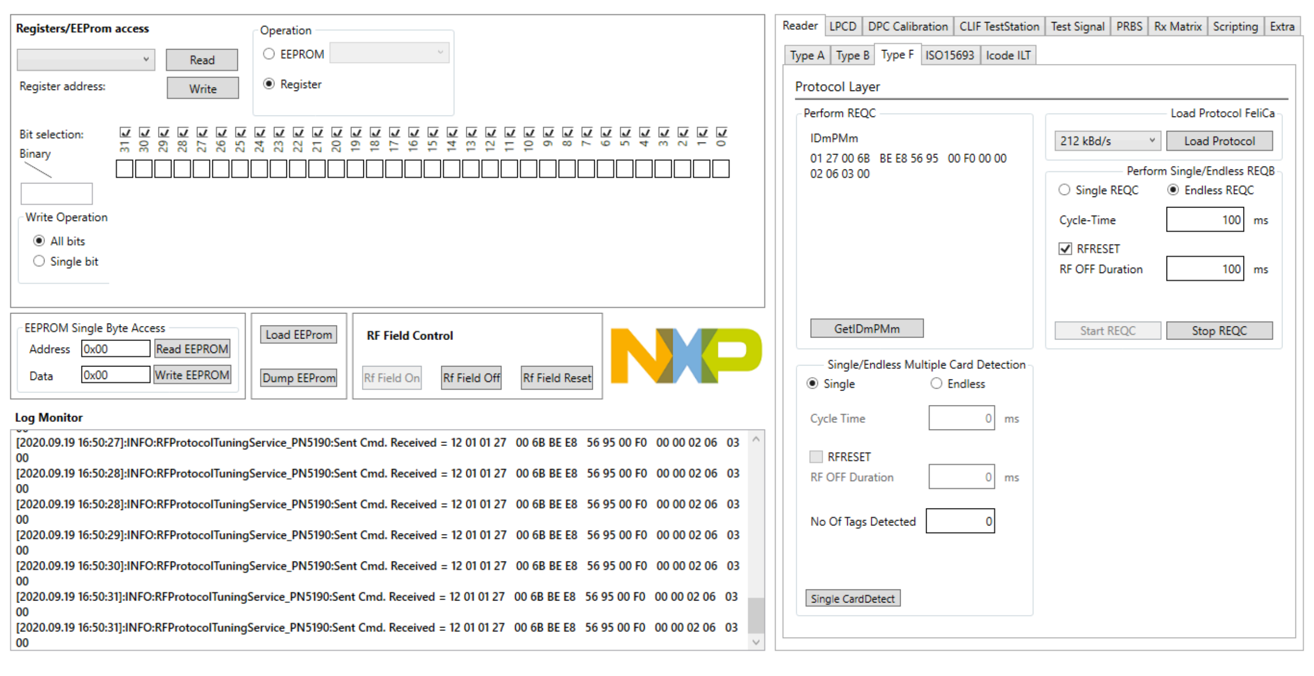


Figure 52.  Endless REQC with RF Reset

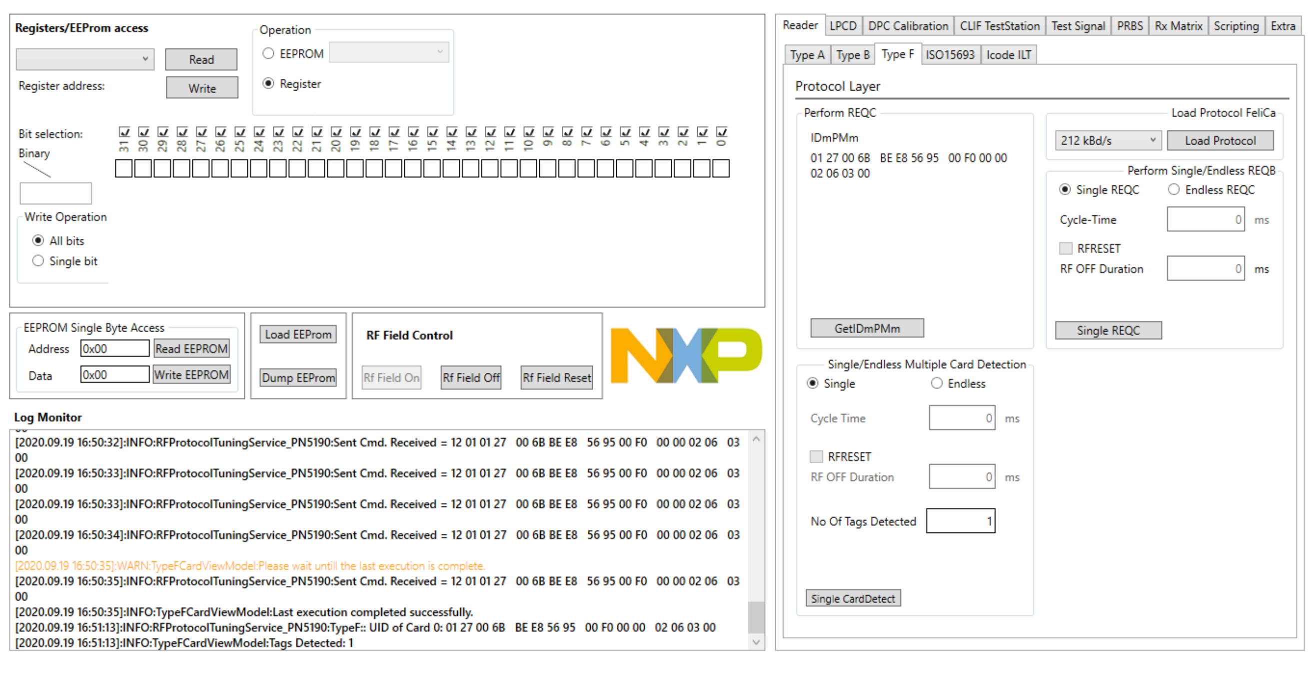


Figure 53.  Single CardDetect



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

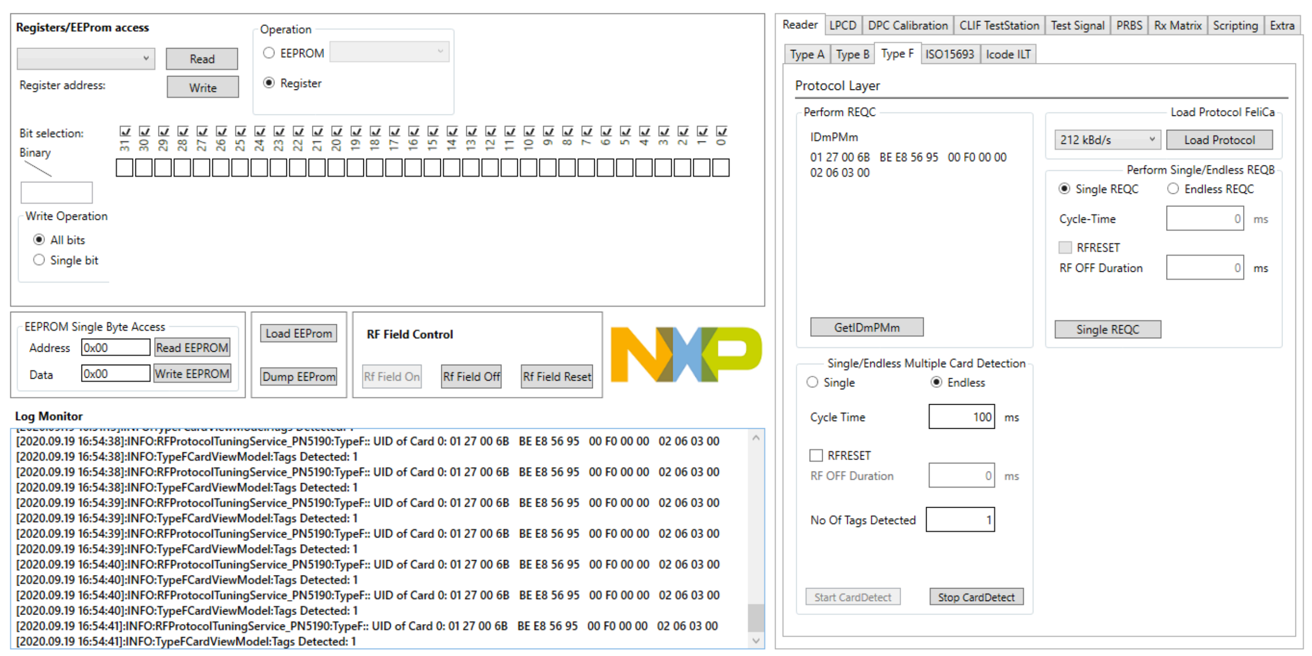


Figure 54.  Endless CardDetect without RF Reset

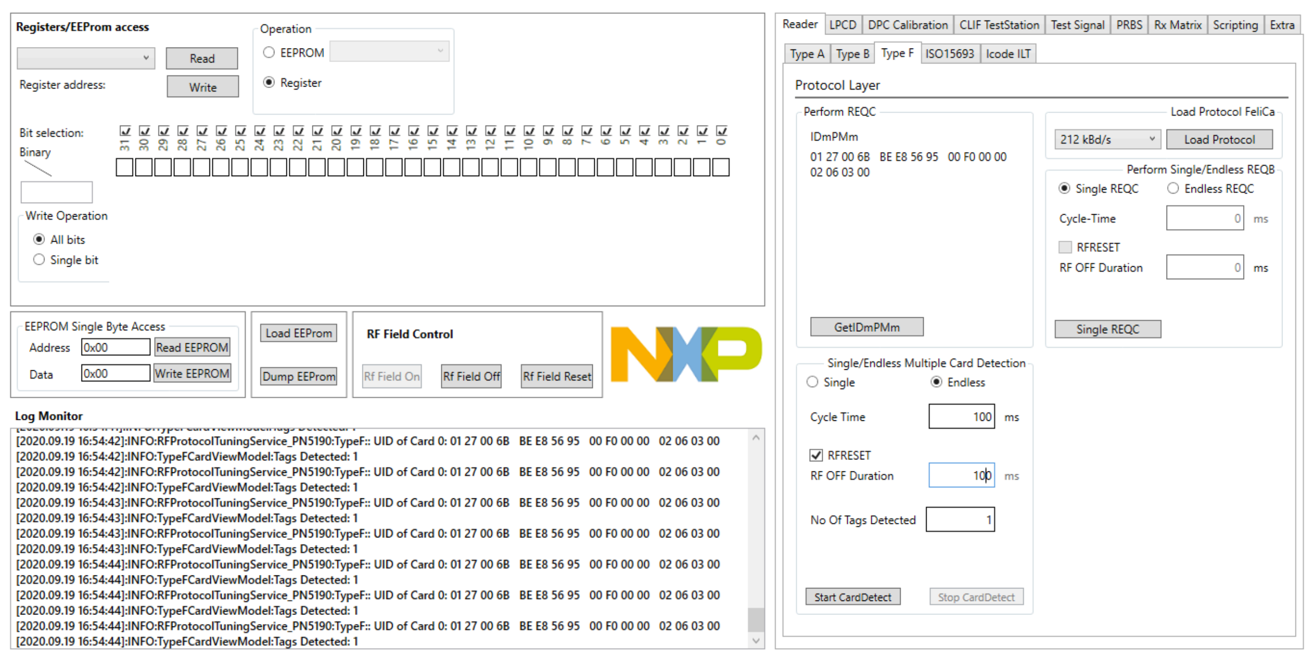


Figure 55.  Endless CardDetect with RF Reset



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.3.4 ISO15693

**Type Sli15693** feature allows the user to do operations such as

| | |
|---|---|
| **Inventory commands** | Responds depending on the number of slots selected |
| **Protocol Tuning** | Protocol Tuning with Single/Endless Inventory and Single/Endless Fast Inventory |
| **Data Exchange** | High-level data exchange with/without CRC |
| **Read Block** | Read a specific block number from memory after inventory |
| **High baud rate** | High baud rate and timing selection. |

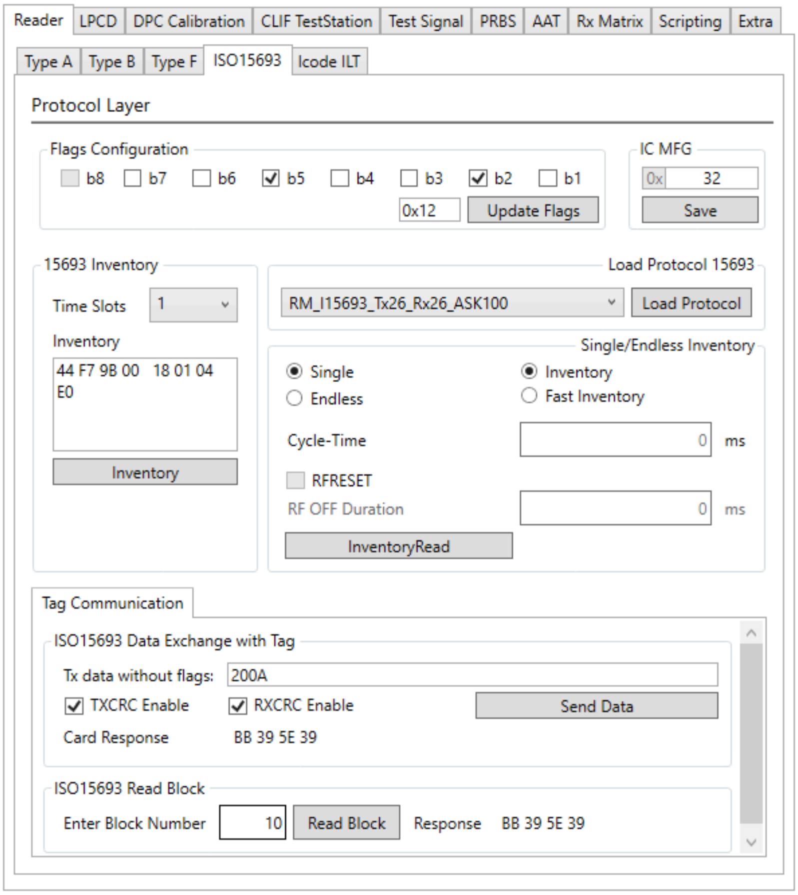



**Figure 56. Inventory**



© 2023 NXP B.V. All rights reserved.

### 4.3.5 ISO18000P3M3

*NxpNfcCockpit* provides support for ISO18000 Part 3 Mode 3 protocol wherein it features operations such as:

| | |
|---|---|
| **Inventory Commands** | The tag replies based on the number of slots selected to the inventory commands |
| **Protocol Tuning** | Protocol tuning is performed with operations like Single/Endless Card Detect. |
| **Begin Round** | Begin round instructs the tag to load new Q values to the slot counter. |

#### 4.3.5.1 Inventory with One Slot

Inventory is the tag identification operation. Inventory operation can be performed through NXP NFC Cockpit. The reader detects the tag and requests for a reply from the tag. Cockpit displays this reply or UI of the tag when it is detected by the reader during Inventory operation.

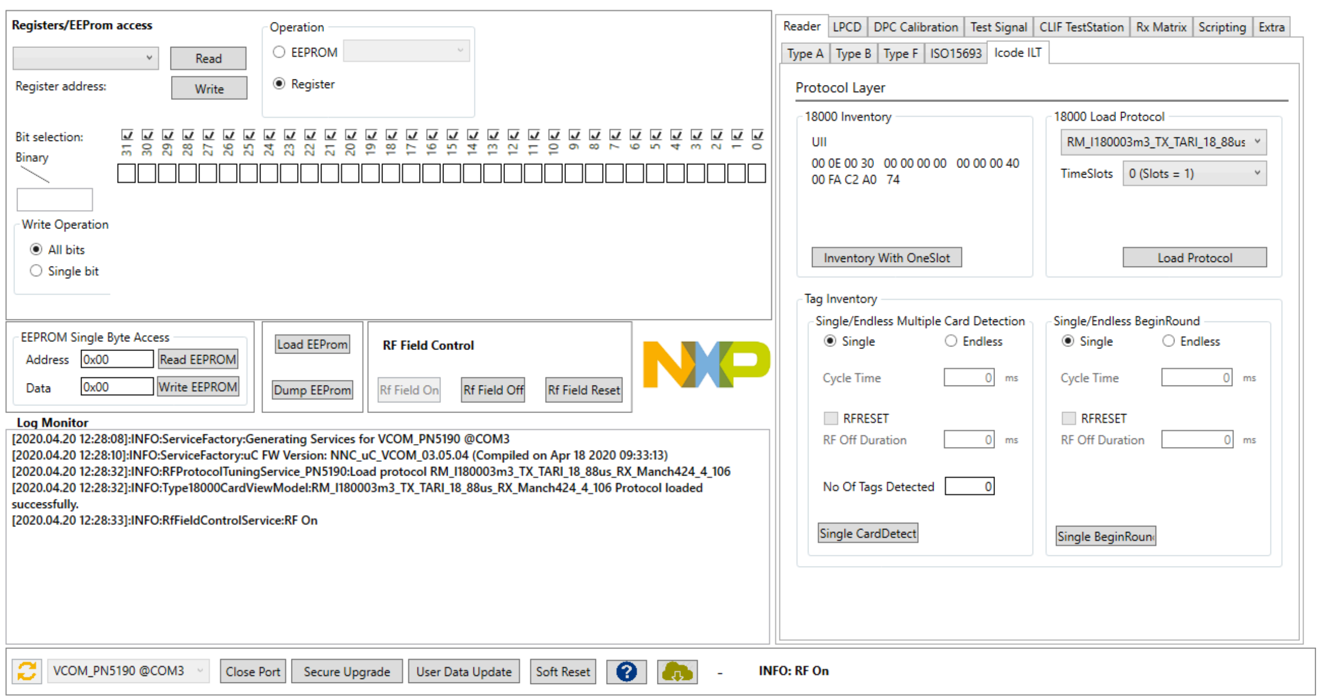


**Figure 57. Inventory With One Slot**



© 2023 NXP B.V. All rights reserved.

### 4.3.5.2 Single Card Detect

Single Card detect operation detects any number of cards present on the reader antenna at once. The User can select the number of slots by selecting a Q value from 0 to 7. The number of slots depend on the Q values (Number of Slots = 2^Q). For instance, if we select random number Q as 4 then number of slots is 16 so the the card has the choice to reply in any of the 16 slots. If the card replies in a particular slot, there is a 17 byte response else 3 byte response containing 02 00 00.

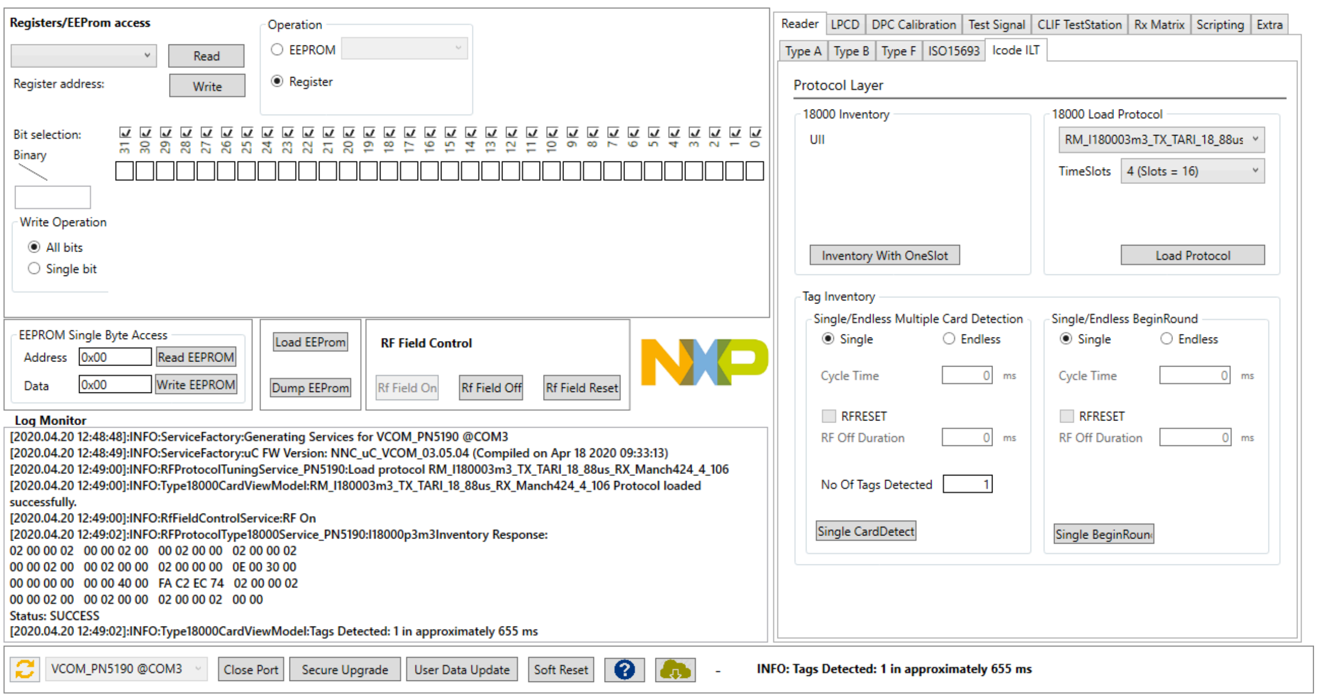



**Figure 58. Single Card Detect**

### 4.3.5.3 Endless Card Detect

Endless Card detect operation detects any number of cards present on the reader antenna in a continuous loop unless the user clicks. The user can select the number of slots by selecting a Q value from 0 to 7. If the number of slots selected is one, then the card responds in that slot itself else if number of slots is more then card can choose to reply in any of the slots. The response is continuously displayed on the cockpit log monitor until user clicks "Stop Card Detect" button.

Endless card detect can be performed in 2 ways:

**Endless Card Detect With RF Reset**

In this mode, we can perform field reset at regular intervals during Endless Card Detect and can specify the RF Off duration



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

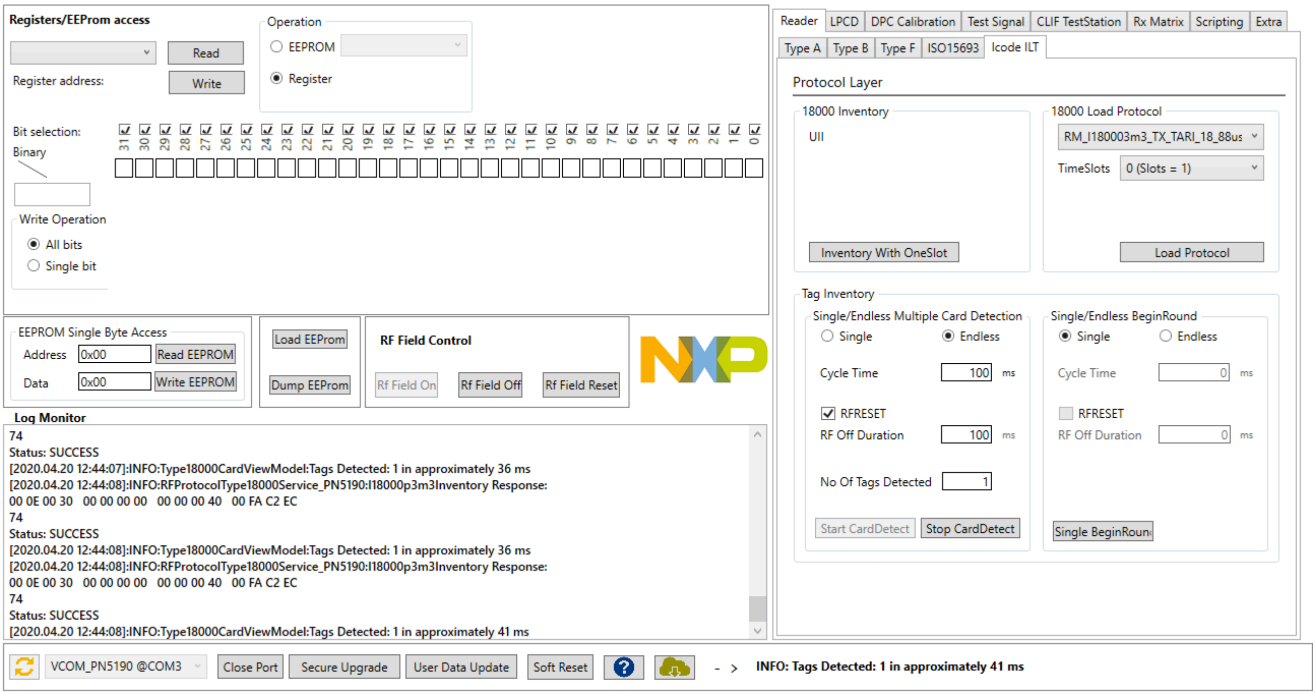



**Figure 59. Endless Single Card Detect With Reset**

### pEndless Card Detect Without RF Reset

In this mode, Endless Card Detect is performed without any resetting the RF field.

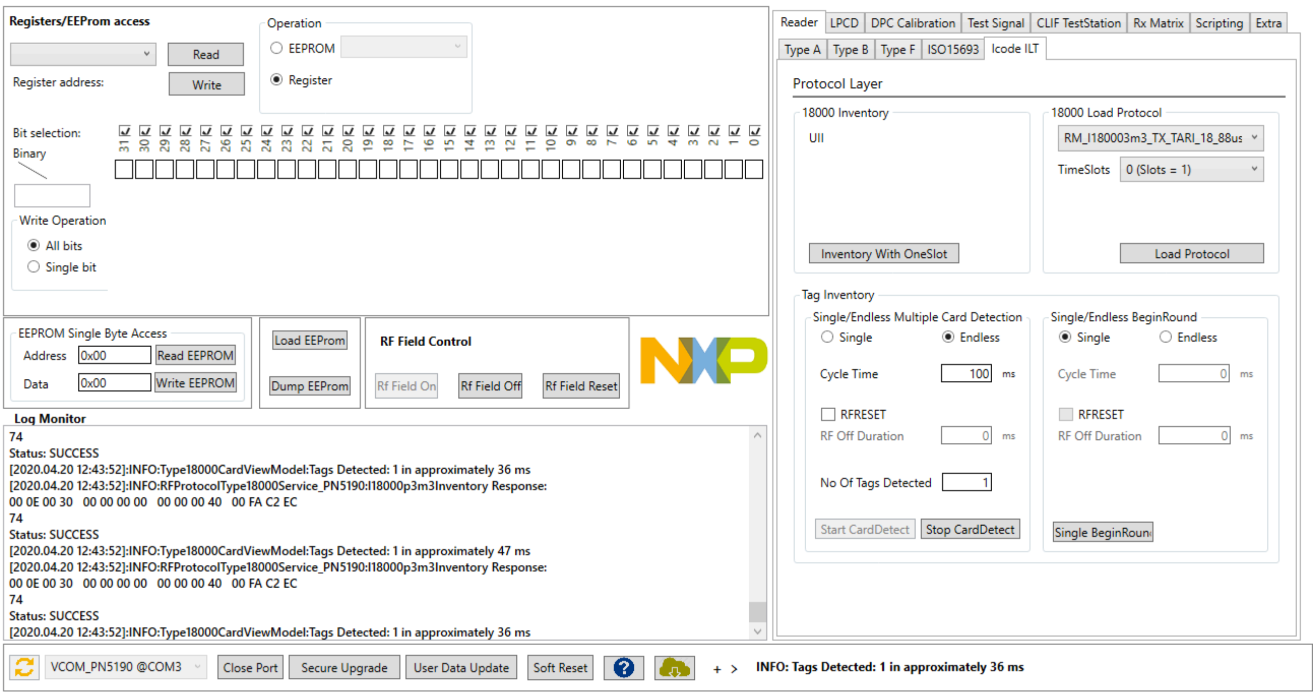



**Figure 60. Endless Single Card Without Reset**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

### 4.3.5.4 Begin Round

When Begin round command is sent to the card, it instructs the tag to load slot counter with new Q values. The card can choose the slot to reply and typically replies with 2 byte response. Unlike single card detect Begin Round does not exhibit anti-collision mechanism and therefore cannot deal with multiple cards.

Begin round also has single and endless modes.

**Single Begin Round**

While single mode is selected, Begin round command is sent to the card and card will reply back a 2 byte response.

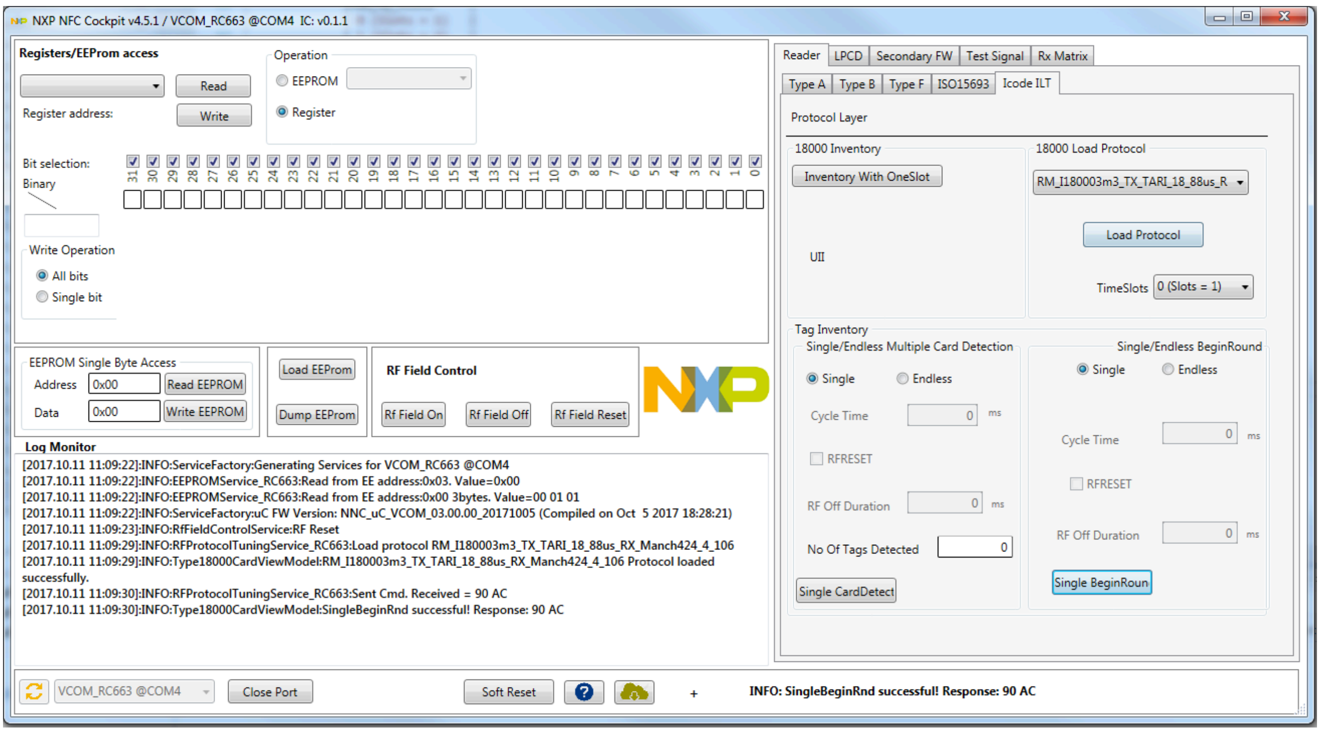



**Figure 61. Single Begin Round**

**Endless Begin Round**

In endless mode, Begin round command is sent to the card in a continuous loop. We can specify a cycle time in milliseconds for Begin round to be performed. Endless begin round can be performed with RF reset by specifying the RF off duration and without RF reset as well.



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

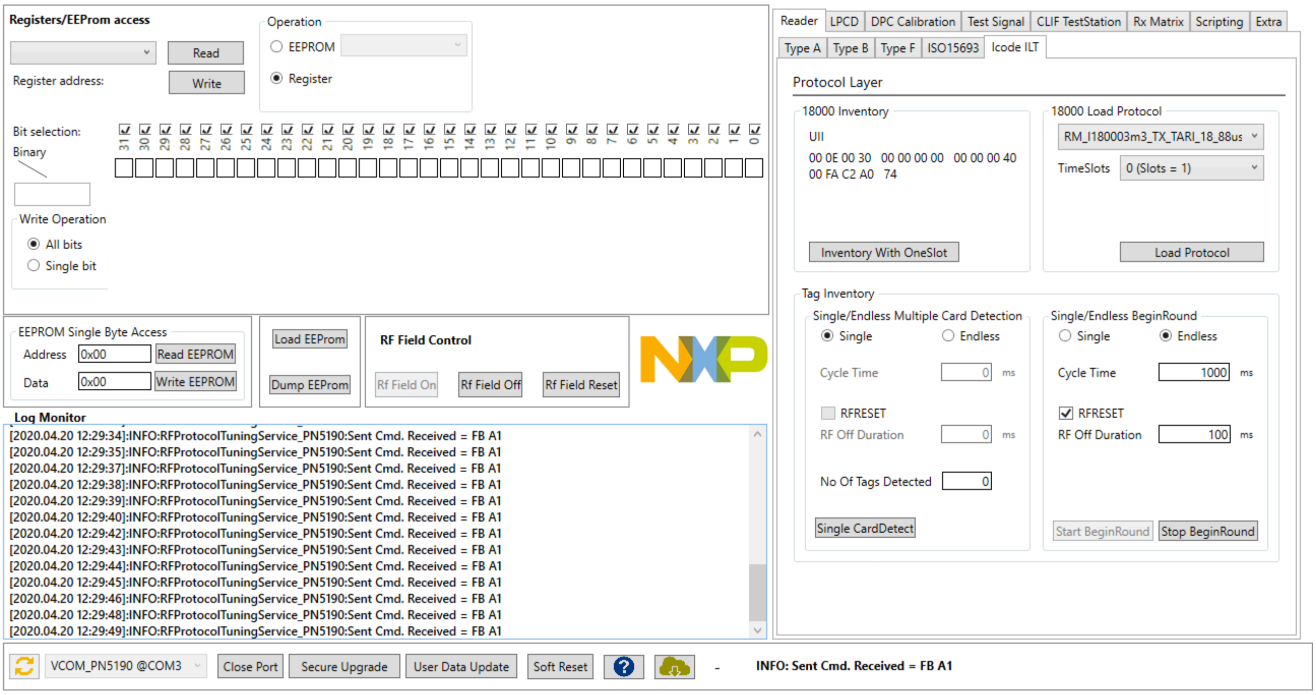



**Figure 62. Endless Begin Round With RF Reset**

## 4.4 Protocol tuning

**Protocol Tuning** helps in performing a continuous ping operation where the user can modify / alter other components and check the effect of that alteration on the transmission and reception.

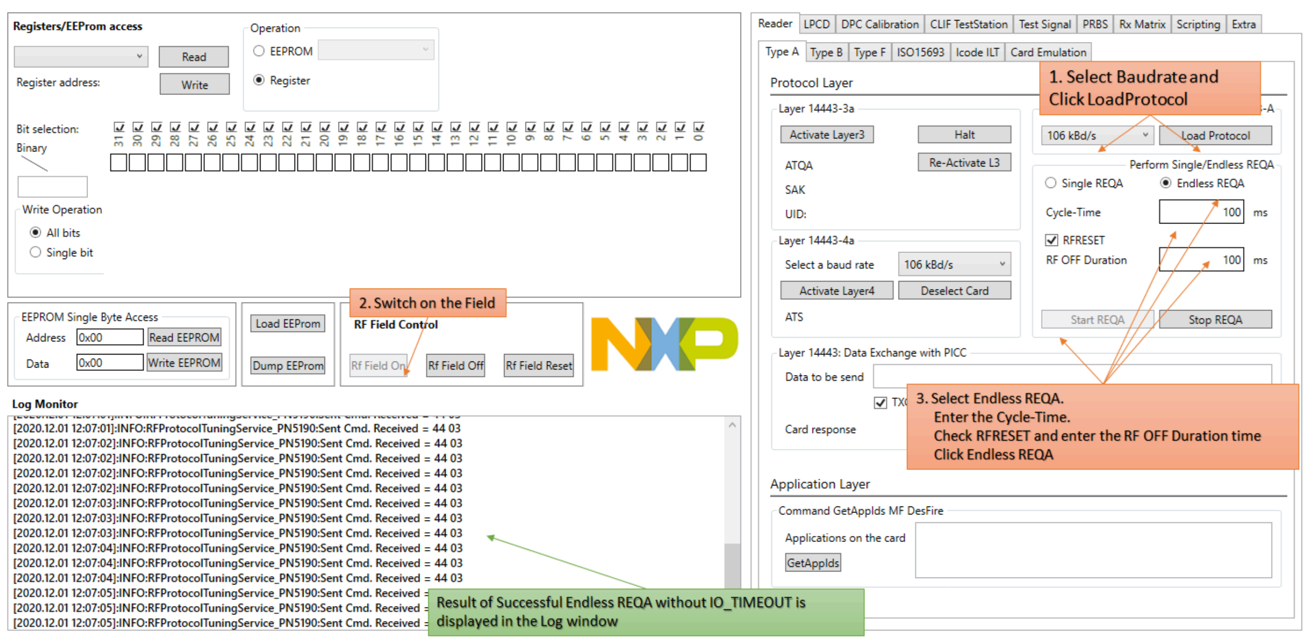



**Figure 63. Protocol Tuning**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

## 4.5 LPCD

The Low Power Card Detection (LPCD) allows saving battery charge during polling for NFC Counterparts like cards and mobile phones. In general, the low-power card detection provides a functionality, which allows to power down the reader for a certain amount of time to save energy. When a card is detected, the reader becomes active again to process the cards. If no card is detected, the reader goes back to the power down state. During the polling time, a host controller can be set to a power-saving mode. An interrupt request from the IC allows waking up the host controller in case an antenna detuning by a card or cell phone had been detected.

### 4.5.1 CLRC663

The NFC Cockpit allows the configuration and test of the Low Power Card Detection (LPCD) of the CLRC663 as shown in Figure 64.

The LPCD parameter, which is used to define the LPCD performance (sensitivity versus robustness) can be entered manually, if needed (details refer to [CLRC66302HN]).

Otherwise the standby time can be entered and the LPCD can be started. During the LPCD being activated, the CLRC663 does not react on any command, so only a detuning (-> place a card) or a Reset (press <Stop LPCD>) can end the LPCD mode.

***Note:*** *The NFC Cockpit automatically stops the LPCD after 60 seconds.*

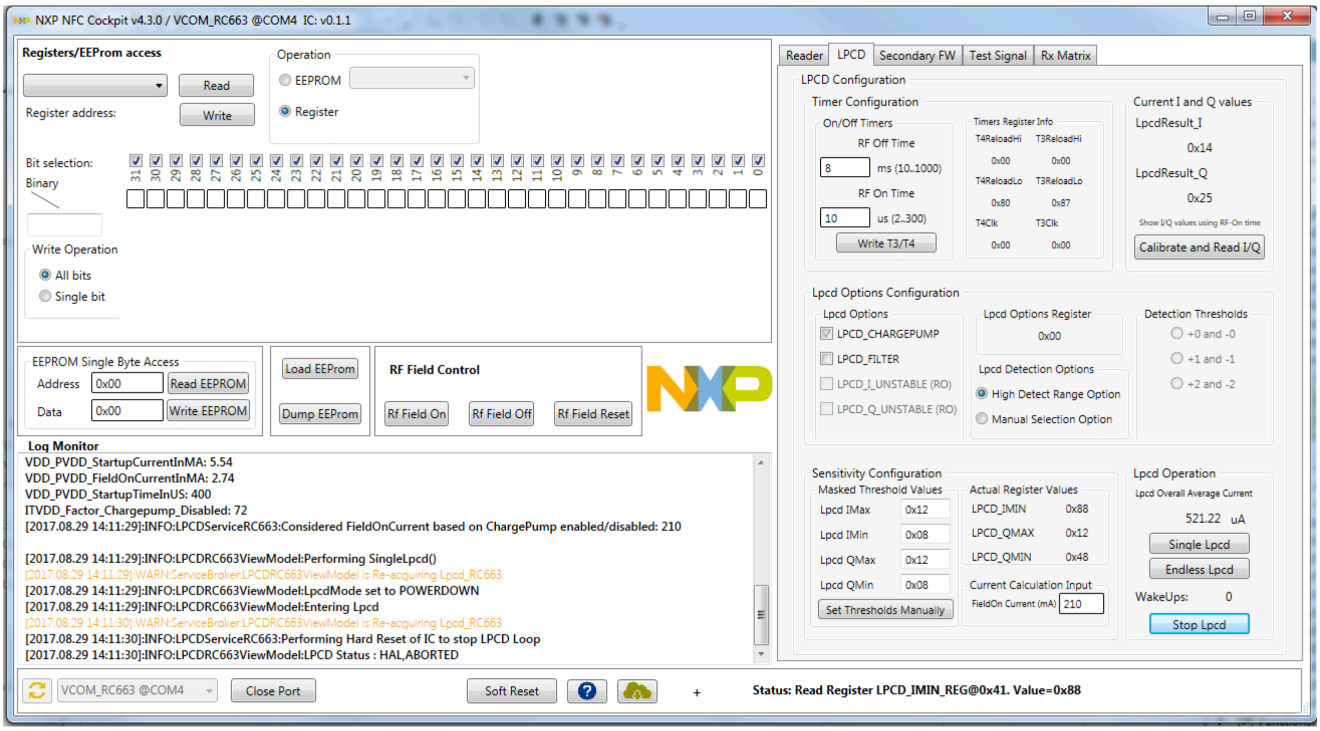



**Figure 64. CLR663 LPCD**



© 2023 NXP B.V. All rights reserved.

### 4.5.2 PN7462AU

LPCD PN7462AU runs continuously looking for a load on the antenna and notifies the user if the load crossed the threshold AGC.

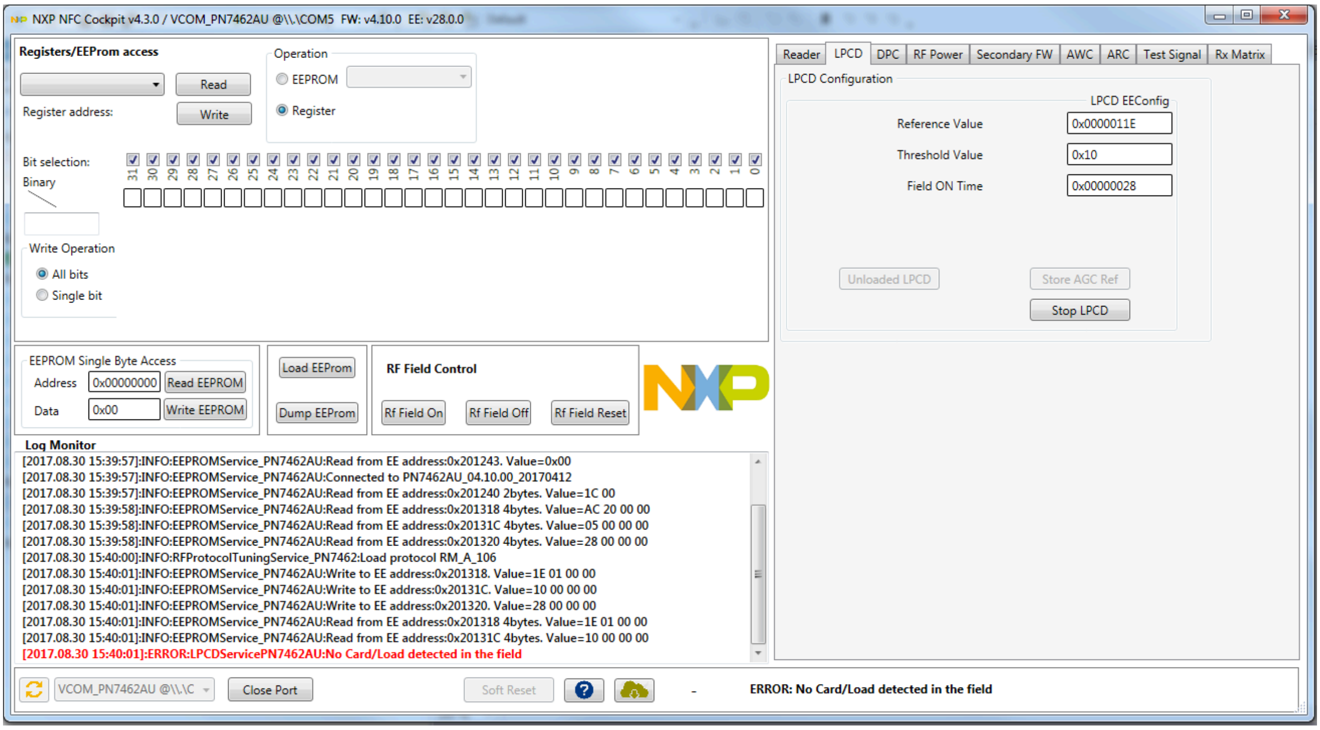



**Figure 65. PN7462AU Unloaded LPCD**

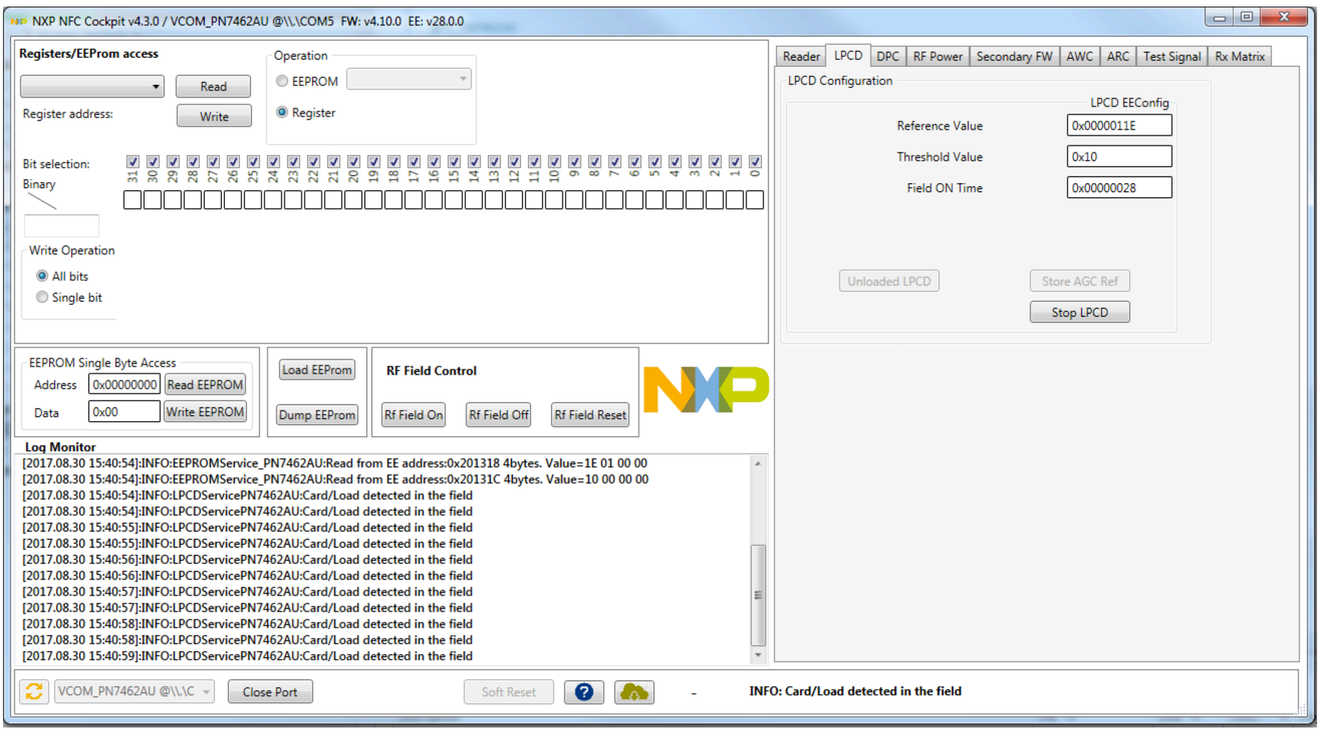



**Figure 66. PN7462 LPCD with Load**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.5.3 PN5180

**LPCD PN5180** Low-Power Card Detection operates in two modes.

- Auto Calibration: Takes Threshold, Gear number, and all other parameters from EEPROM
- Self-Calibration: Takes Gear number and all other parameters from registers

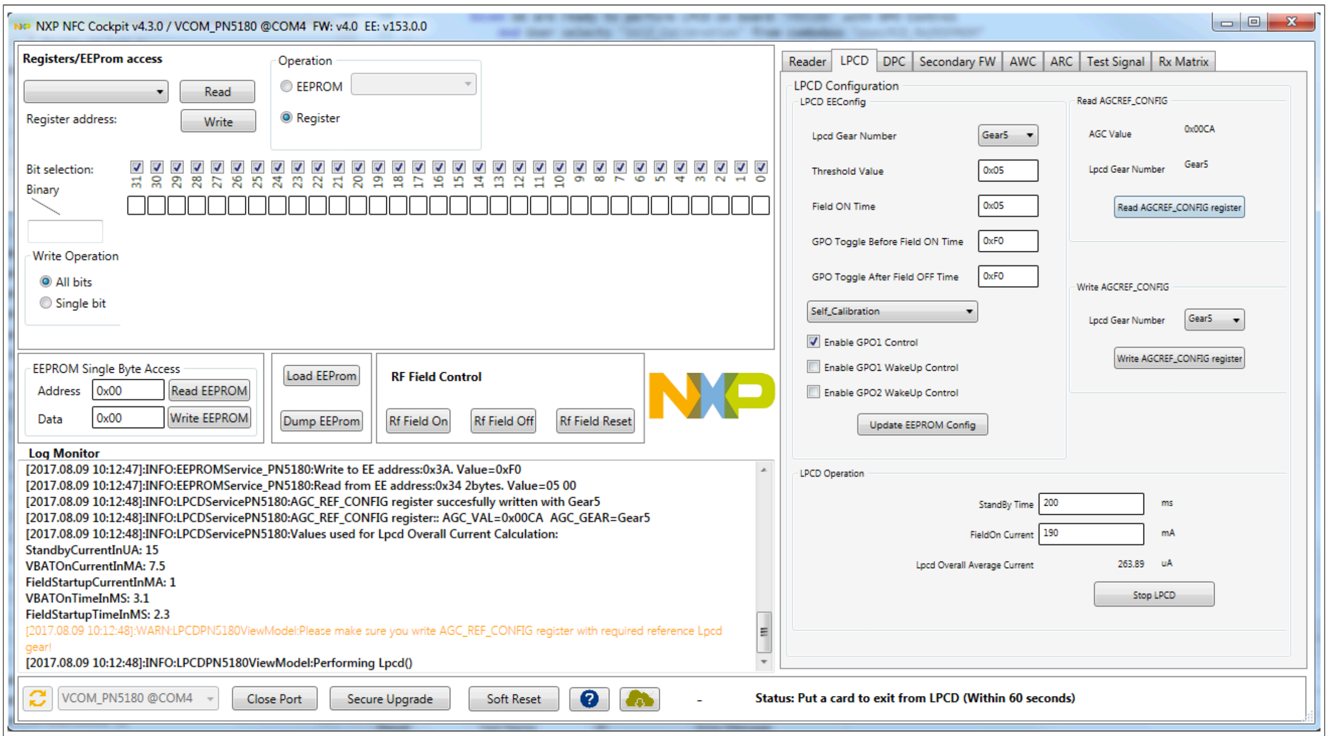



Figure 67. Performlpcd

### 4.5.4 PN5190

**LPCD PN5190** Low-Power Card Detection operates as mentioned below.

1. Semi-Autonomus LPCD
2. LPCD
3. ULPCD

#### 1. Semi-Autonomous LPCD

Below are the steps to perform Semi-Autonomous LPCD

- Update the Target and Hysteresis Register Settings.
- Perform Calibrate. During this phase, the Target and Hysteresis will be written to LPCD Calibrate Ctrl register with freeze bit set to zero.
- Perform Read I/Q Single or Endless. During this phase, the Calibration starts and can be observed from I and Q Register.
- Stop Read stops the endless read of I / Q channel values.
- Freeze Target and Hysteresis save the Target and Hysteresis values enter in Register settings to EEPROM.



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

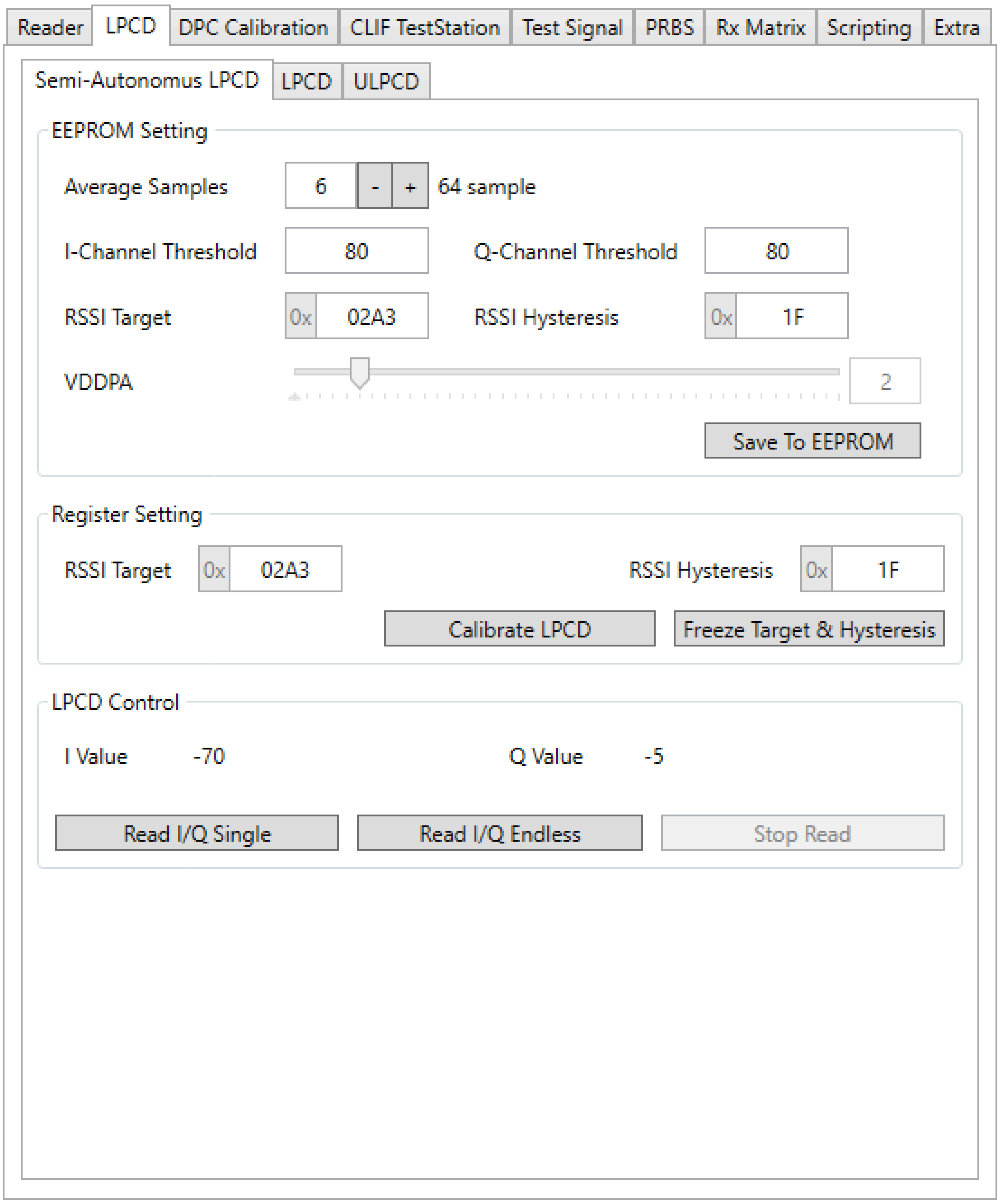



**Figure 68. Semi-Autonomous LPCD Without Load**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

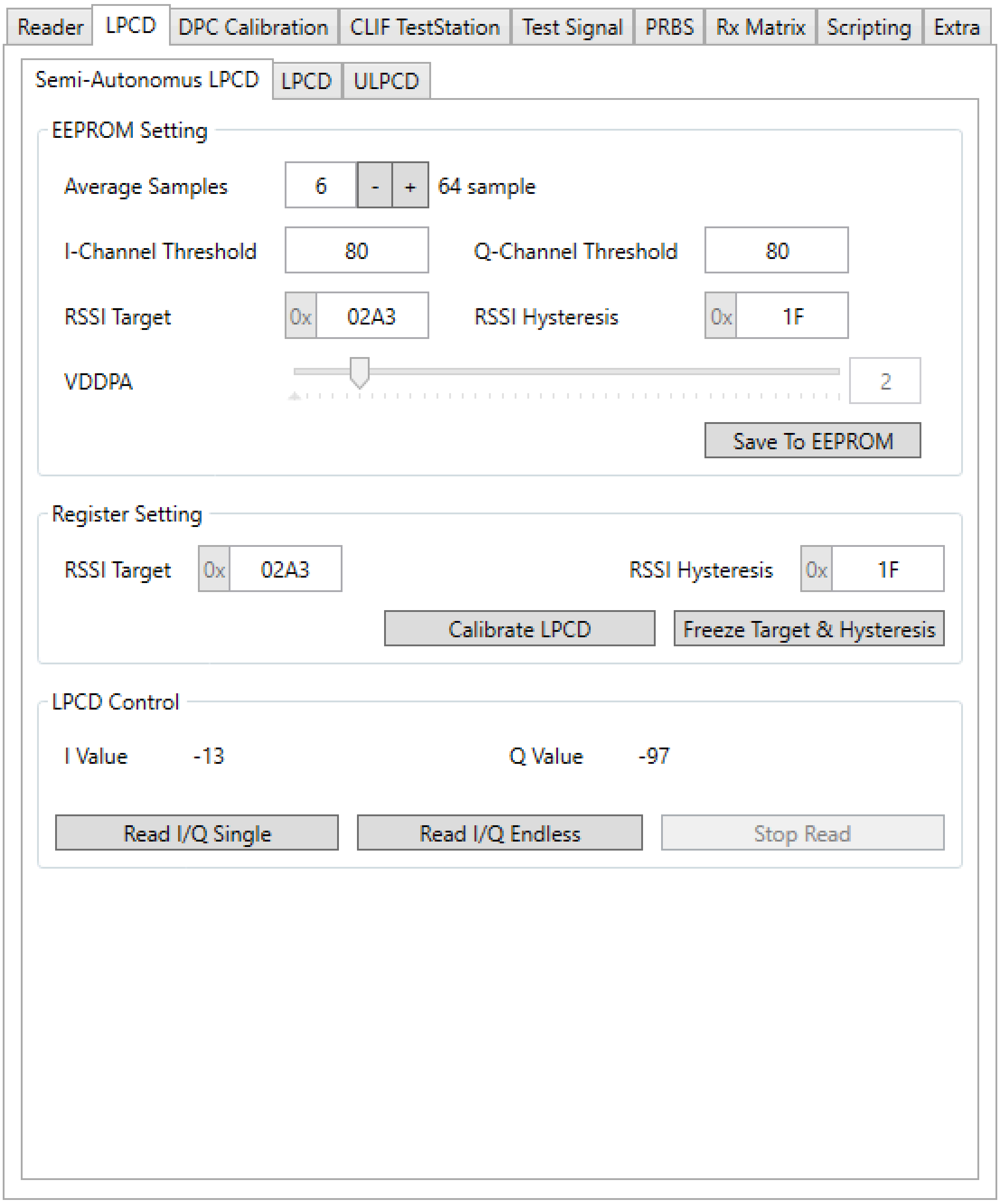



**Figure 69. Semi-Autonomous LPCD With Load**



© 2023 NXP B.V. All rights reserved.

## 2. LPCD

Below are the steps to perform LPCD

- Update the EEPROM settings.
- Perform Calibrate.
- Perform Single / Endless / Auto LPCD. The execution gets frozen until load change is detected or 60 s time-out is reached.
- For Endless LPCD, the reference is used only once and the loop will run endlessly. The number of load changes will be displayed as wake-up count.
- For Auto LPCD, the previous reference is taken to detect load change.
- Stop LPCD stops the Endless / Auto LPCD executions.
- Single Mode, Calibration, and detection of load change is taken care by the IC.

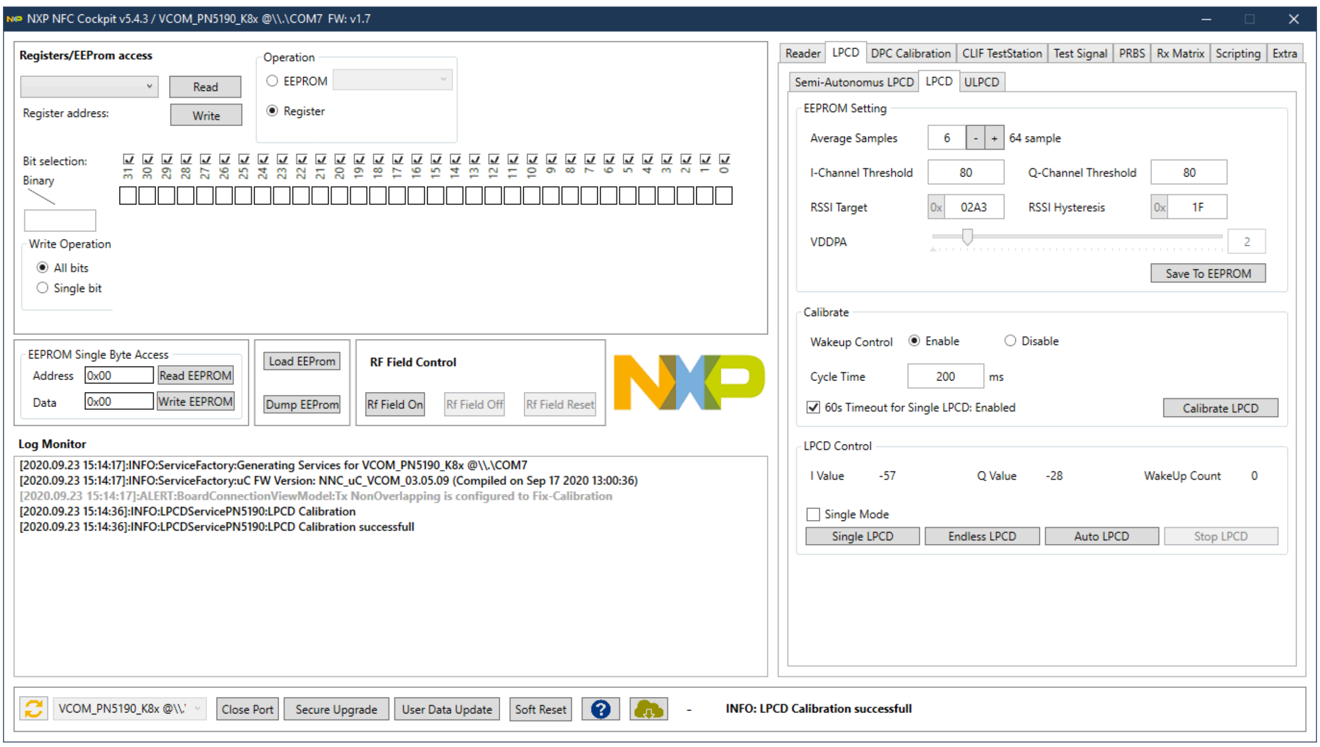



Figure 70. LPCD Calibration



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

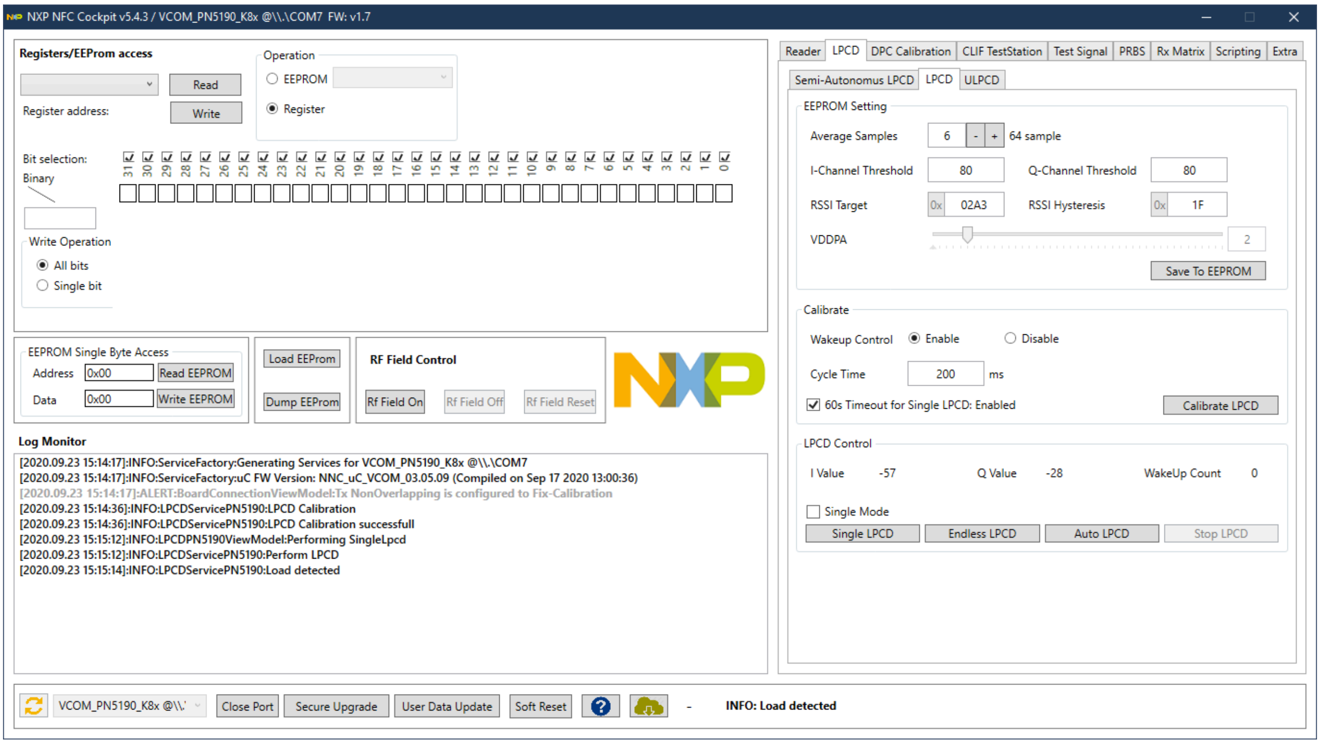



**Figure 71. LPCD Load Change Detection**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 3. ULPCD

Below are the steps to perform ULPCD

- Update the EEPROM settings.
- Perform Calibrate.
- Perform Single ULPCD. The execution gets frozen until load change is detected.
- Stop ULPCD stops the load change detection executions.

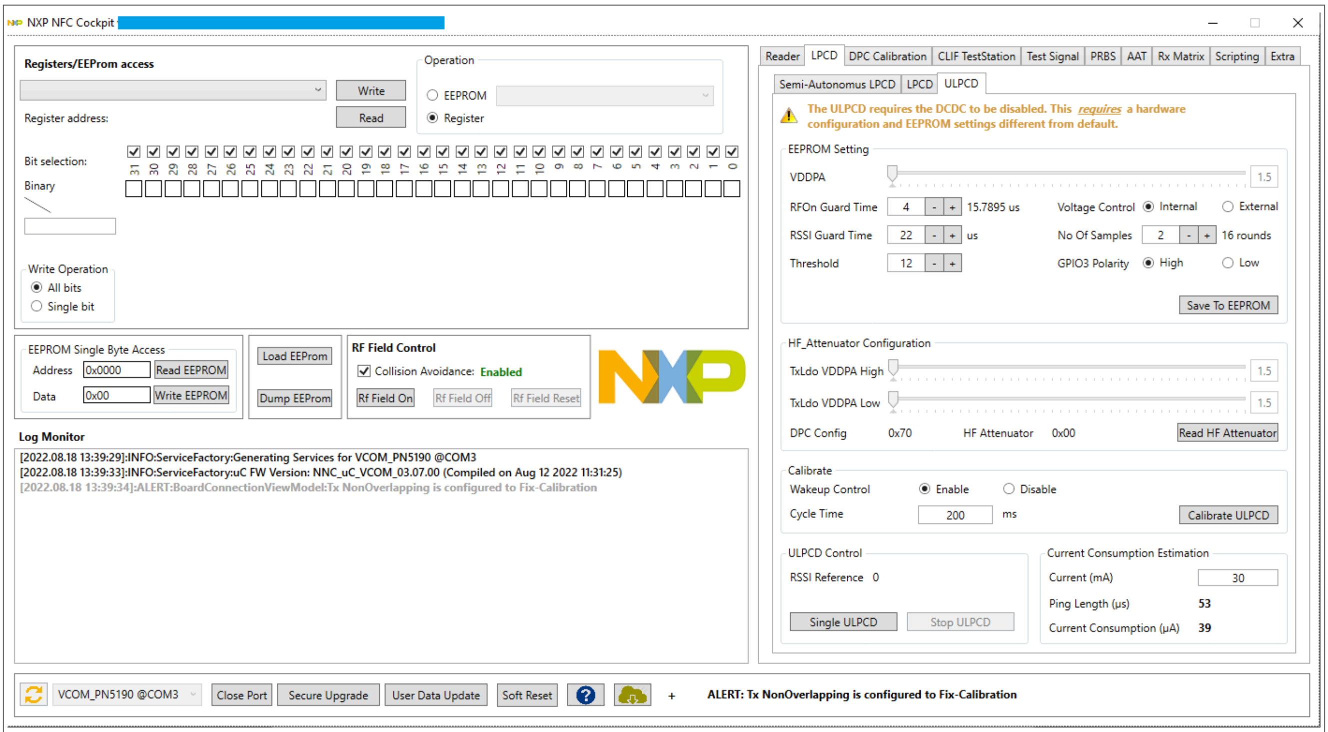



**Figure 72. ULPCD**

## 4.5.5 PN7642

**LPCD PN7642** Low-Power Card Detection operates in two modes.

Below are the steps to perform LPCD

- Update the EEPROM settings.
- Perform Calibrate.
- Perform Single / Endless / Auto LPCD. The execution gets frozen until load change is detected or 60 s time-out is reached.
- For Endless LPCD, the reference is used only once and the loop will run endlessly. The number of load changes are displayed as wake-up count.
- For Auto LPCD, the previous reference is taken to detect load change.
- Stop LPCD stops the Endless / Auto LPCD executions.
- Single Mode, Calibration, and detection of load change is taken care by the IC.



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

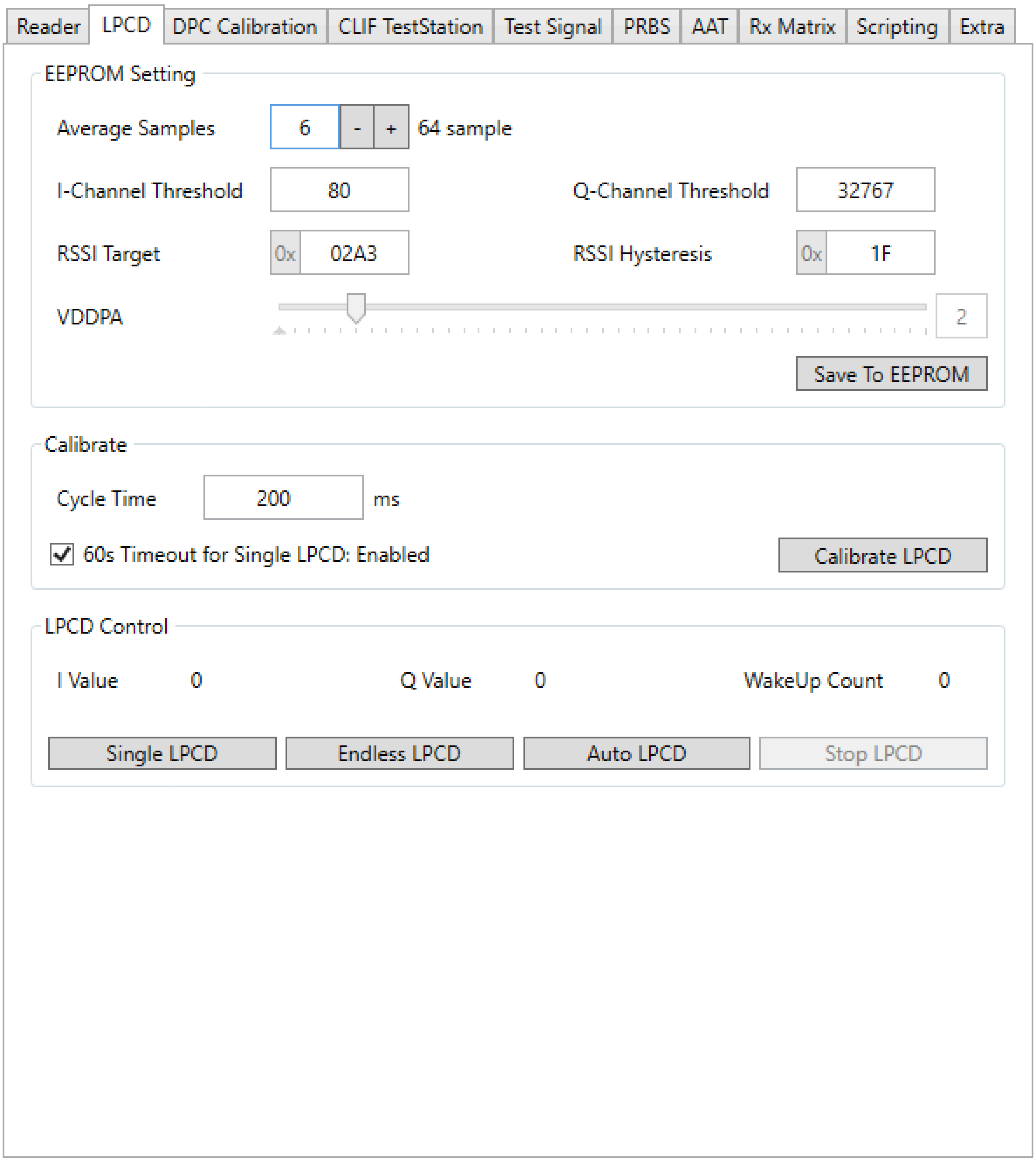



**Figure 73. Performlpcd**

## 4.6 DPC

Dynamic Power Control (DPC) is a mechanism used to avoid the IC from being hit by the sudden reverse current generated by the load on the antenna.



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.6.1 PN5180 / PN7462AU

Dynamic Power Control (DPC) is a mechanism used to avoid the IC from being hit by the sudden reverse current generated by the load on the antenna. It uses gears for the regulation of ITvdd depending on the AGC value.

#### 4.6.1.1 Correlation

**DPC Correlation** gives user the ability to understand the relation between ITvdd current and AGC value of the IC. It gives user the range of variation that occurs between an unloaded ITvdd and a maximum ITvdd with a user-defined step size.

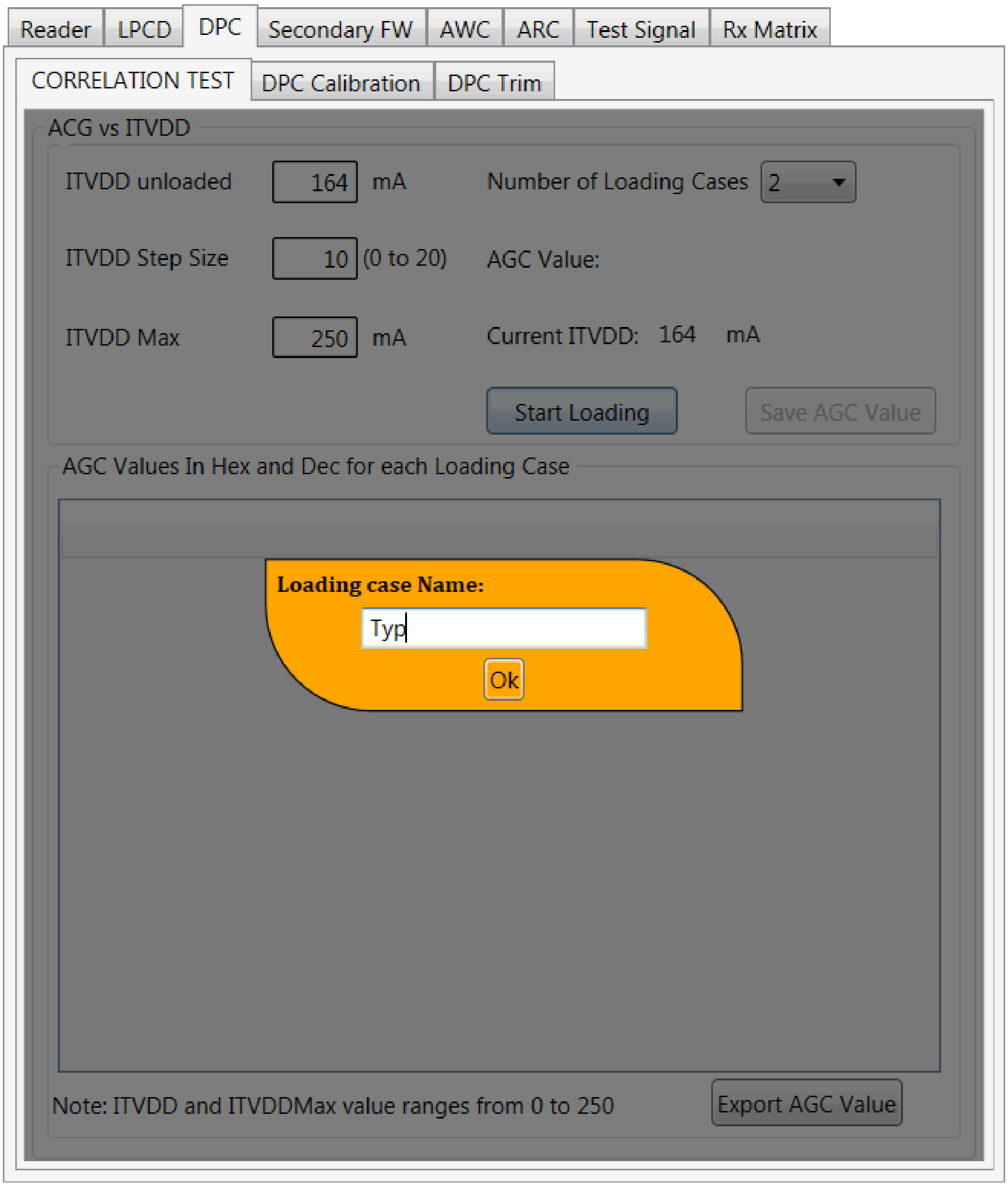



**Figure 74. DPC: Correlation: Loading Case**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

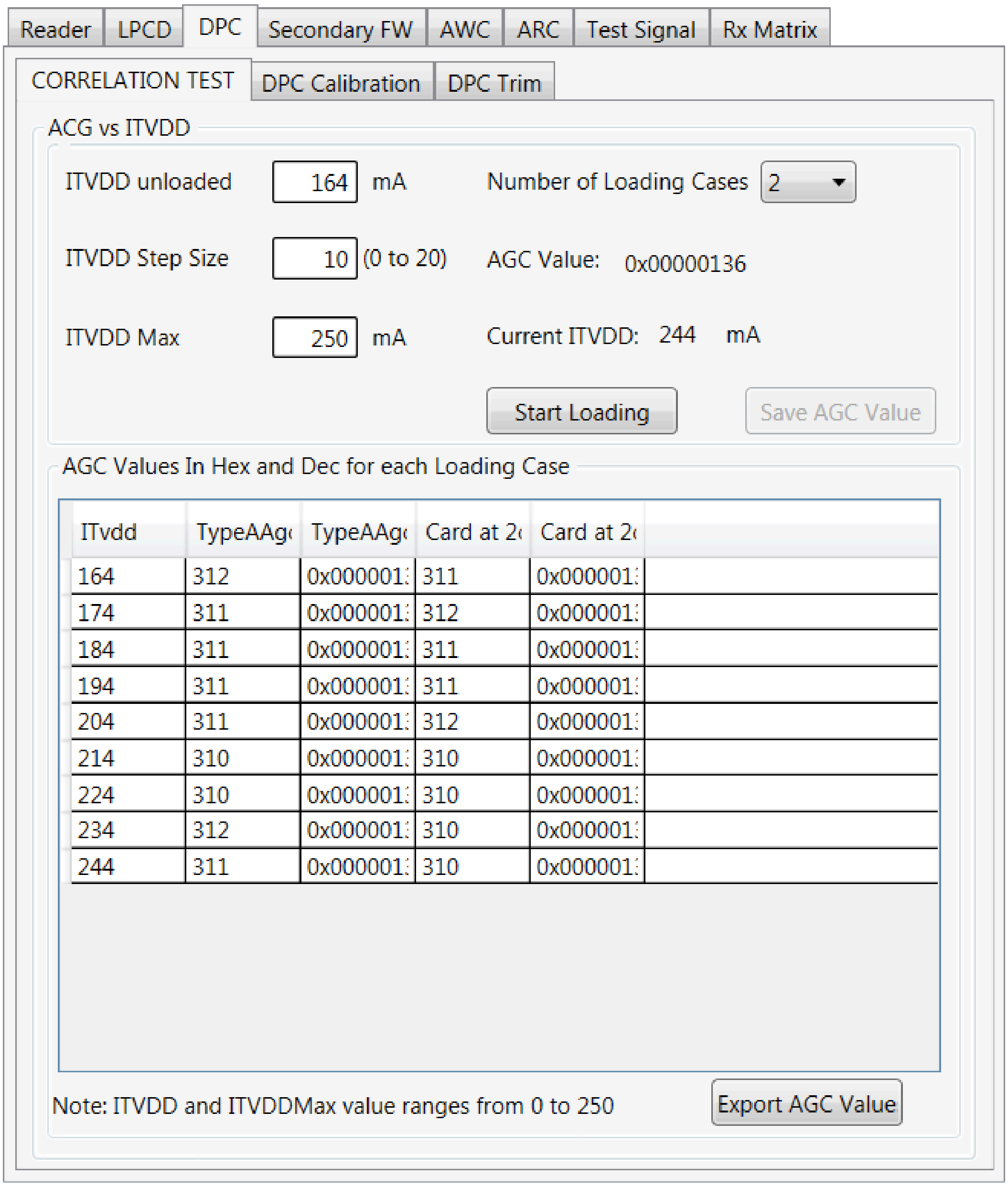



**Figure 75. DPC: Correlation Test**



© 2023 NXP B.V. All rights reserved.

### 4.6.1.2 Calibration

**DPC Calibration** gives user the ability to mark the boundaries for each DPC gear. This setting of threshold can be based on ITvdd limit and GearTx settings.

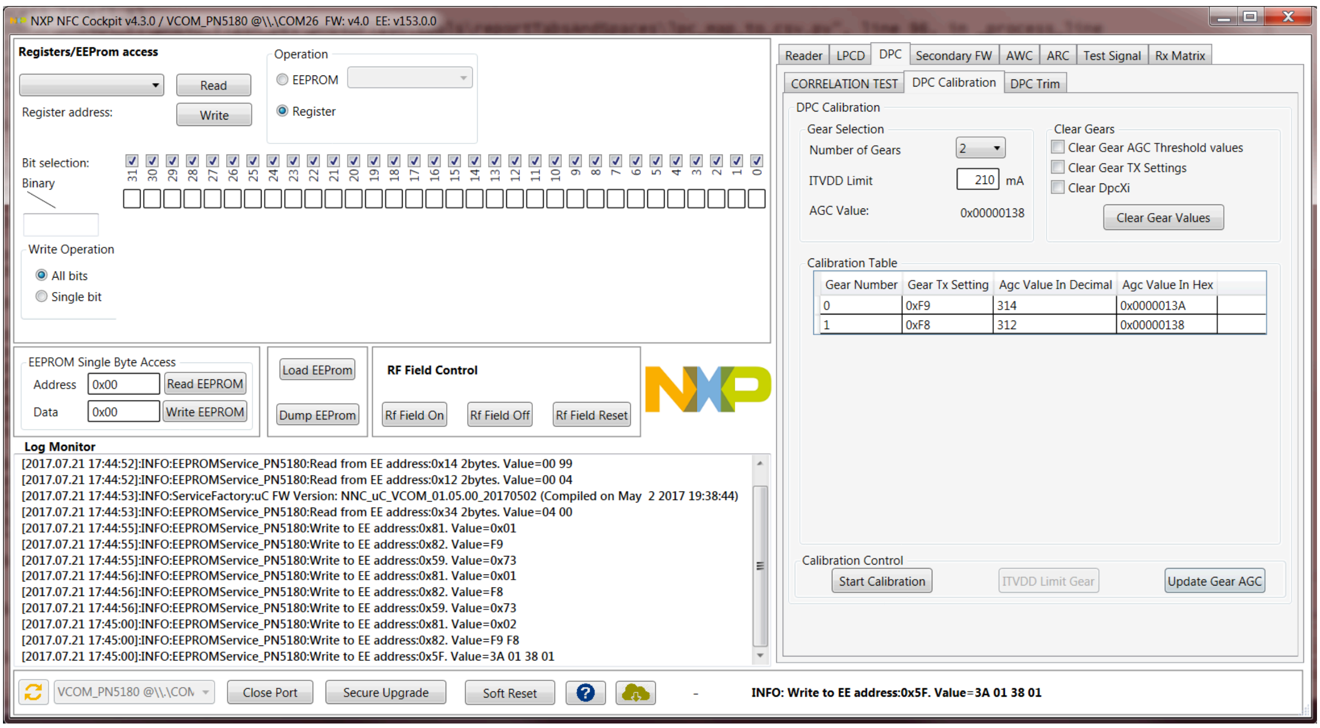



Figure 76. DPC: Calibration



© 2023 NXP B.V. All rights reserved.

#### 4.6.1.3 Trim

**DPC Trim** allows the user to randomly know the DPC gear at which the system is currently running. It also helps in calculating AGCXi value by giving AGC reference value as an input.

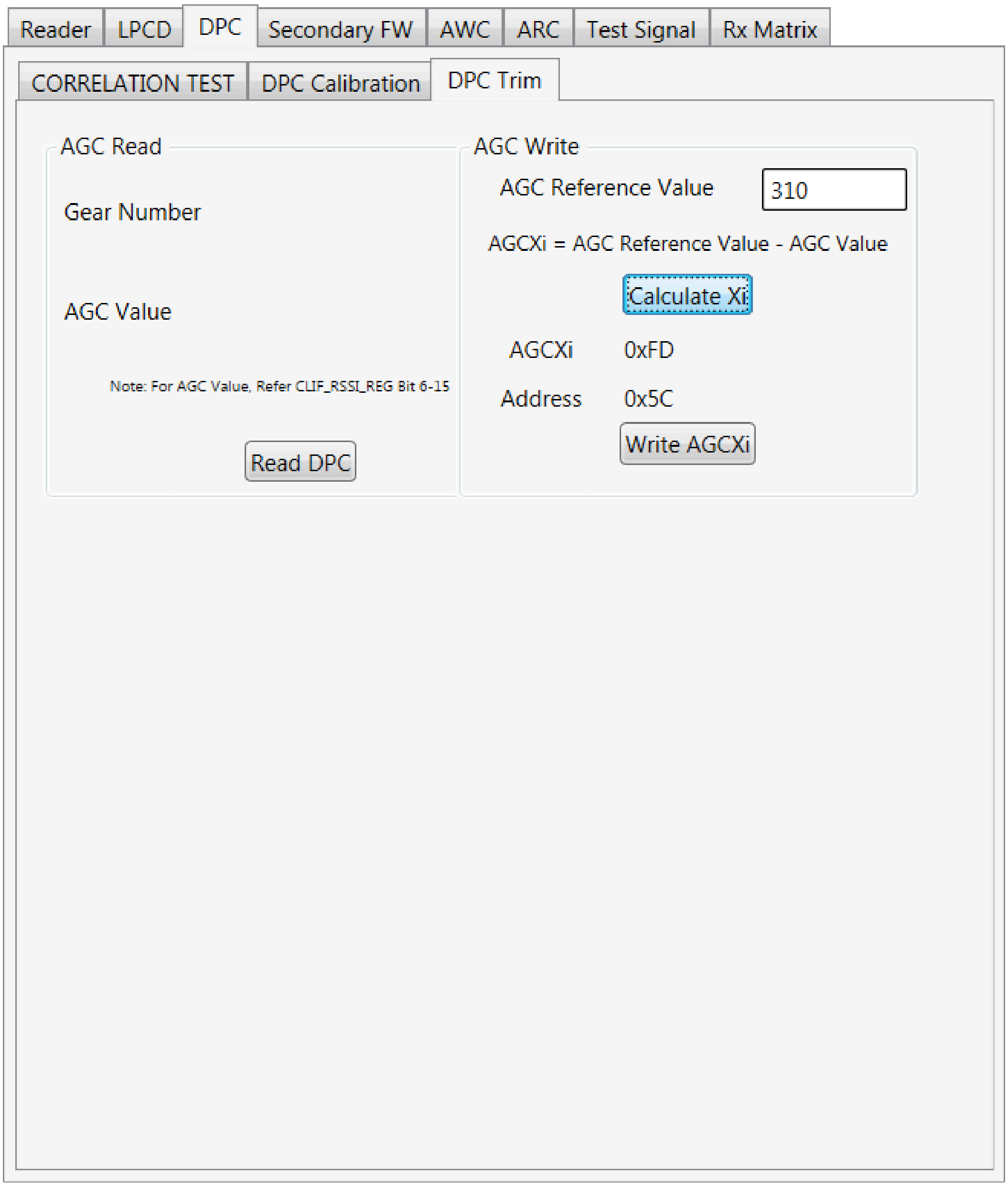



**Figure 77. DPC TRIM: Calculate AGC Xi**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.6.2 PN5190 / PN76XX

The below content describes the usage of DPC features for PN5190 and PN7640/PN7642 reader IC.

#### 4.6.2.1 Calibration

**DPC Calibration** allows the user to perform the following,

- Perform Load Protocol. Here the protocol for TypeA 106 will be loaded and will turn on the field.
- Updates the VDPPA step by step to know the Current being consumed.
- Updates the Hysteresis Loading / UnLoading and Target Current to be used.
- Computes the Current Reduction values for each VDDPA based on 8 measured ITVDD's.

##### 4.6.2.1.1 Perform the below steps to save 8 fixed points.

1. Move **VDDPA Slider** for respective VDDPA's available in the table.
2. Enter **ITVDD** for the set VDPPA.
3. Click **Compute and Move to LUT** for Computation of Current Reduction and updating the table available in Current Reduction tab.
4. Click **Clear button** to clear all the updated ITVDD values.

***Note:*** *When Compute and Move to LUT is pressed, the process will not save the computed Current Reduction to EEPROM. It will update the Table available in Current Reduction tab.*


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

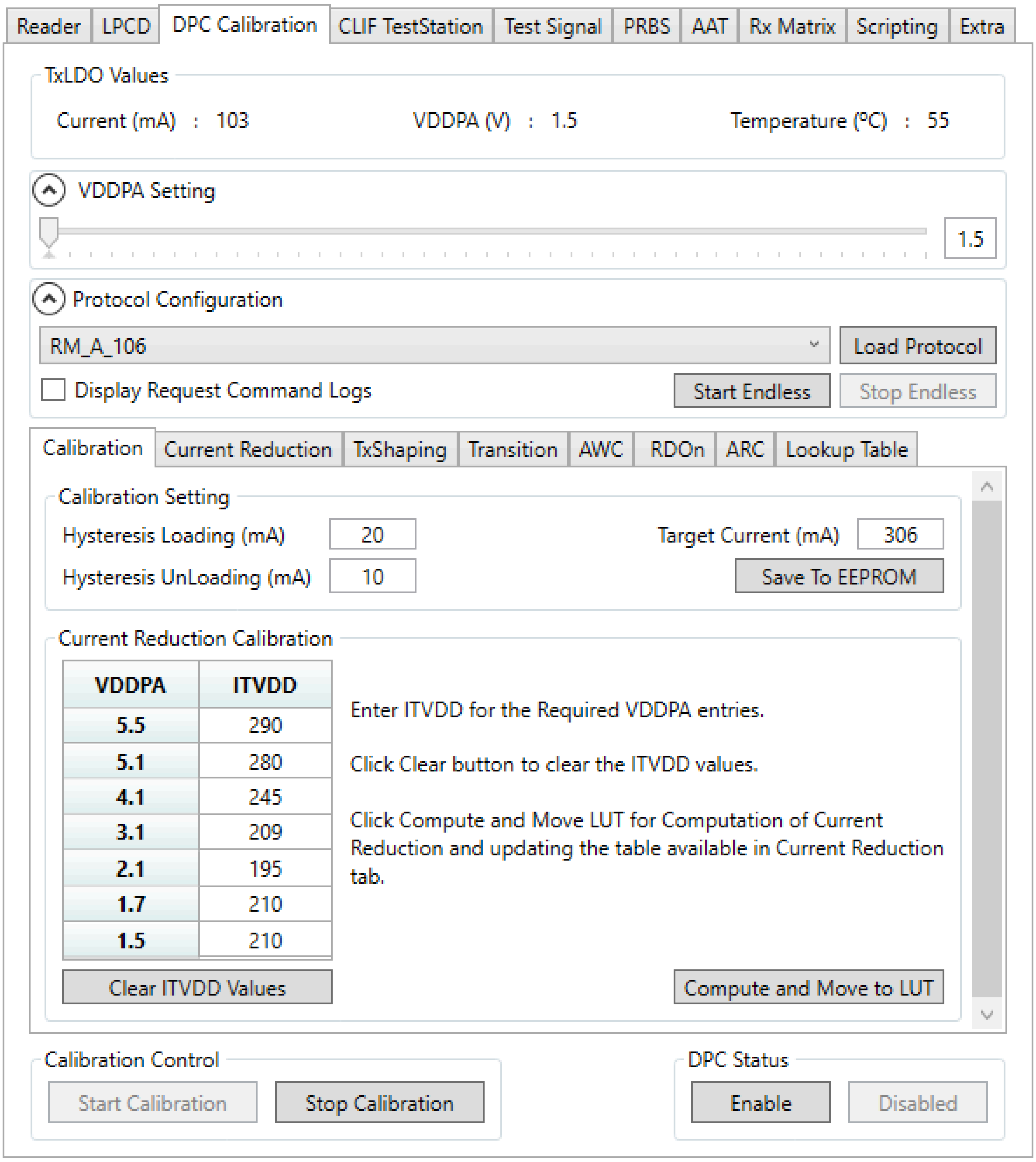



**Figure 78. Calibration**

### 4.6.2.2 Current Reduction

**DPC CurrentReduction** allows the user to configure Reduction Current per Vddpa. This can be achieved as mentioned below.

- Updates the VDPPA step by step to know the Current being consumed.
- Update the Reduction Current for each VDDPA entry.



© 2023 NXP B.V. All rights reserved.

- Auto fill Reduction current updates the lower Current Reduction values with the Current Reduction value that is set for current VDDPA.
- Clear EEPROM Entries clear the EEPROM entries of all the VDDPA's. Once it clears, the table also resets the values to zero.
- Save EEPROM Save the Current Reduction entries to the EEPROM.

***Note:*** *It is very important to save the settings before switching to other DPC Features.*

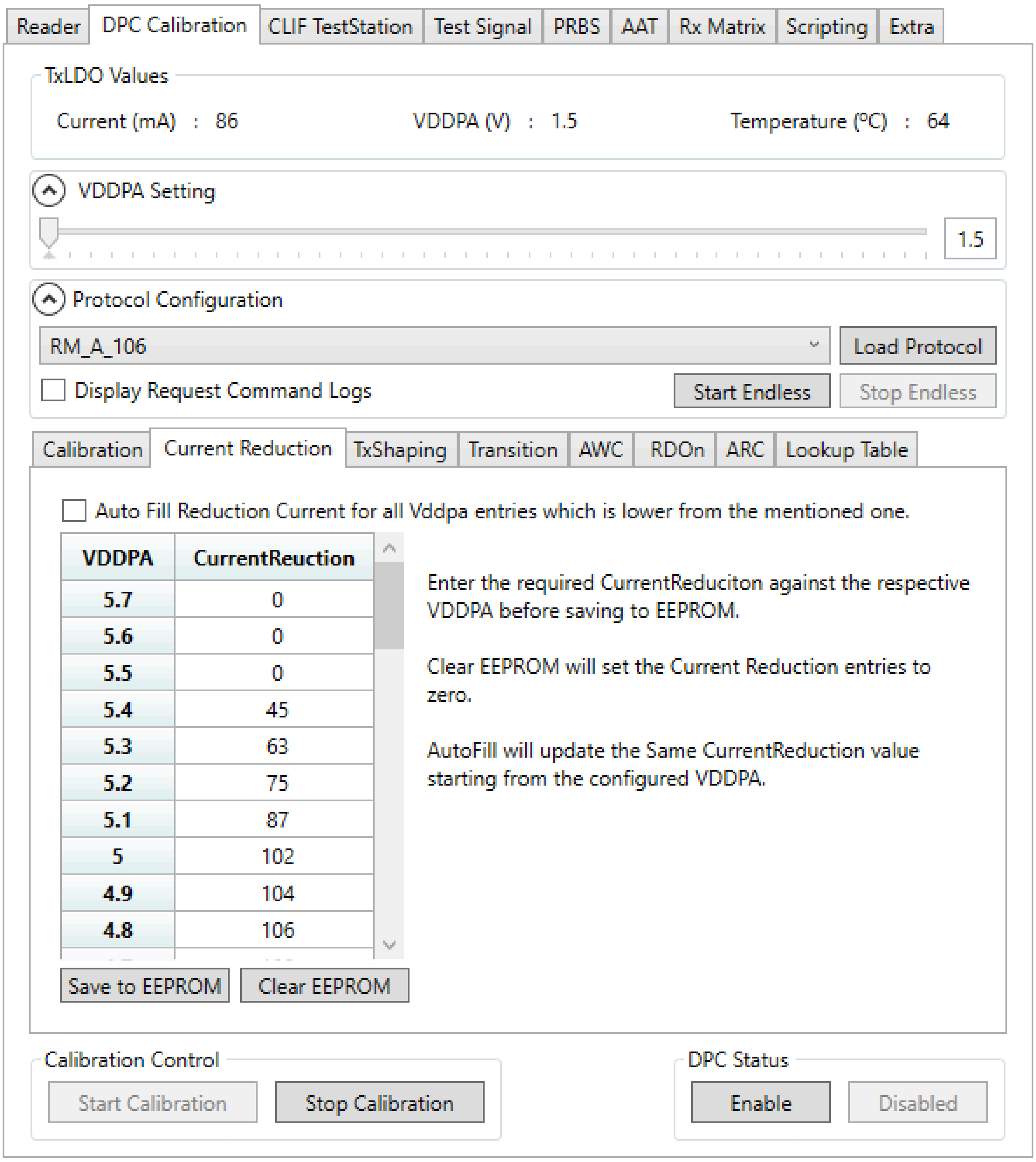



**Figure 79. DPC: CurrentReduction without Auto Fill**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

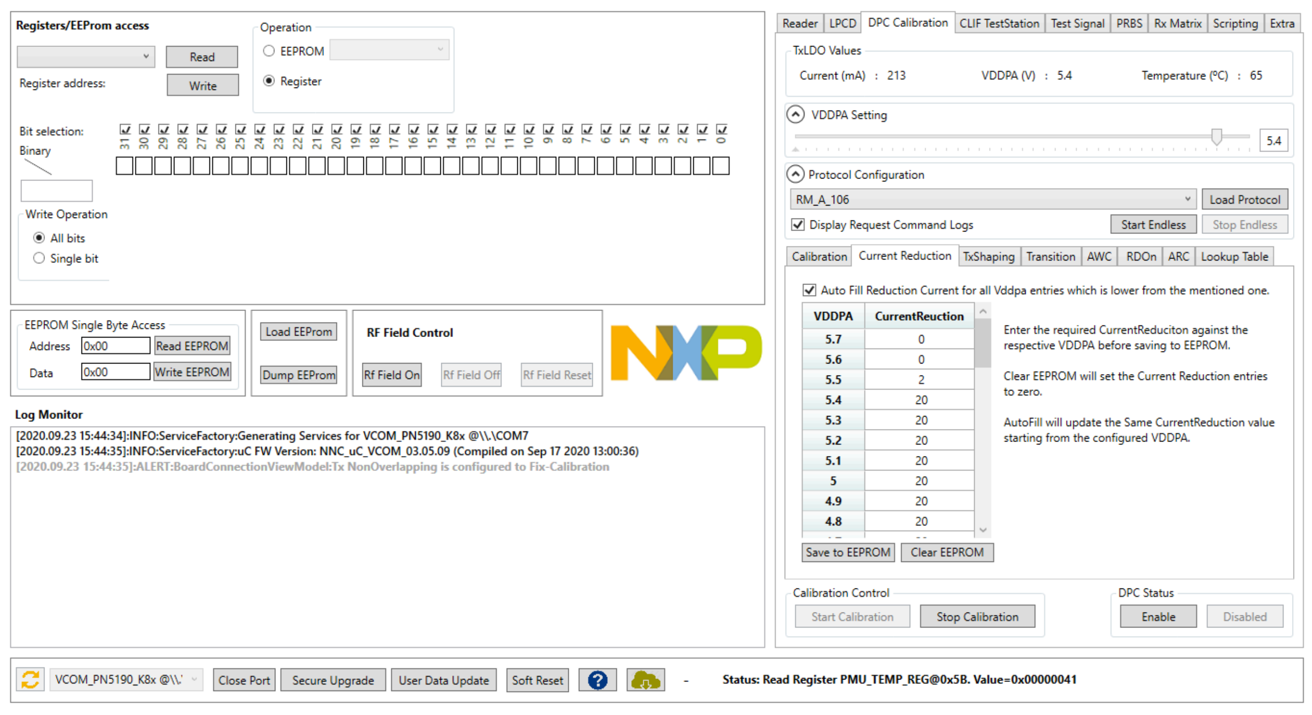



**Figure 80. DPC: CurrentReduction with Auto Fill**

**The Update of Current Reduction can be done using two ways.**

- With 8 fixed ITVDD values that are entered and the remaining are auto calculated. Refer to Section 4.6.2.1.
- Save individual Current Reduction value against each VDDPA.
  - Set the VDDPA.
  - Enter the CurrentReduction value and click Save button.
  - Perform the above two steps for all the VDPPA entries.

### 4.6.2.3 TxShaping

**DPC TxShaping** allows the user to modify the waveform for supported technologies. This is done as mentioned below,

- Select one of the technologies from the drop-down list and click LoadProtocol button. The field will turn on after click of this button.
- Click StartEndless button. On click of the button, continuous Request command starts.
- Configure the CRO (Oscilloscope) to view the waveform of the select technology for the Request command transmission.
- Vary the respective controls to view the change in the waveform.
- Once done with the settings, click Save To EEPROM. This saves the current controls values to EEPROM for the selected technology.
- Click of Clear EEPROM button sets the EEPROM contents and the controls to zero.

***Note:*** *It is very important to save the settings before moving to other technology.*



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

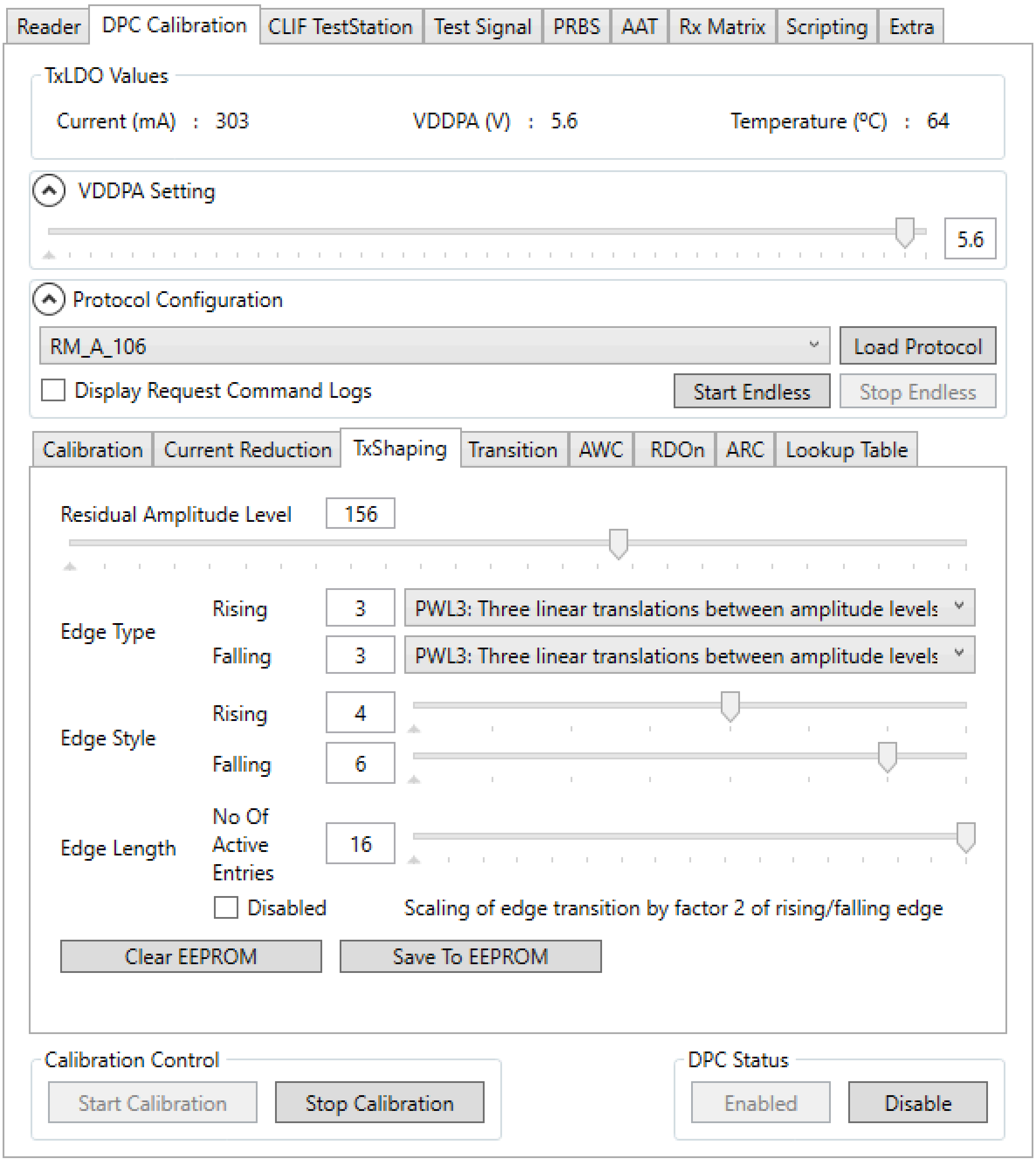



**Figure 81. TxShaping**



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

### 4.6.2.4 Look Up

This feature displays the complete EEPROM entries related to TxShaping, DPC, and ARC that are modified in other features.

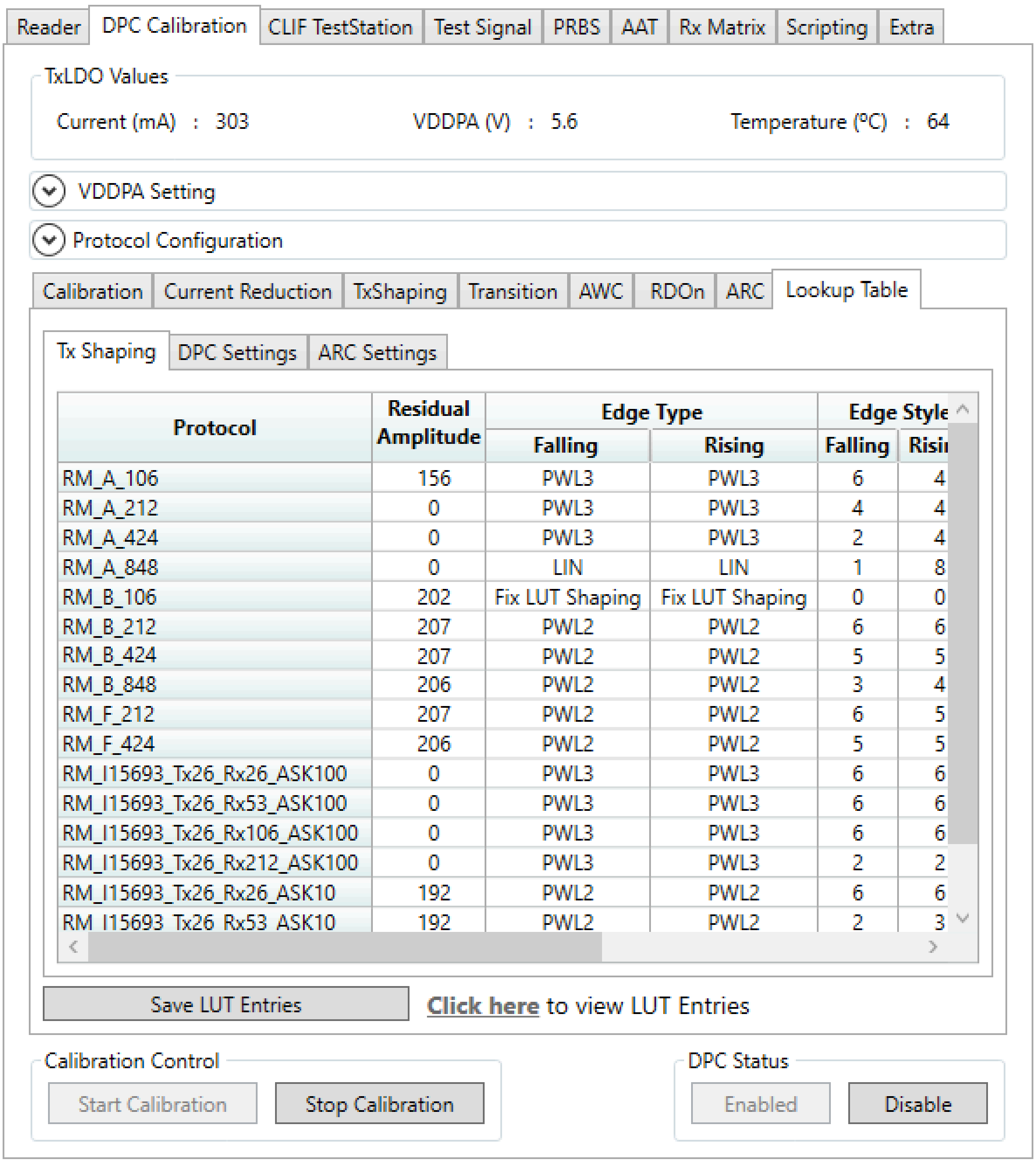
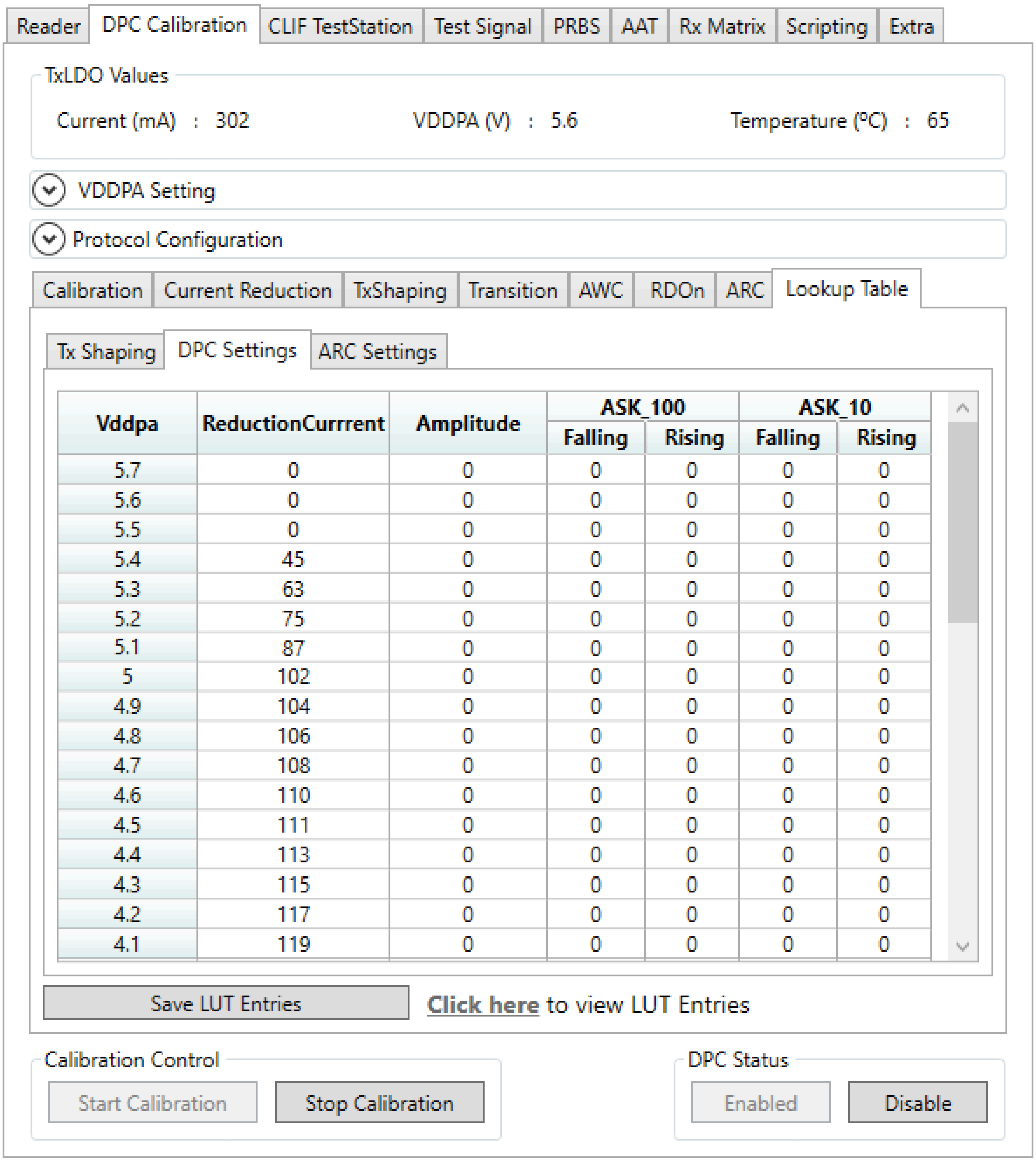



Figure 82. DPC: TxShaping



© 2023 NXP B.V. All rights reserved.




| Vddpa | ReductionCurrrent | Amplitude | ASK_100 | | ASK_10 | |
|---|---|---|---|---|---|---|
| | | | Falling | Rising | Falling | Rising |
| 5.7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5.6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5.4 | 45 | 0 | 0 | 0 | 0 | 0 |
| 5.3 | 63 | 0 | 0 | 0 | 0 | 0 |
| 5.2 | 75 | 0 | 0 | 0 | 0 | 0 |
| 5.1 | 87 | 0 | 0 | 0 | 0 | 0 |
| 5 | 102 | 0 | 0 | 0 | 0 | 0 |
| 4.9 | 104 | 0 | 0 | 0 | 0 | 0 |
| 4.8 | 106 | 0 | 0 | 0 | 0 | 0 |
| 4.7 | 108 | 0 | 0 | 0 | 0 | 0 |
| 4.6 | 110 | 0 | 0 | 0 | 0 | 0 |
| 4.5 | 111 | 0 | 0 | 0 | 0 | 0 |
| 4.4 | 113 | 0 | 0 | 0 | 0 | 0 |
| 4.3 | 115 | 0 | 0 | 0 | 0 | 0 |
| 4.2 | 117 | 0 | 0 | 0 | 0 | 0 |
| 4.1 | 119 | 0 | 0 | 0 | 0 | 0 |

**Figure 83. DPC: DPC Entries**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

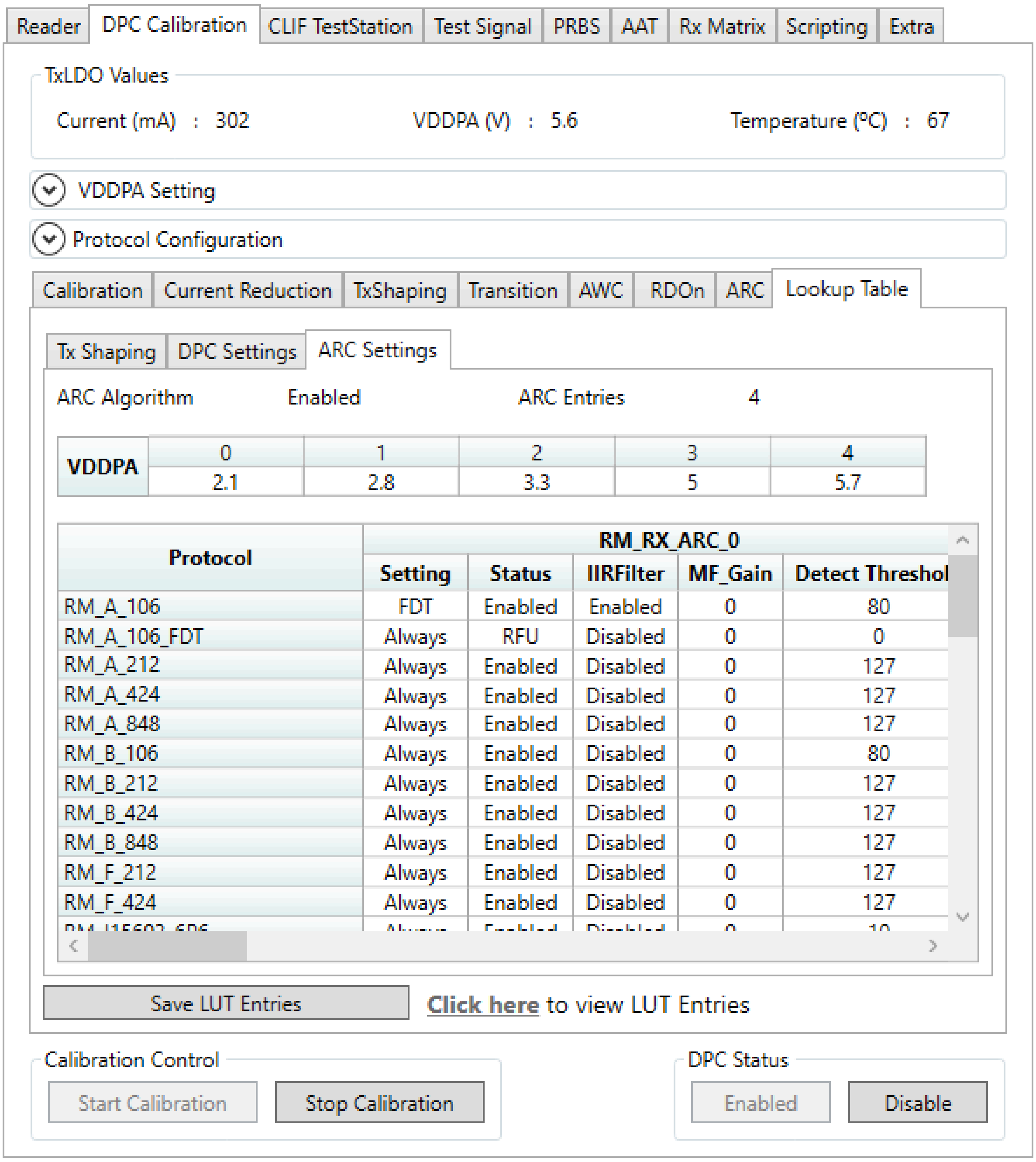



**Figure 84. DPC: ARC Entries**

### 4.6.2.4.1 Transition

**DPC Transition** allows the user to modify each byte of Transition registers. This is done as mentioned below,

- Select one of the technologies from the drop-down list and click LoadProtocol button. Upon LoadProtocol, the Registers will be read and will be updated in the respective UI elements.
- Click **+** to increment the value and **-** to decrement value. Here the value will auto update to zero if it has reached 255 which is maximum value of a byte.



© 2023 NXP B.V. All rights reserved.

- On press of **+ / -** button, the value is written to particular byte of the register.
- Read Registers reads the current values of the Rise / Fall values from the register and update the UI controls.

***Note:*** *The above process is same for Tx1 / Tx2 and Rise / Fall.*

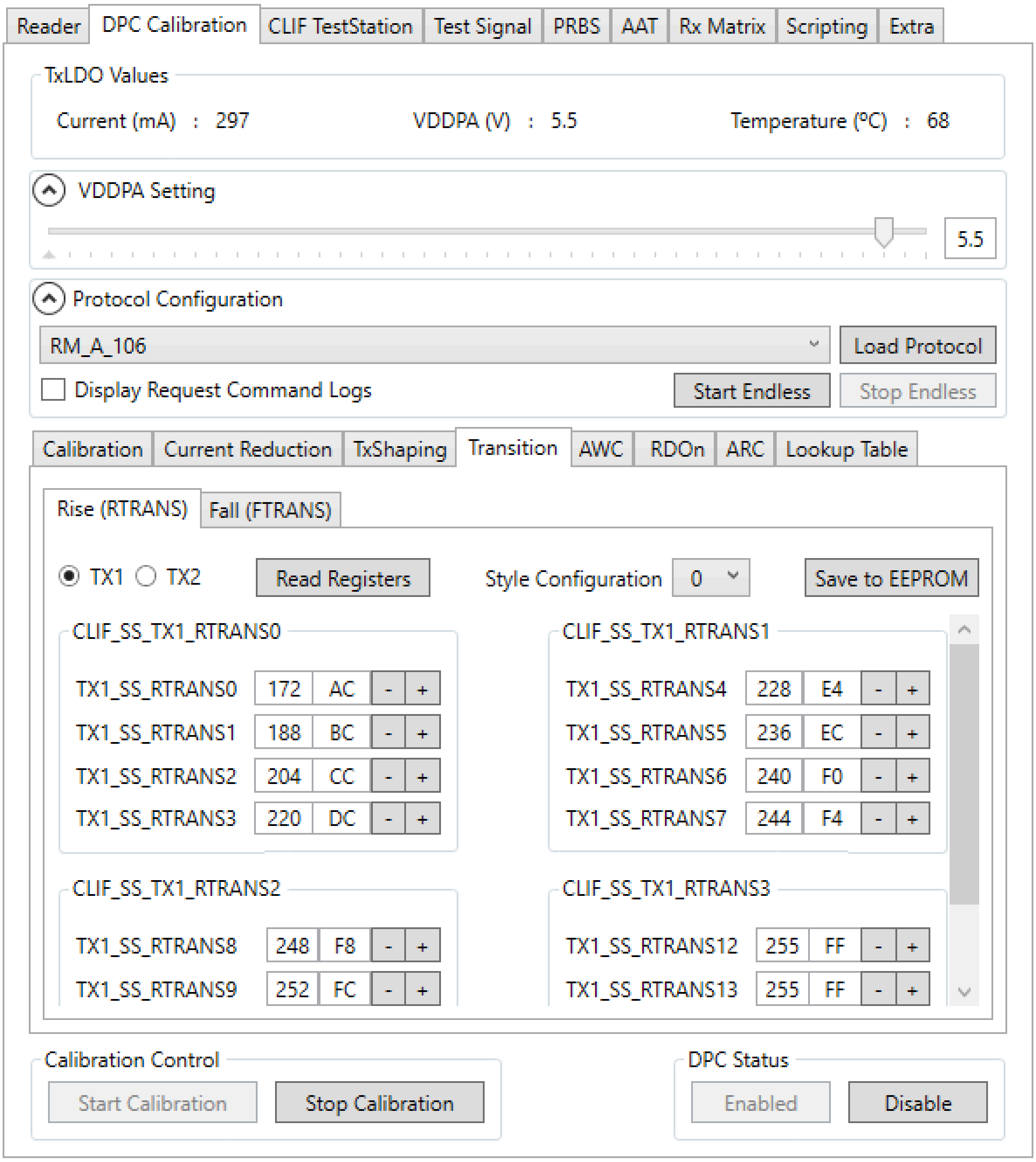



**Figure 85. Transition**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

## 4.7 AWC

**AWC (Automatic Waveshape Control)** enables the user to modify the waveshape of the transmitter signal using the help of sliders. It supports all protocols and helps in generating an AWC configuration string. This configuration string written to EEPROM is used by controller firmware to handle waveshape dynamically depending on the transmission channel load.

### 4.7.1 PN5180 / PN7462AU

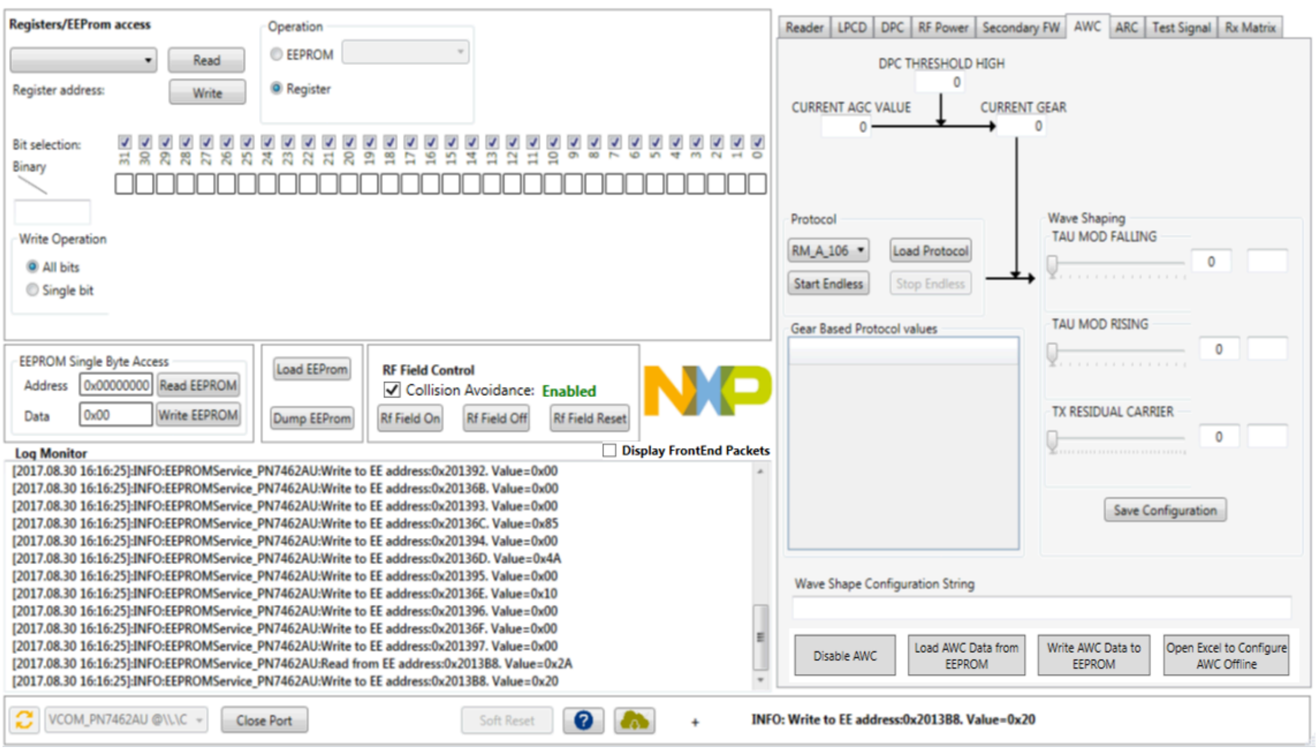


Figure 86. Load AWC String from EEPROM



© 2023 NXP B.V. All rights reserved.

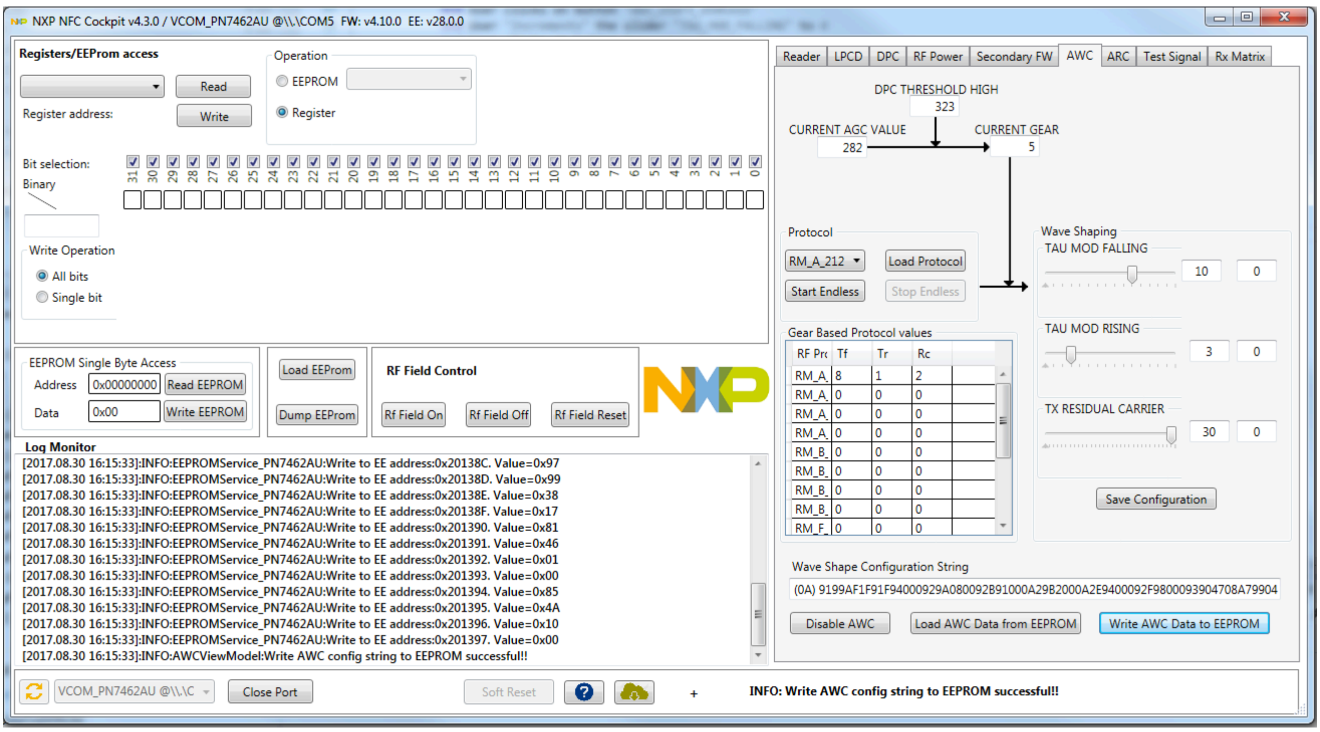



**Figure 87. Write AWC String**

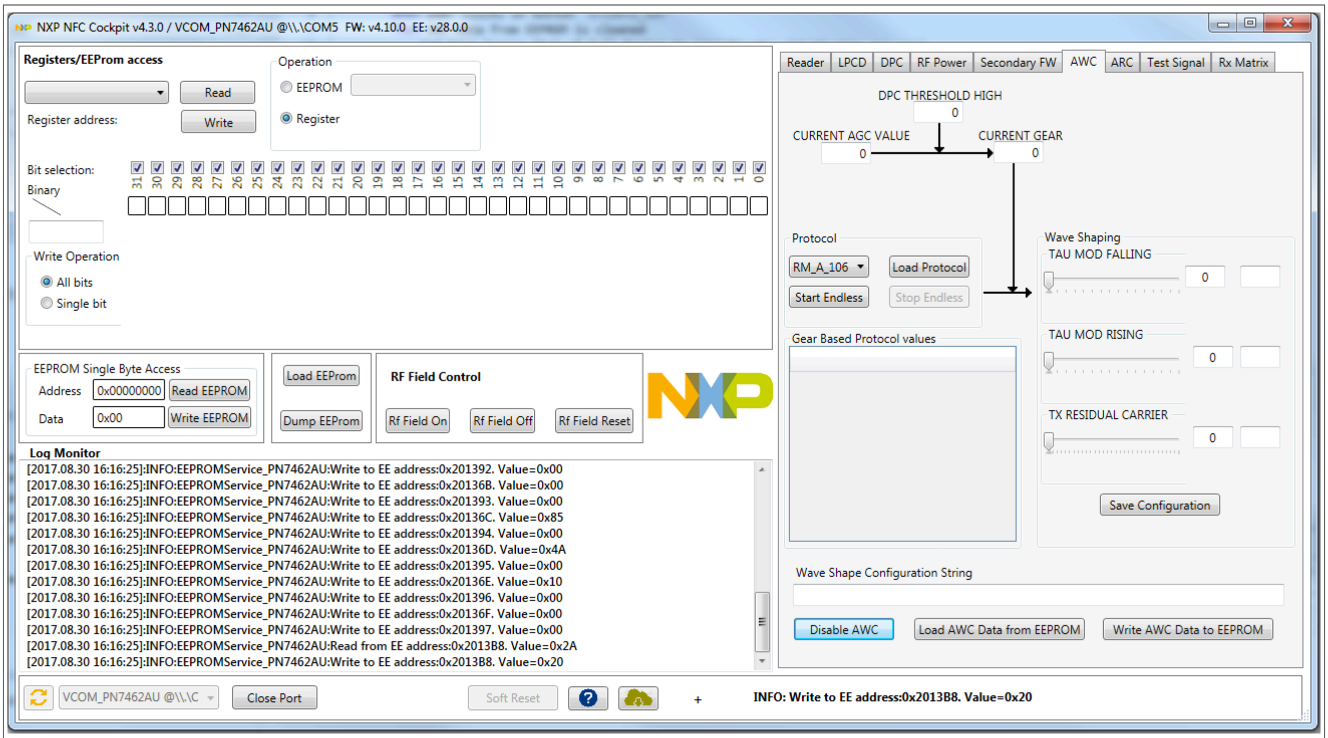



**Figure 88. Disable AWC**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

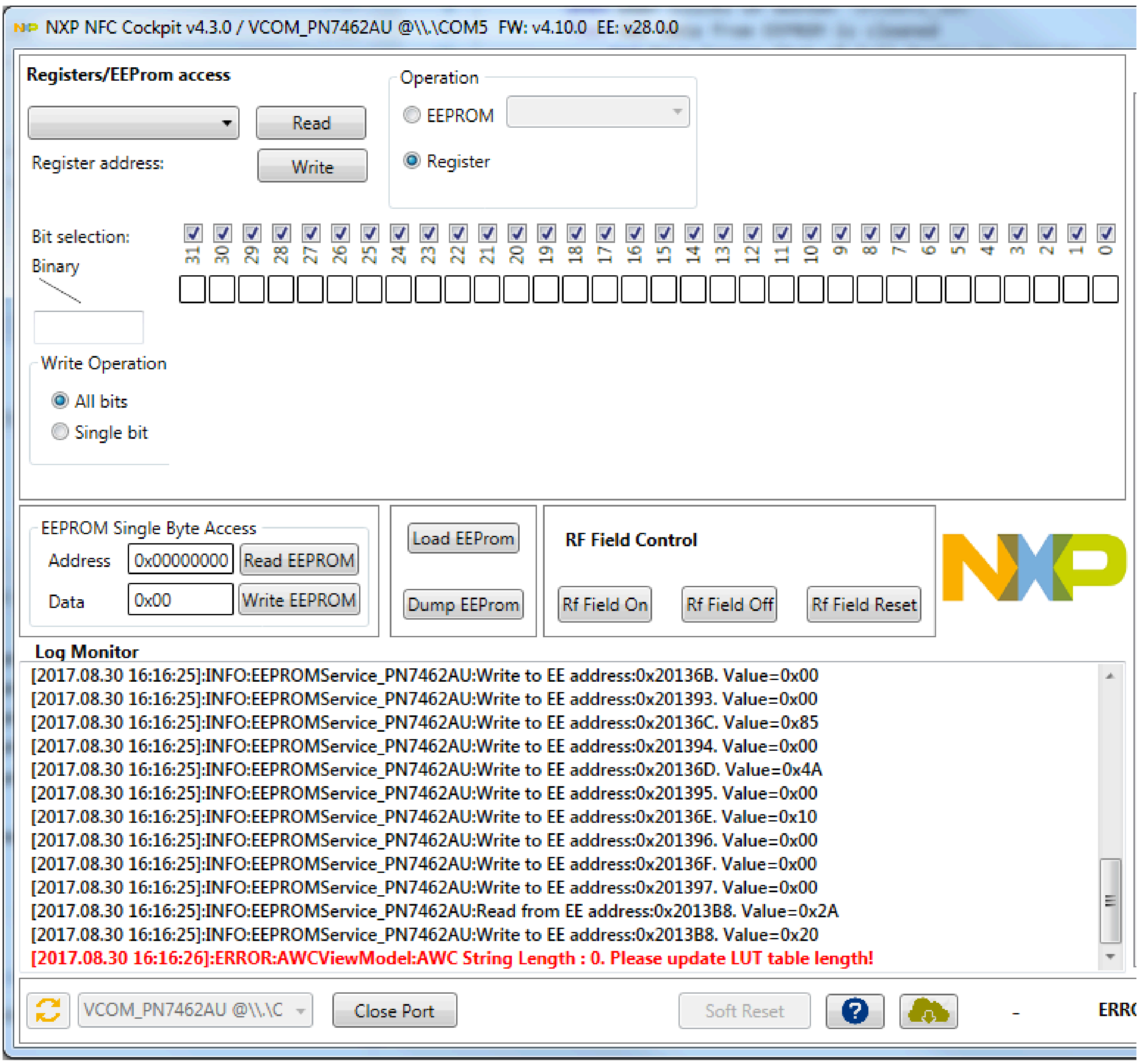



Figure 89. AWC String Zero

### 4.7.2 PN5190 / PN76XX

**AWC** allows the user to further modify the waveform for supported technologies. This is done as mentioned below,

- Select one of the technologies from the drop-down list and click LoadProtocol button. The field will turn on after click of this button.
- Click StartEndless button. On click of the button, continuous Request command will start.
- Configure the CRO (Oscilloscope) to view the waveform of the select technology for the Request command transmission.
- Vary the respective controls to view the change in the waveform.



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

- Once done with the settings, click Save To EEPROM. This saves the current controls values to EEPROM for the selected VDDPA.
- Click of Clear EEPROM button sets the EEPROM contents and the controls to zero.

***Note:*** *It is very important to save the settings before moving to other technology.*

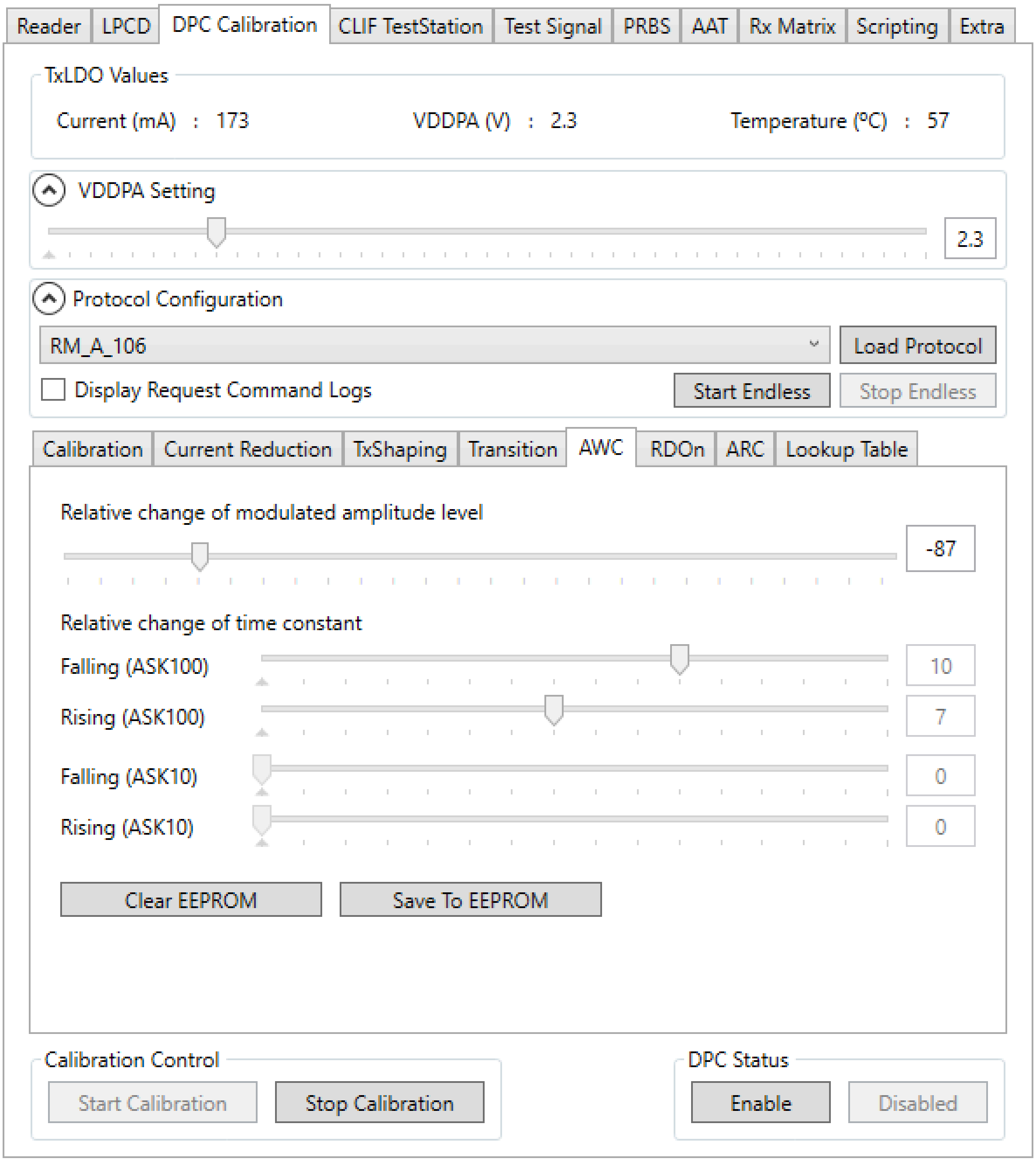



**Figure 90. AWC**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

## 4.8 ARC

**ARC (Automatic Receiver Control)** enables the user to modify the waveshape of the receiver signal using the help of sliders. It supports all protocols and helps in generating a configuration string. This configuration string written to EEPROM is used by controller firmware to handle waveshape dynamically depending on the receiver channel load.

### 4.8.1 PN5180 / PN7462AU

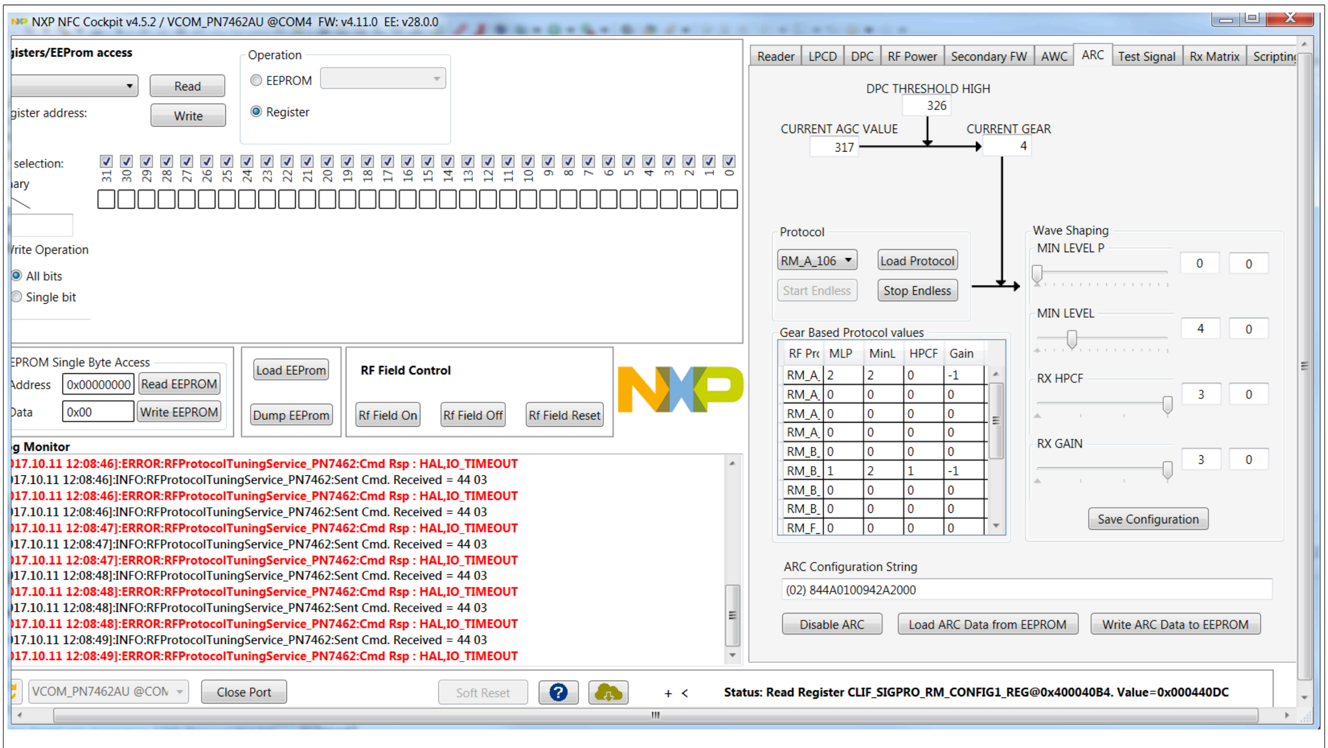



**Figure 91. ARC Save Config**

### 4.8.2 PN5190 / PN76XX

**ARC** allows the user to further modify the waveform for supported technologies. This is done as mentioned below,

- Select one of the technologies from the drop-down list and click LoadProtocol button. The field will turn on after click of this button.
- Click StartEndless button. On click of the button, continuous Request command will start.
- Configure the CRO (Oscilloscope) to view the waveform of the select technology for the Request command transmission.
- Vary the respective controls to view the change in the waveform.
- Once done with the settings, click Save To EEPROM. This saves the current controls values to EEPROM for the selected VDDPA.
- Click of Clear EEPROM button sets the EEPROM contents and the controls to zero.

***Note:*** *It is very important to save the settings before moving to other technology.*



© 2023 NXP B.V. All rights reserved.

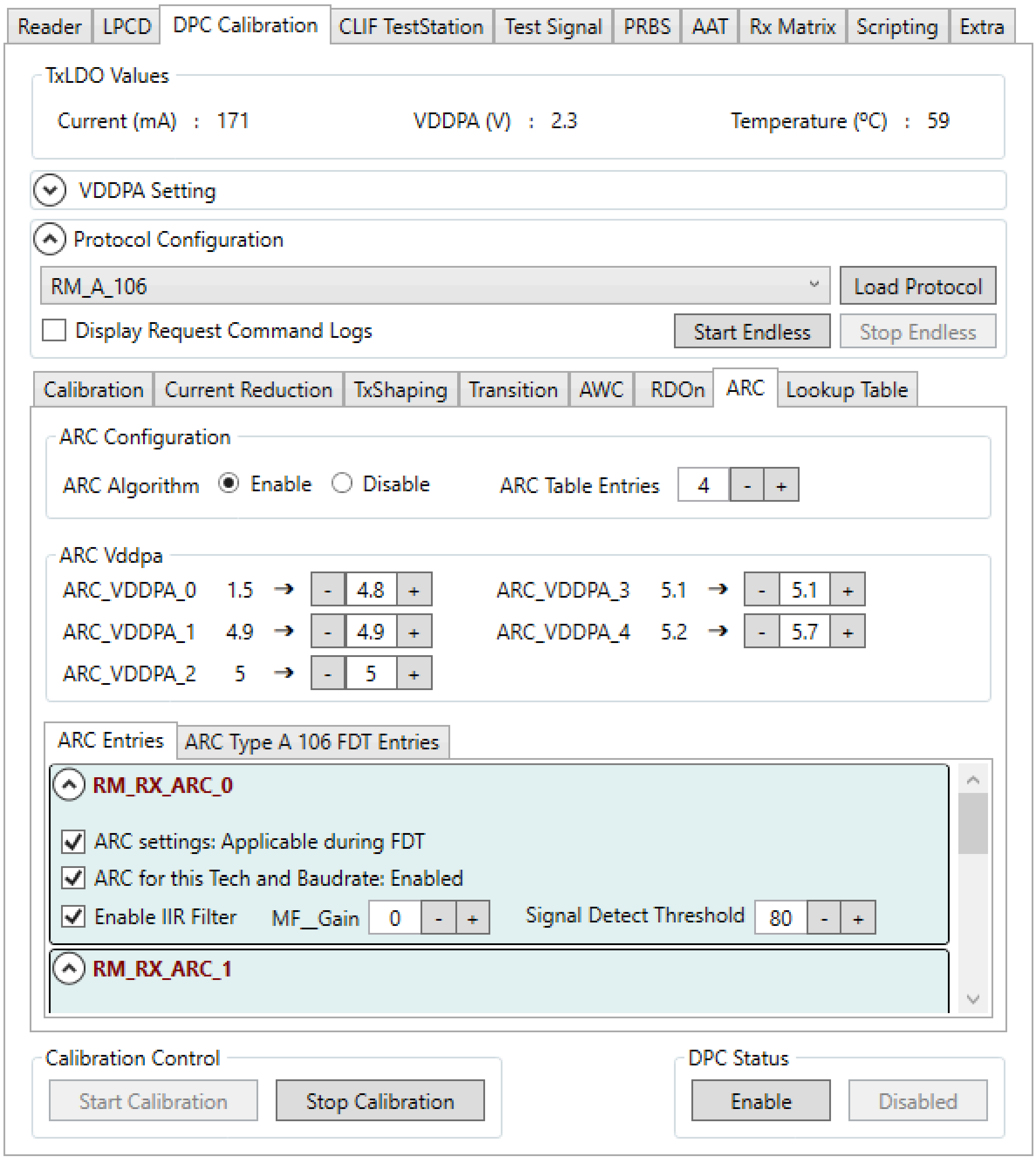



**Figure 92. ARC**

## 4.9 Test bus

Test bus feature allows the user to route any required signal through any of the output pins and further analyze the signal for any correction.



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.9.1 CLRC663

The NFC cockpit allows to use the CLRC663 internal test bus, to route analog test signals to the AUX1/AUX2 and digital test signals to SIGOUT.

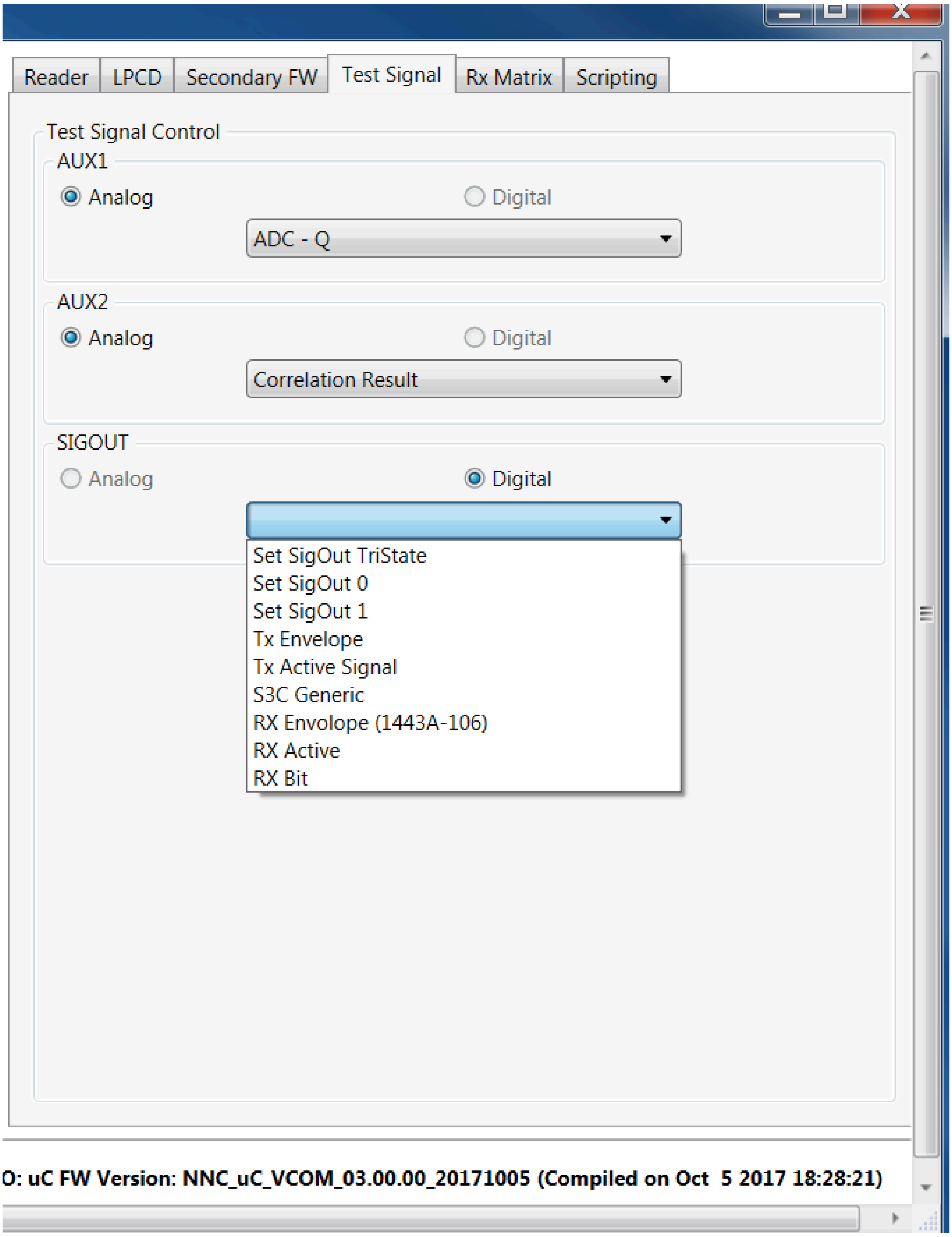



Figure 93. Test bus CLRC663



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.9.2 PN7462

The NFC cockpit allows to use the PN7462AU internal test bus, to route digital and analog test signals to the given test pins (GPIO1/2 and GPIO4/5), as shown in. All details on the test signals can be found in [3].

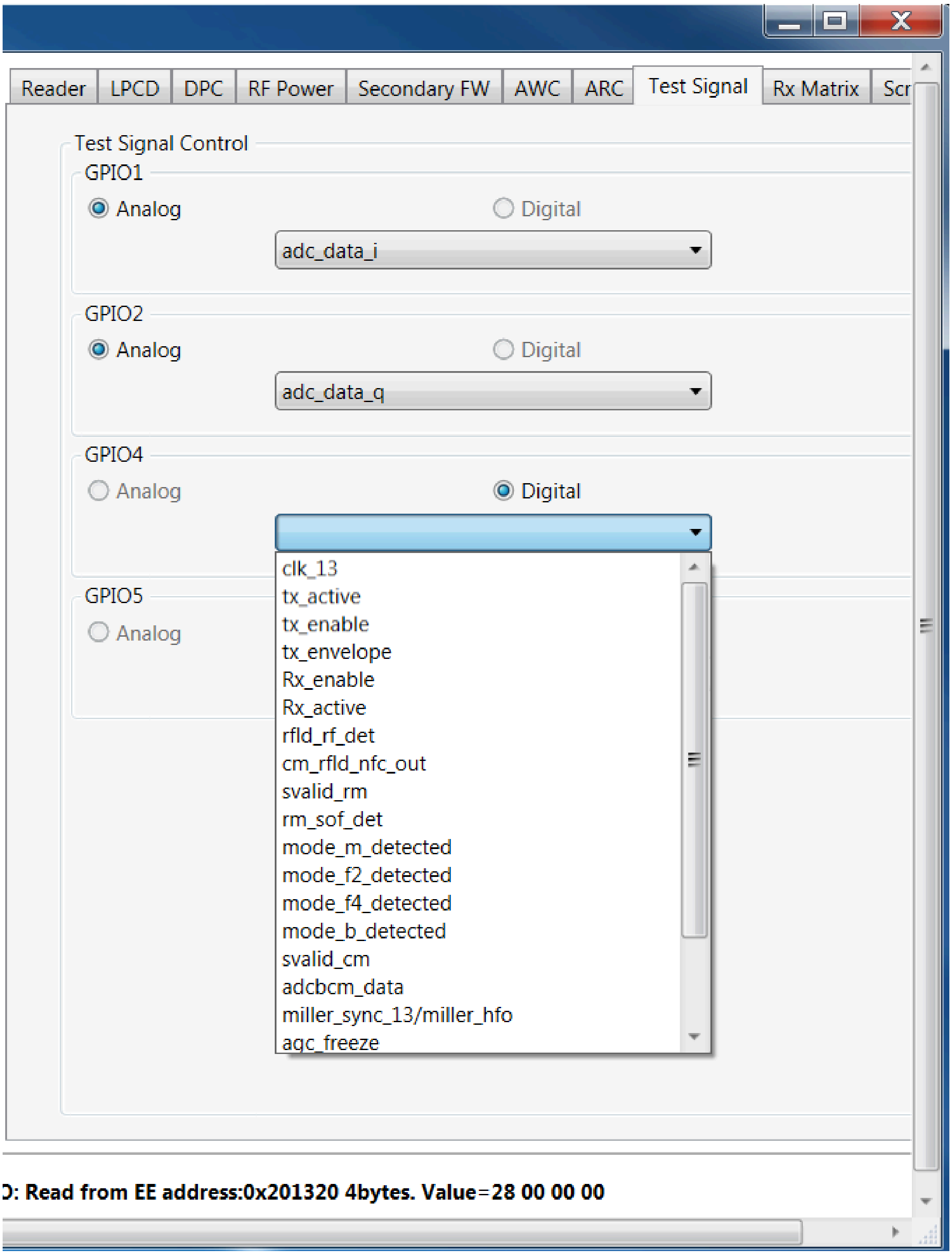



Figure 94. Test bus PN7462AU



© 2023 NXP B.V. All rights reserved.

### 4.9.3 PN5180

The NFC cockpit allows to use the PN5180 internal test bus, to route analog test signals to the AUX1/AUX2/GPIO1/IRQ and digital test signals to IRQ/GPIO1.

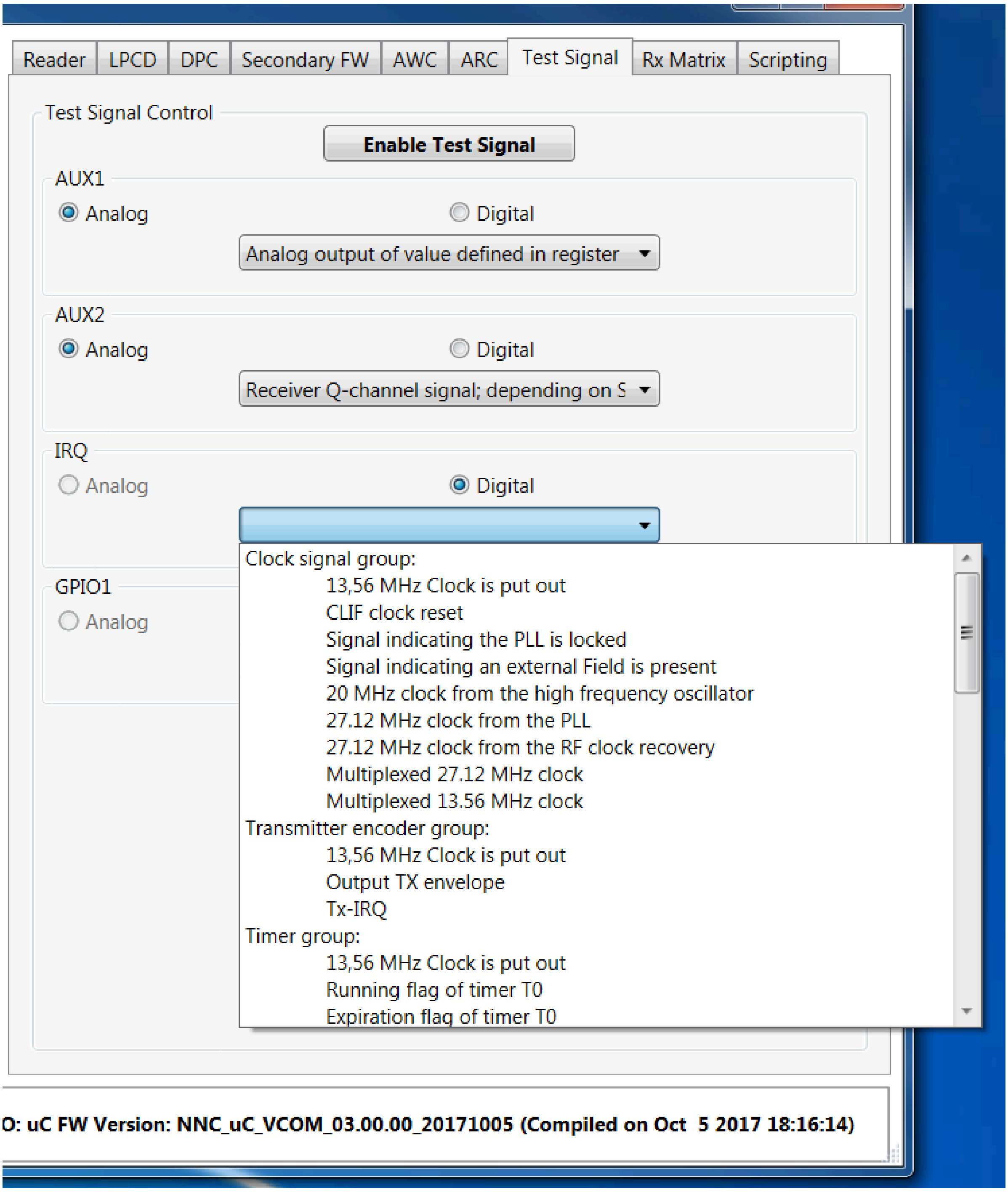



**Figure 95. Test bus PN5180**



© 2023 NXP B.V. All rights reserved.

### 4.9.4 PN5190

The NFC cockpit allows to use the PN5190 internal test bus, to route analog test signals to the AUX1 and AUX2 and digital test signals to GPIO0, GPIO1, GPIO2, and GPIO3.

The analog configuration can be performed in two ways,

**RAW Mode** Used to configure a specific bit of a signal. The output is routed on the AUX pins based on the ShiftIndex and Mask values.

**Combined Mode** Used to configure two sets of signals on either AUX1 or AUX2

***Note:*** *While selecting the Digital Signals, the Bits should be selected and not the signal. If the signal is selected, a warning message is displayed in the log.*

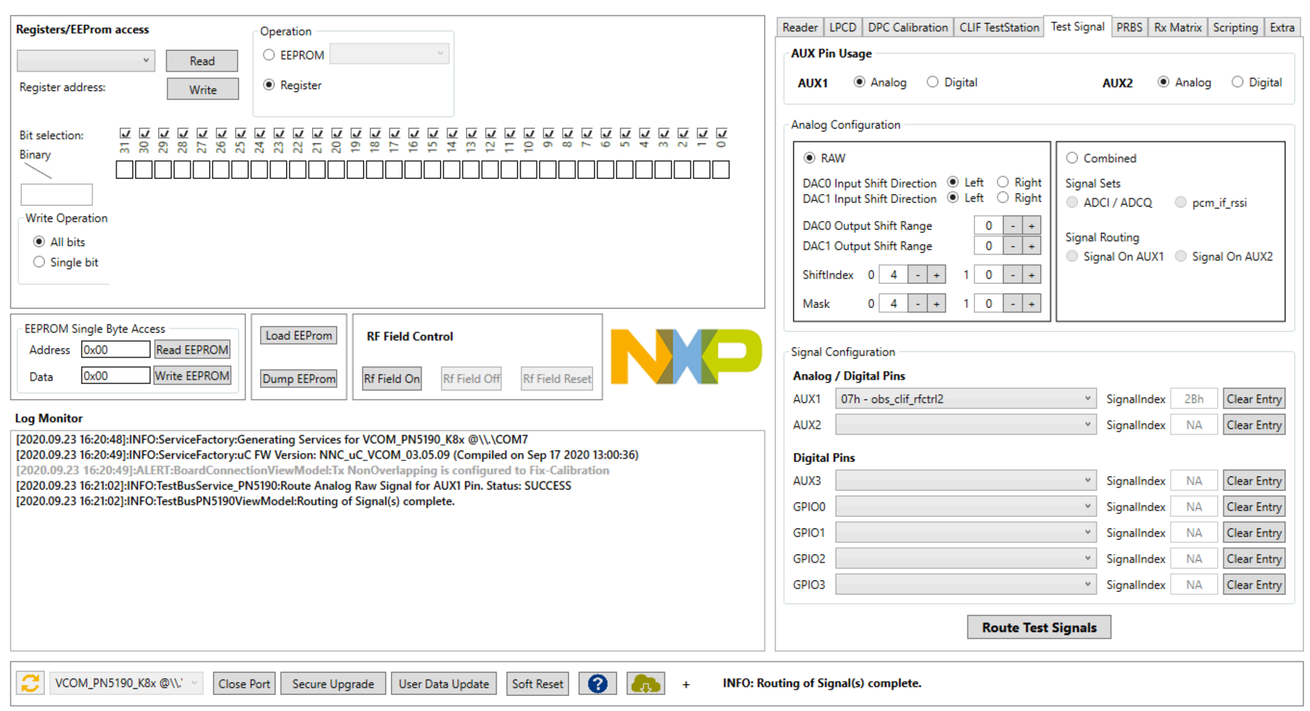



**Figure 96. Analog (RAW Mode) Signal Routing**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### Dual test bus

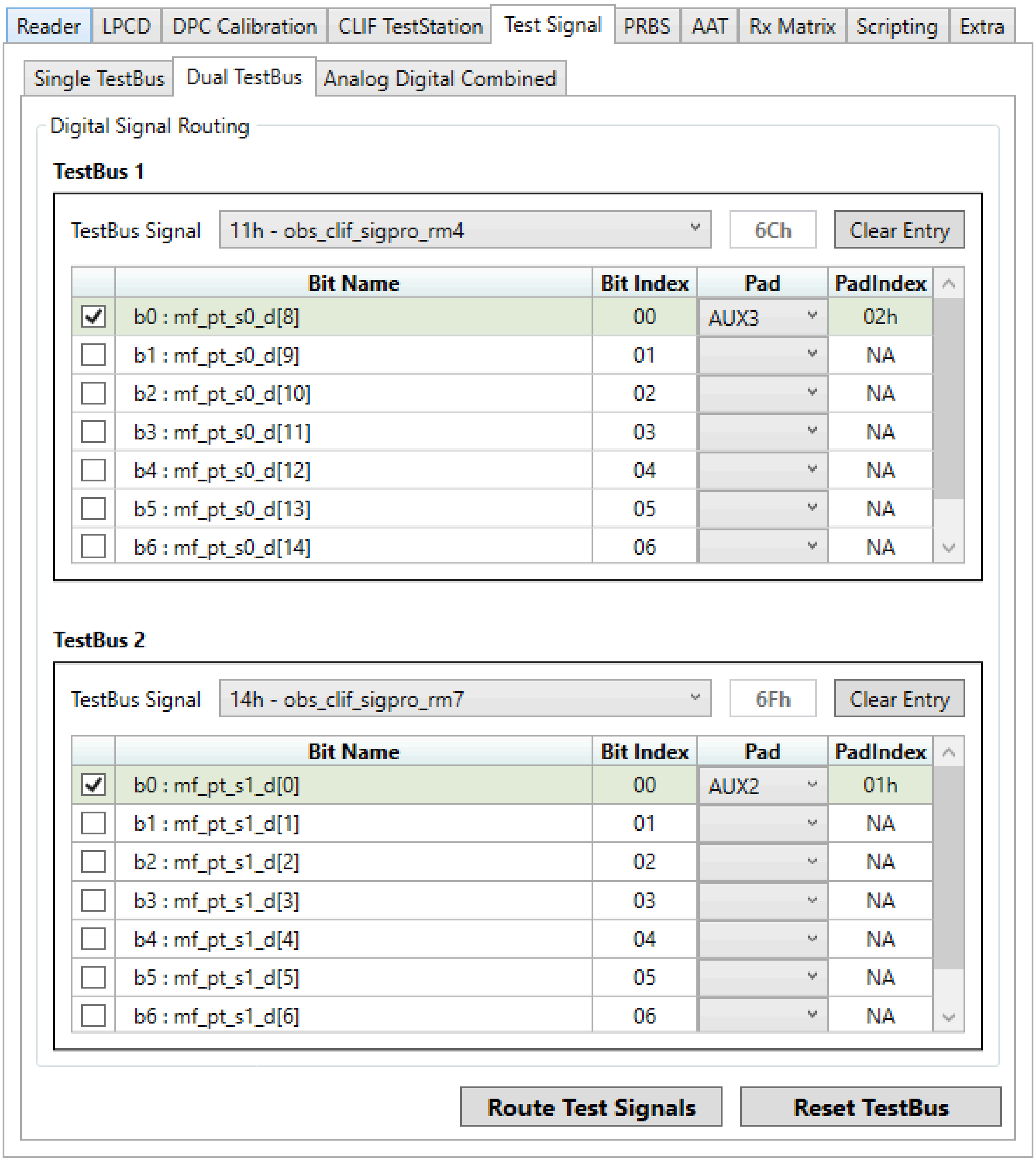



Figure 97. Digital (Warning, if Bit is not selected and Signal is selected)



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### Analog Digital Combined

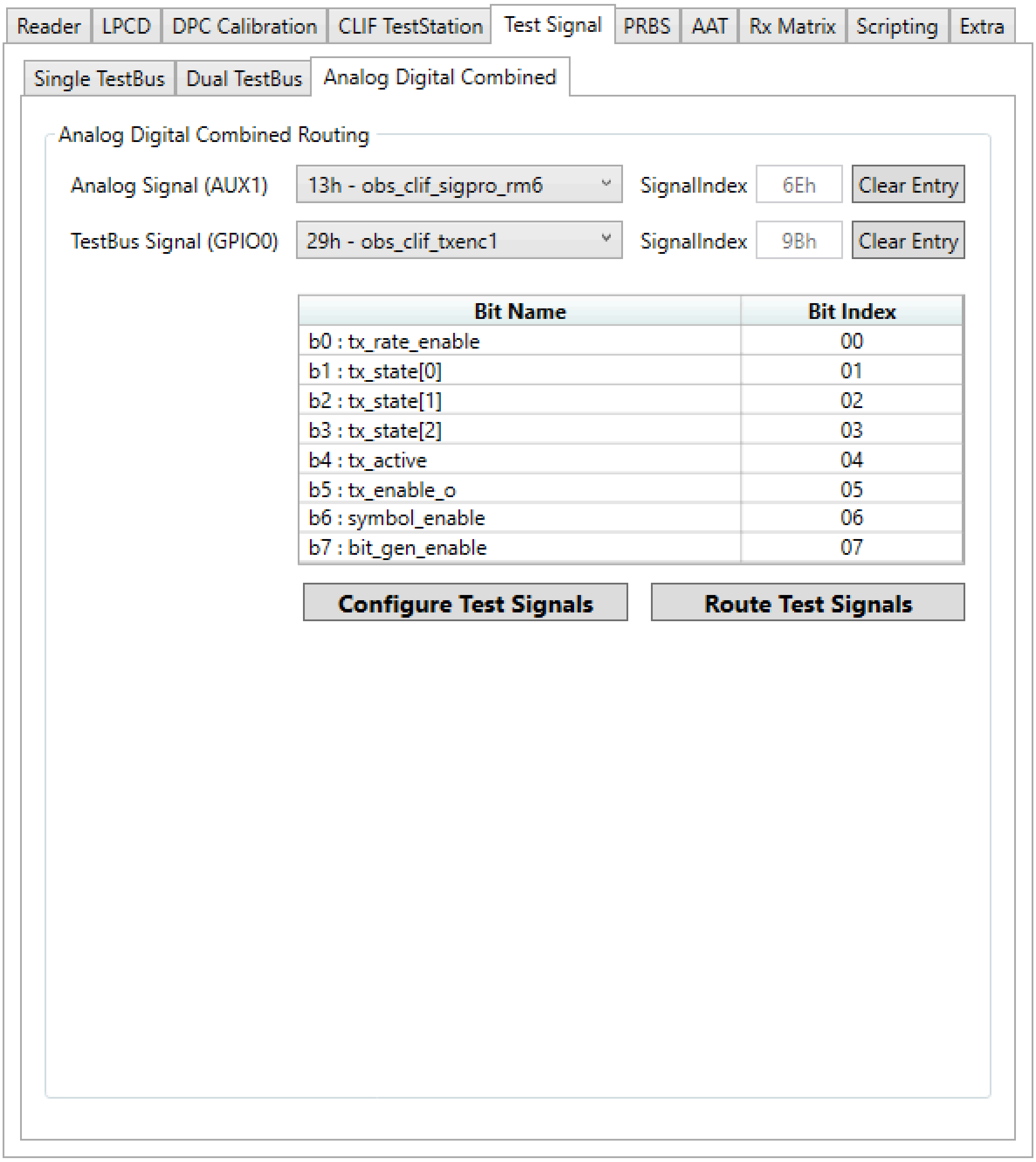



Figure 98. Digital (Warning, if Bit is not selected and Signal is selected)



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

### 4.9.5 PN76XX

The NFC cockpit allows to use the PN7640 / PN7642 internal test bus, to route analog test signals to the AUX1 and AUX2 and digital test signals to GPIO0, GPIO1, GPIO2, and GPIO3.

The analog configuration can be performed in two ways,

| | |
|---|---|
| **RAW Mode** | Used to configure a specific bit of a signal. The output is routed on the AUX pins based on the ShiftIndex and Mask values. |
| **Combined Mode** | Used to configure two sets of signals on either AUX1 or AUX2 |

***Note:*** *While selecting the Digital Signals, the Bits should be selected and not the signal. If the signal is selected, a warning message is displayed in the log.*

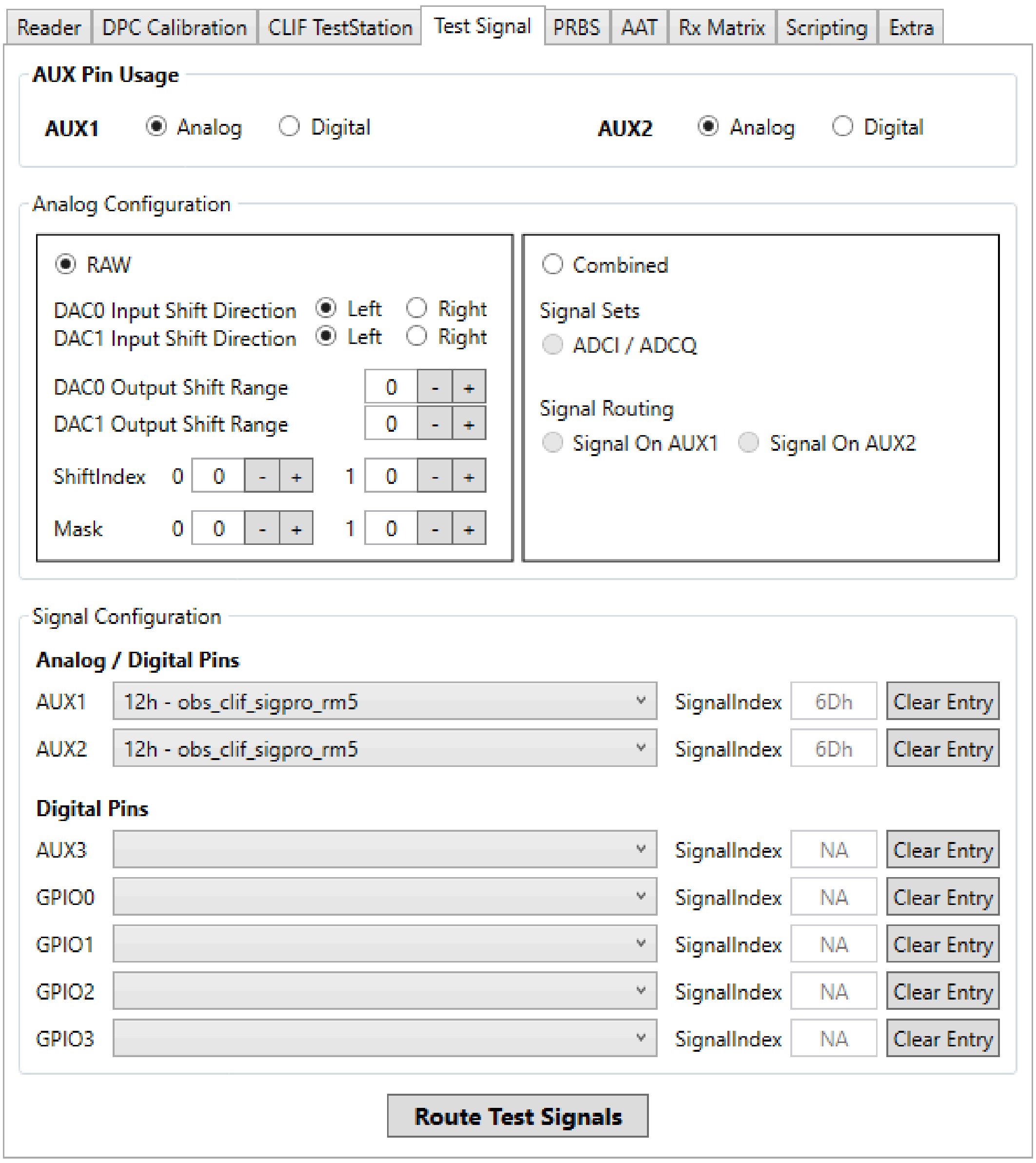



**Figure 99. Analog (RAW Mode) Signal Routing**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

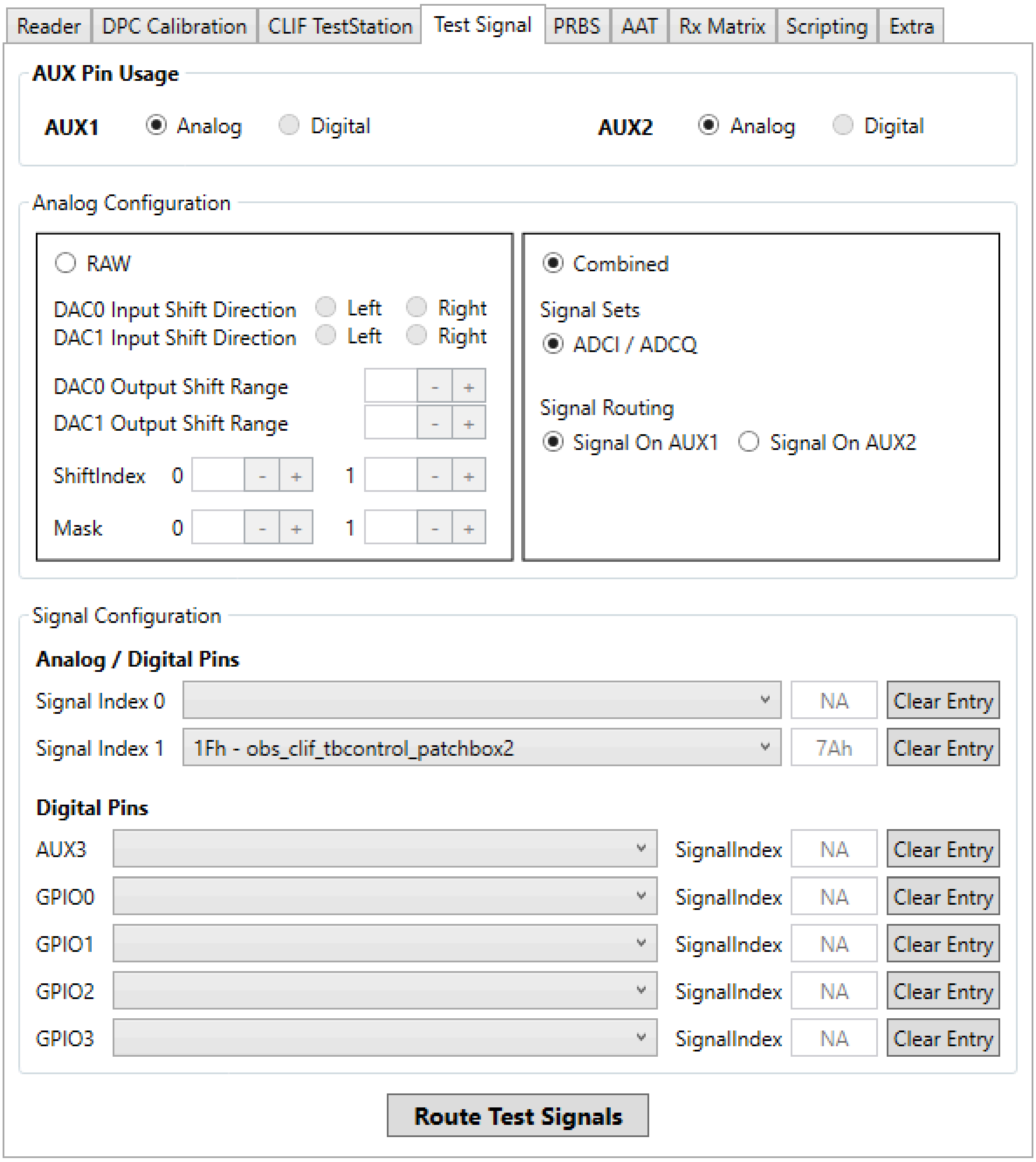



**Figure 100. Analog (Combined Mode) Signal Routing**

## 4.10 Rx Matrix

**RxMatrix** is a feature provided by NFC Cockpit to find the right combination of tx and rx register values for a successful transmission and reception. User is given an option to load an xml with the input as registers to be modified and other configuration parameters to get the output as an average of successful transreceive.


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

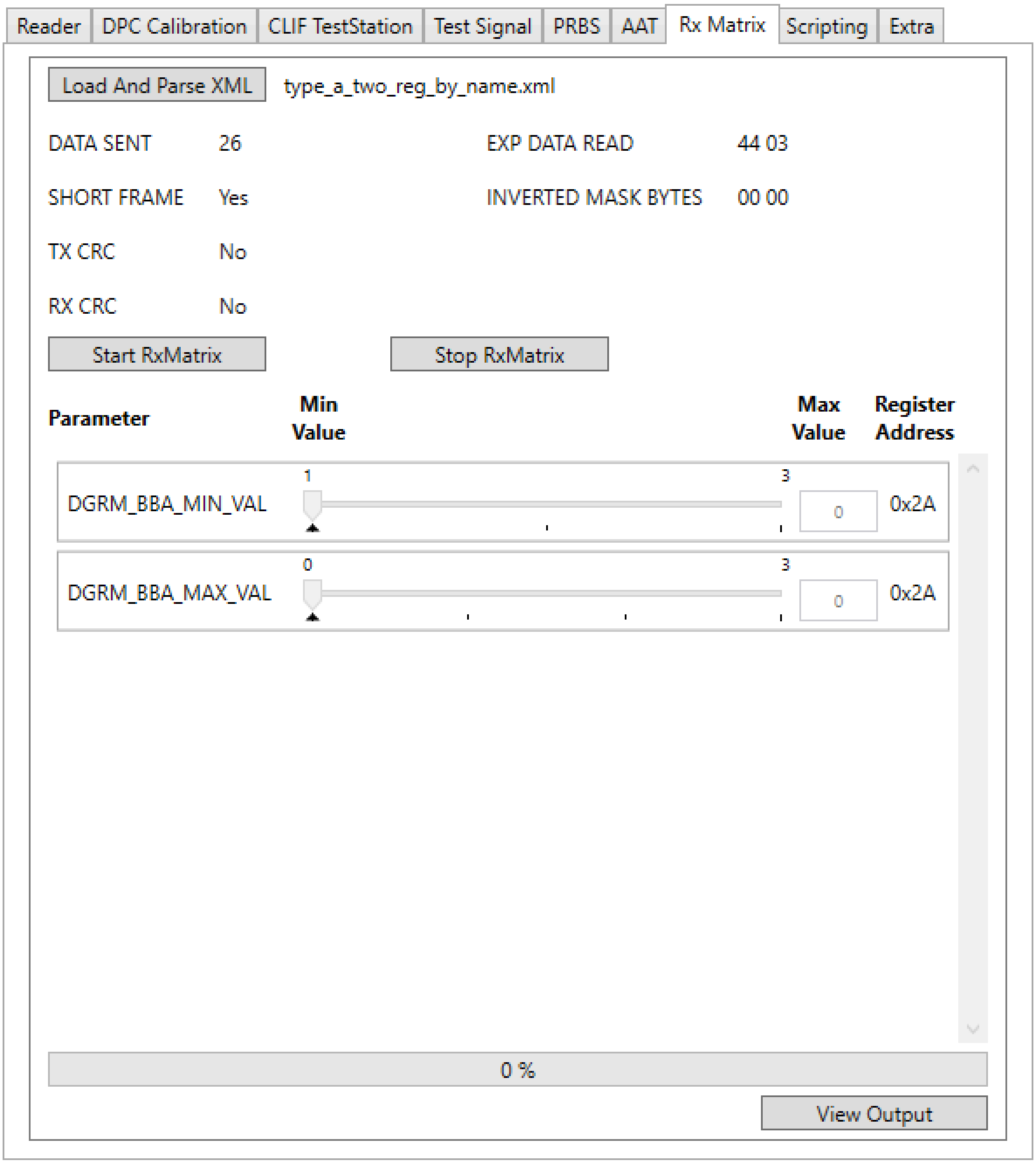



Figure 101. Rx Matrix

## 4.10.1 XML Tags and Attributes

Below are the list of tags and attributes that are supported in RxMatrix feature.

### 4.10.1.1 Root element

Below are the list of tags and attributes that are supported in RxMatrix feature.



© 2023 NXP B.V. All rights reserved.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **Test** | | RxMatrix configuration file root element name. All other elements should be inside this main root element. |
| | numberMaxOfPasses | Number of times to check for successful behavior |
| | skipAfterFailures | Check of the failures and end the RxMatrix execution if failures reach the expected count. |
| | delayMS | Delay between each iteration of execution. Should be in milliseconds. |
| | fieldReset | Should the field be reset for each iteration. Supported values are YES or NO. |
| | protocolType | The supported reader mode protocol type. |
| | **Element value is not supported.** | |

### 4.10.1.2 Child Element (SendData)

Below are the list of tags and attributes that are supported in RxMatrix feature.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **SendData** | | RxMatrix configuration file first child element. Element for configuring the data to be sent. |
| | shortFrame | Is the frame short frame. Supported values are YES or NO. |
| | rxCRC | Is there a CRC on the response data. Supported values are YES or NO. |
| | txCRC | Should there be a CRC after the data to be sent. Supported values are YES or NO |
| | timeOutInUs | Timeout for the data to send. Should be in microseconds. |
| | **Element value is supported. The value should be the data to be sent.** | |

### 4.10.1.3 Child Element (ReadData)

Below are the list of tags and attributes that are supported in RxMatrix feature.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **ReadData** | | RxMatrix configuration file next child element. Element for configuring the data to receive. |
| | invertedMaskBytes | |
| | **Element value is supported. The value that will be received.** | |

### 4.10.1.4 Child Element (Frequency)

Below are the list of tags and attributes that are supported in RxMatrix feature.



© 2023 NXP B.V. All rights reserved.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **Frequency** | | RxMatrix configuration file next child element. Element for configuring the AWG frequency. |
| | Minimum | The start value. |
| | Maximum | The end value. |
| | StepSize | Frequency levels between Minimum and Maximum. Ex. If Minimum = 1000 and Maximum = 2000 and StepSize = 200, AWG generates 6 set of frequencies between 1000 - 2000 which is nothing but 1000, 1200, 1400, 1600, 1800 and 2000 |
| | Unit | The unit to be used while generating the frequencies. Supported units are Hx, kHz, MHz, and GHz. |
| | **Element value is not supported.** | |

#### 4.10.1.5 Child Element (Voltage)

Below are the list of tags and attributes that are supported in RxMatrix feature.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **Voltage** | | RxMatrix configuration file next child element. Element for configuring the AWG frequency. |
| | minValueInmV | The start value. |
| | maxValueInmV | The end value. |
| | stepSizeInmV | Voltage levels between Minimum and Maximum. Ex. If Minimum = 1000 and Maximum = 2000 and StepSize = 200, AWG generates 6 set of Voltage Levels between 1000 - 2000 which is nothing but 1000, 1200, 1400, 1600, 1800 and 2000 |
| | **Element value is not supported.** | |

#### 4.10.1.6 Child Element (Parameter)

Below are the list of attributes for Element Parameter. Multiple Registers and Fields can be configured using multiple Parameter elements. Refer to Section 4.10.3.

##### 4.10.1.6.1 Parameter - By Name

List of attributes to perform RxMatrix using Register information as parameters.

| Element Name | Attribute Name | Description |
| --- | --- | --- |
| **Parameter** | | RxMatrix configuration file next child element. Element for configuring the Registers. |
| | register | The name of the register for which the values need to be updated. |



© 2023 NXP B.V. All rights reserved.

| Element Name | Attribute Name | Description |
|---|---|---|
| | field | The fields of the above register to which the values will be updated. |
| | minValue | The Start Value. |
| | maxValue | The End Value. |
| | **Element value is not supported.** | |

#### 4.10.1.6.2 Parameter - By Range

List of attributes to perform RxMatrix using Register bit length, Bit Position and value ranges as parameters.

| Element Name | Attribute Name | Description |
|---|---|---|
| **Parameter** | | RxMatrix configuration file next child element. Element for configuring the Registers. |
| | name | User name for the parameter. Can be of any name. |
| | minValue | The Start Value. Values should be based on the bit length information. |
| | maxValue | The End Value. Values should be based on the bit length information. |
| | registerAddress | The address of the register to update. |
| | bitPosition | The bit position in the above register to update. |
| | bitLength | The number of bits to be used for updating the above mentioned bit position. |
| | **Element value is not supported.** | |

#### 4.10.1.6.3 Parameter - By Sequence

List of attributes to perform RxMatrix using Register bit length and direct values.

| Element Name | Attribute Name | Description |
|---|---|---|
| **Parameter** | | RxMatrix configuration file next child element. Element for configuring the Registers. |
| | name | User name for the parameter. Can be of any name. |
| | registerAddress | The address of the register to update. |
| | bitPosition | The bit position in the above register to update. |
| | bitLength | The number of bits to be used for updating the above mentioned bit position. |
| | values | The values to update. Values should be based on the bit length information. |
| | **Element value is not supported.** | |

### 4.10.2 Running

1. Press `Load And Parse XML` in Figure 101



© 2023 NXP B.V. All rights reserved.

2. Select a reference XML File from the dialog box and load
3. Press `Start RxMatrix`

### 4.10.3 RxMatrix: Input Format

RxMatrix feature process and uses configuration in an XML Format.

The configuration files can be found at `C:\nxp\NxpNfcCockpit_v<Version>\cfg\RxMatrix`

#### 4.10.3.1 CLRC663 Reference Script

Here is a reference RxMatrix script for CLRC663 (`type_a_two_reg_by_name.xml` provided with NxpNfcCockpit).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Test SYSTEM "NNC_RxMatrix_Rc663.dtd">
<!-- This is an example of a TypeA test script for Rc663 where we acess
 regsiters by names -->
<Test numberMaxOfPasses="10" skipAfterFailures="4" delayMS="0" fieldReset="YES"
 protocolType="RM_A_106" >
    <SendData shortFrame="YES" rxCRC="NO" txCRC="NO" timeOutInMs="145"
 sli15693FastInv="NO"> 0x26 </SendData>
    <ReadData invertedMaskBytes="0x00, 0x00"> 0x44, 0x03 </ReadData>

    <!--
    Frequency levels, decimal values.
    Supported Units: Hz, KHz, MHz and GHz
    -->
    <!-- <Frequency Minimum="14900" Maximum="15100" StepSize="100" Unit="Hz"/>
 -->

    <Voltage levels can be float or decimal values -->
    <VoltageLevel minValueInmV="100" maxValueInmV="2000" stepSizeInmV="500"/>


    <Parameter register="RXANA_REG" field="RCV_GAIN" minValue="0x01"
 maxValue="0x03" />
    <Parameter register="RXANA_REG" field="RCV_HPCF" minValue="0x00"
 maxValue="0x03" />
</Test>
```

#### 4.10.3.2 PN7462 Reference Script

Here is a reference RxMatrix script for PN7462 (`type_a_two_reg_by_name.xml` provided with NxpNfcCockpit).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Test SYSTEM "NNC_RxMatrix_Pn7462AU.dtd">
<!-- This is an example of a TypeA test script for Pn7462AU where we acess
 regsiters by names -->
<Test numberMaxOfPasses="10" skipAfterFailures="4" delayMS="0" fieldReset="YES"
 protocolType="RM_A_106" >
    <SendData shortFrame="YES" rxCRC="NO" txCRC="NO" timeOutInMs="145"
 sli15693FastInv="NO"> 0x26 </SendData>
    <ReadData invertedMaskBytes="0x00, 0x00">0x44, 0x03 </ReadData>

    <!--
```



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

```
    Frequency levels, decimal values.
    Supported Units: Hz, KHz, MHz and GHz
    -->
    <!-- <Frequency Minimum="14900" Maximum="15100" StepSize="100" Unit="Hz"/>
 -->

    <Voltage levels can be float or decimal values -->
    <VoltageLevel minValueInmV="100" maxValueInmV="2000" stepSizeInmV="500"/>

    <Parameter register="CLIF_ANA_RX_REG" field="RX_GAIN" minValue="0x01"
 maxValue="0x03" />
    <Parameter register="CLIF_ANA_RX_REG" field="RX_HPCF" minValue="0x00"
 maxValue="0x03" />
</Test>
```

#### 4.10.3.3 PN5180 Reference Script

Here is a reference RxMatrix script for PN5180 (`type_a_two_reg_by_name.xml` provided with NxpNfcCockpit).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Test SYSTEM "NNC_RxMatrix_Pn5180.dtd">
<!-- This is an example of a TypeA test script for PN5180 where we acess
 regsiters by names -->
<Test numberMaxOfPasses="10" skipAfterFailures="4" delayMS="0" fieldReset="YES"
 protocolType="RM_A_106" >
    <SendData shortFrame="YES" rxCRC="NO" txCRC="NO" timeOutInMs="145"
 sli15693FastInv="NO"> 0x26 </SendData>
    <ReadData invertedMaskBytes="0x00, 0x00"> 0x44, 0x03 </ReadData>

    <!--
    Frequency levels, decimal values.
    Supported Units: Hz, KHz, MHz and GHz
    -->
    <!-- <Frequency Minimum="14900" Maximum="15100" StepSize="100" Unit="Hz"/>
 -->

    <Voltage levels can be float or decimal values -->
    <VoltageLevel minValueInmV="100" maxValueInmV="2000" stepSizeInmV="500"/>

    <Parameter register="RF_CONTROL_RX" field="RX_GAIN" minValue="0x01"
 maxValue="0x03" />
    <Parameter register="RF_CONTROL_RX" field="RX_HPCF" minValue="0x00"
 maxValue="0x03" />
</Test>
```

#### 4.10.3.4 PN5190 Reference Script

Here is a reference RxMatrix script for PN5190 (`type_a_two_reg_by_name.xml` provided with NxpNfcCockpit).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Test SYSTEM "NNC_RxMatrix_Pn5190.dtd">
<!-- This is an example of a TypeA test script for PN5190 where we acess
 regsiters by names -->
<Test numberMaxOfPasses="10" skipAfterFailures="4" delayMS="0" fieldReset="YES"
 protocolType="RM_A_106" >
```


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

```
    <SendData shortFrame="YES" rxCRC="NO" txCRC="NO" timeOutInUs="145000"> 0x26
 </SendData>
    <ReadData invertedMaskBytes="0x00, 0x00"> 0x44, 0x03 </ReadData>

    <!--
    Frequency levels, decimal values.
    Supported Units: Hz, KHz, MHz and GHz
    -->
    <!-- <Frequency Minimum="14900" Maximum="15100" StepSize="100" Unit="Hz"/>
 -->

    <Voltage levels can be float or decimal values -->
    <VoltageLevel minValueInmV="100" maxValueInmV="2000" stepSizeInmV="500"/>

    <Parameter register="CLIF_ANA_RX_CTRL" field="RX_CLKGEN_PH_CTRL"
 minValue="0x01" maxValue="0x03" />
    <Parameter register="CLIF_ANA_RX_CTRL" field="RX_ATB_MON_SEL"
 minValue="0x00" maxValue="0x03" />
</Test>
```

#### 4.10.3.5 PN76XX Reference Script

Here is a reference RxMatrix script for PN7640 and PN7642 (`type_a_two_reg_by_name.xml` provided with NxpNfcCockpit).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE Test SYSTEM "NNC_RxMatrix_Pn76XX.dtd">
<!-- This is an example of a TypeA test script for PN76XX (PN7640 and PN76XX)
 where we acess regsiters by names -->
<Test numberMaxOfPasses="10" skipAfterFailures="4" delayMS="0" fieldReset="YES"
 protocolType="RM_A_106" >
    <SendData shortFrame="YES" rxCRC="NO" txCRC="NO" timeOutInUs="145000"> 0x26
 </SendData>
    <ReadData invertedMaskBytes="0x00, 0x00"> 0x44, 0x03 </ReadData>

    <!--
       Frequency levels, decimal values.
       Supported Units: Hz, KHz, MHz and GHz
    -->
    <!-- <Frequency Minimum="14900" Maximum="15100" StepSize="100" Unit="Hz"/>
 -->

    <Voltage levels can be float or decimal values -->
    <VoltageLevel minValueInmV="100" maxValueInmV="2000" stepSizeInmV="500"/>

    <Parameter register="CLIF_DGRM_BBA" field="DGRM_BBA_MIN_VAL" minValue="0x01"
 maxValue="0x03" />
    <Parameter register="CLIF_DGRM_BBA" field="DGRM_BBA_MAX_VAL" minValue="0x00"
 maxValue="0x03" />
</Test>
```


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

## 4.11 CLIF TestStation

The NFC Cockpit allows the configuration of signals and captures its logs. The logs are then populated on another UI in form of signals.

| | |
|---|---|
| **CTS Config** | Saves the selected configuration info to PN5190 or PN7640 / PN7642 IC. |
| **CTS Enable** | Enables the CTS processor inside PN5190 or PN7640 / PN7642 IC. |
| **CTS Retrieve Log** | Retrieves the logs from PN5190 or PN7640 / PN7642 IC, processes it and displays as a wave using external tool. |

The tool also does the following. This functionality does not communicate to PN5190 or PN7640 /PN7642 IC.

| | |
|---|---|
| **Load Config** | To open any existing configuration (.xml), the information from the xml file will be updated to the UI. |
| **Save Config** | Saves the CTS Configuration Output information to text file and the selected configuration to xml file. |
| **Display Signals** | Process and displays the logs from *.txt file and displays the log in form of waveform using external tool. |

***Note: Available only for PN5190 or PN7640 / PN7642***

1. *PN5190 supports Capture Size up to 8K.*
2. *PN7640 and PN7642 support Capture Size up to 2K.*

**Configuration and Single Capture.**

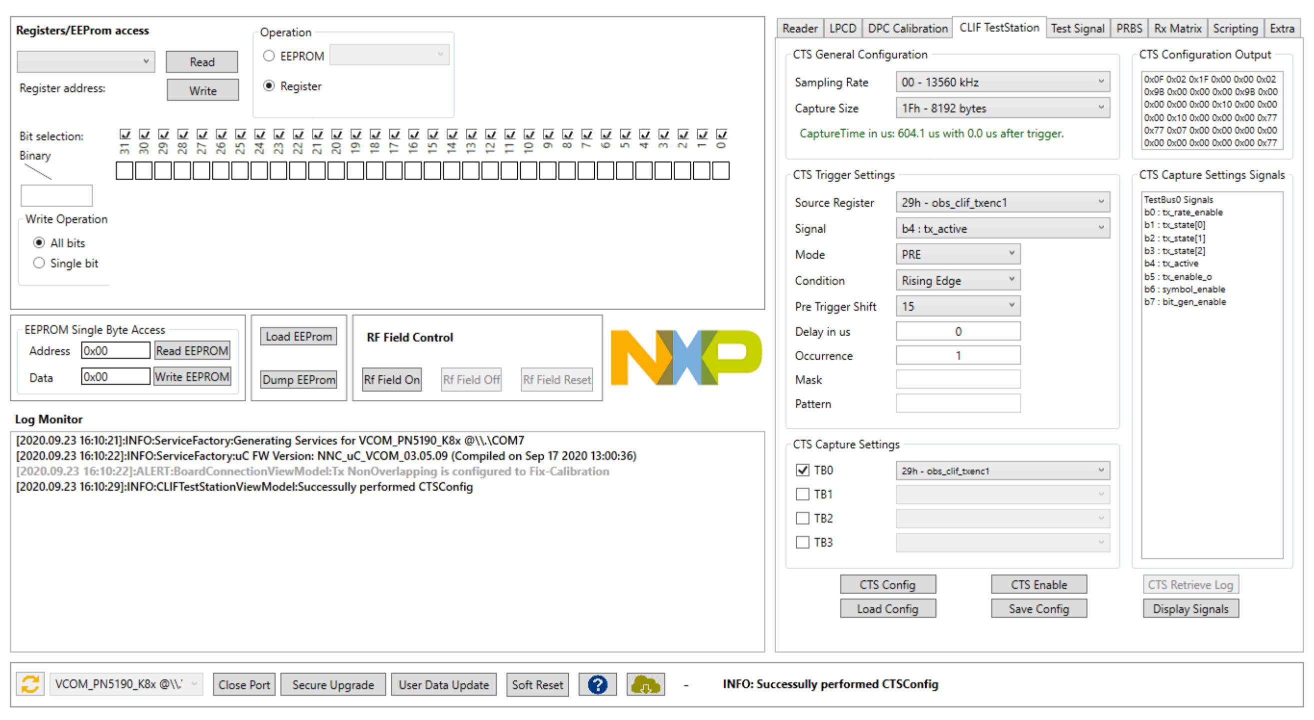


Figure 102. CTS: Configuration



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

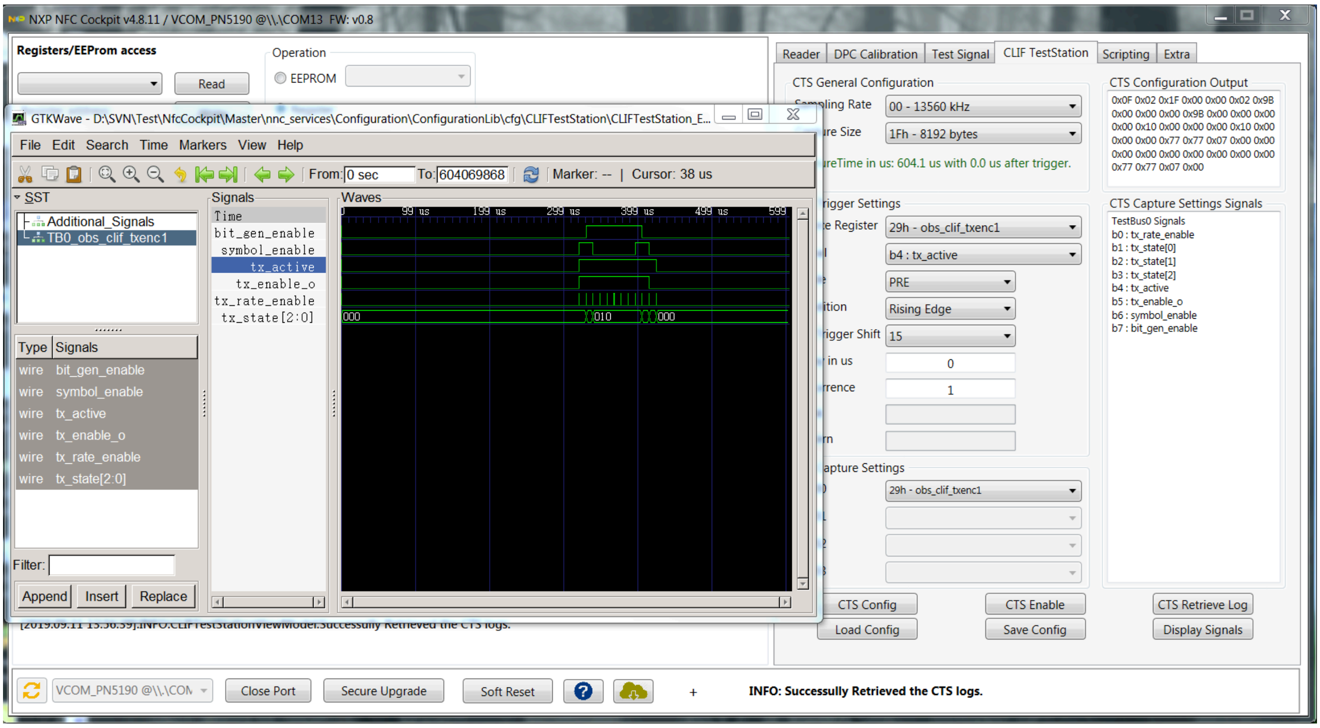


Figure 103. CTS: LogDisplay

### Multi Capture

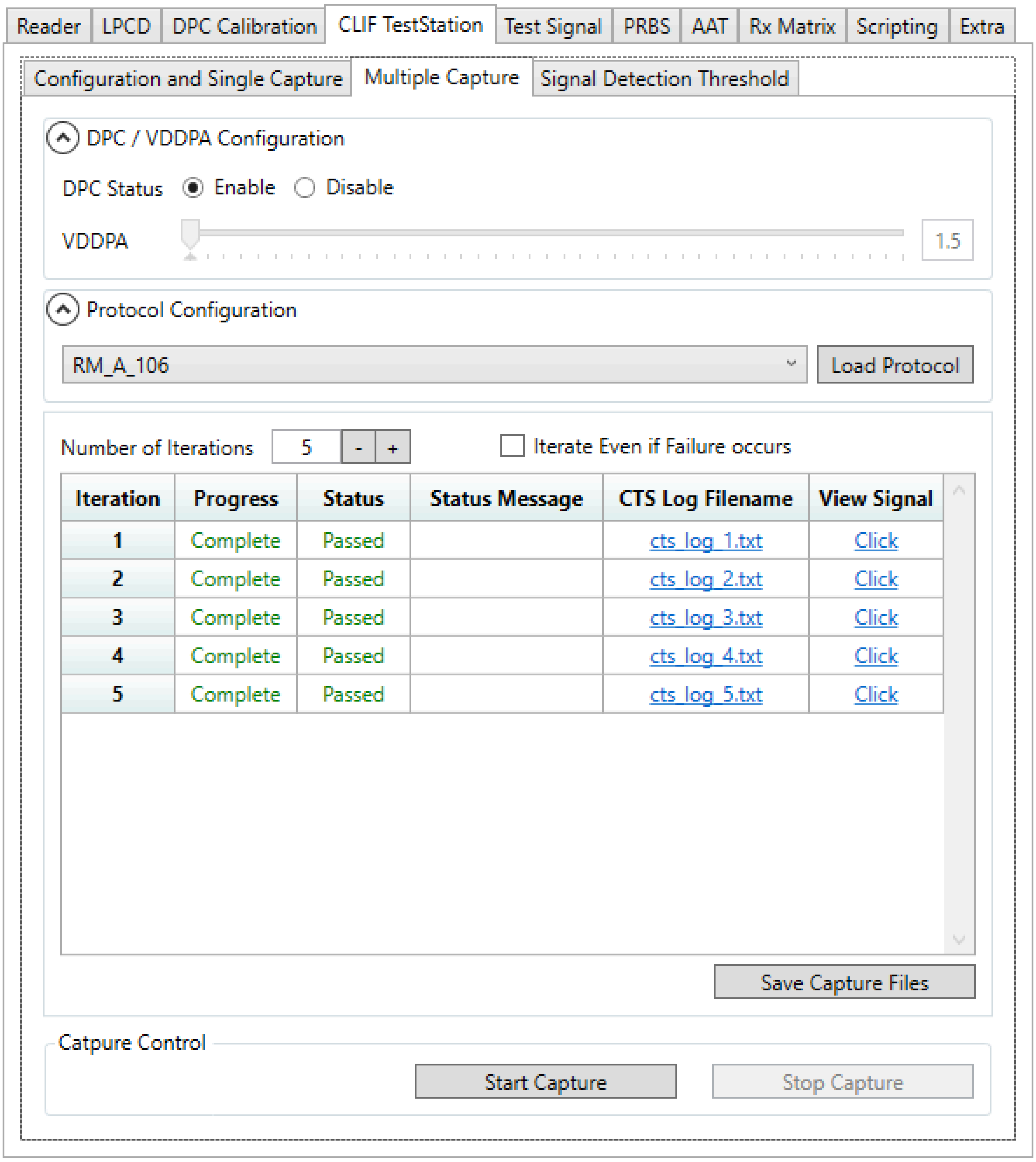



**Figure 104. CTS: MultiCapture**


© 2023 NXP B.V. All rights reserved.

### Signal Detection Threshold

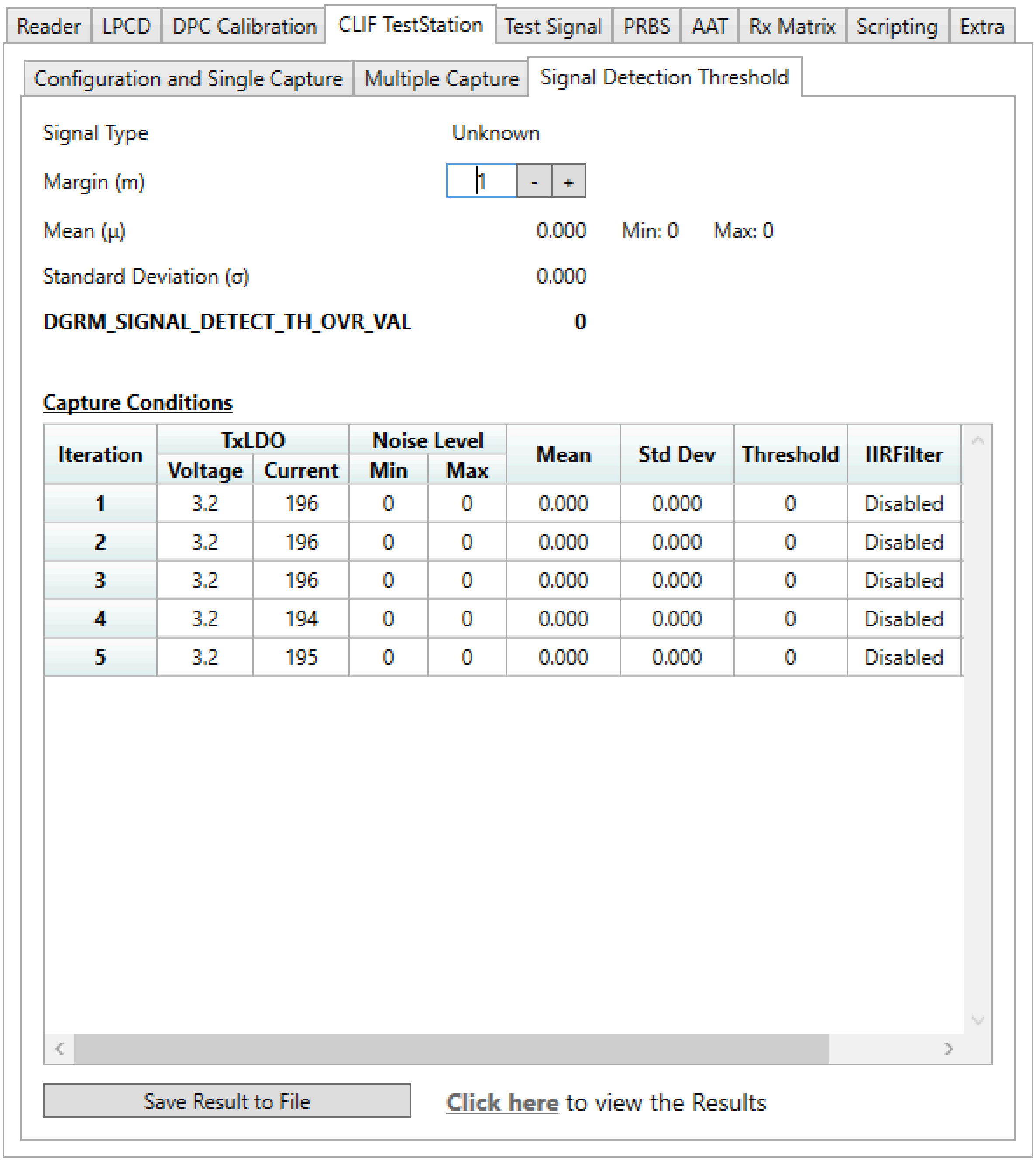



**Figure 105. CTS: Signal DetectionThreshold**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

## 4.12 PRBS

**PRBS: Pseudo** Random Binary Sequence.

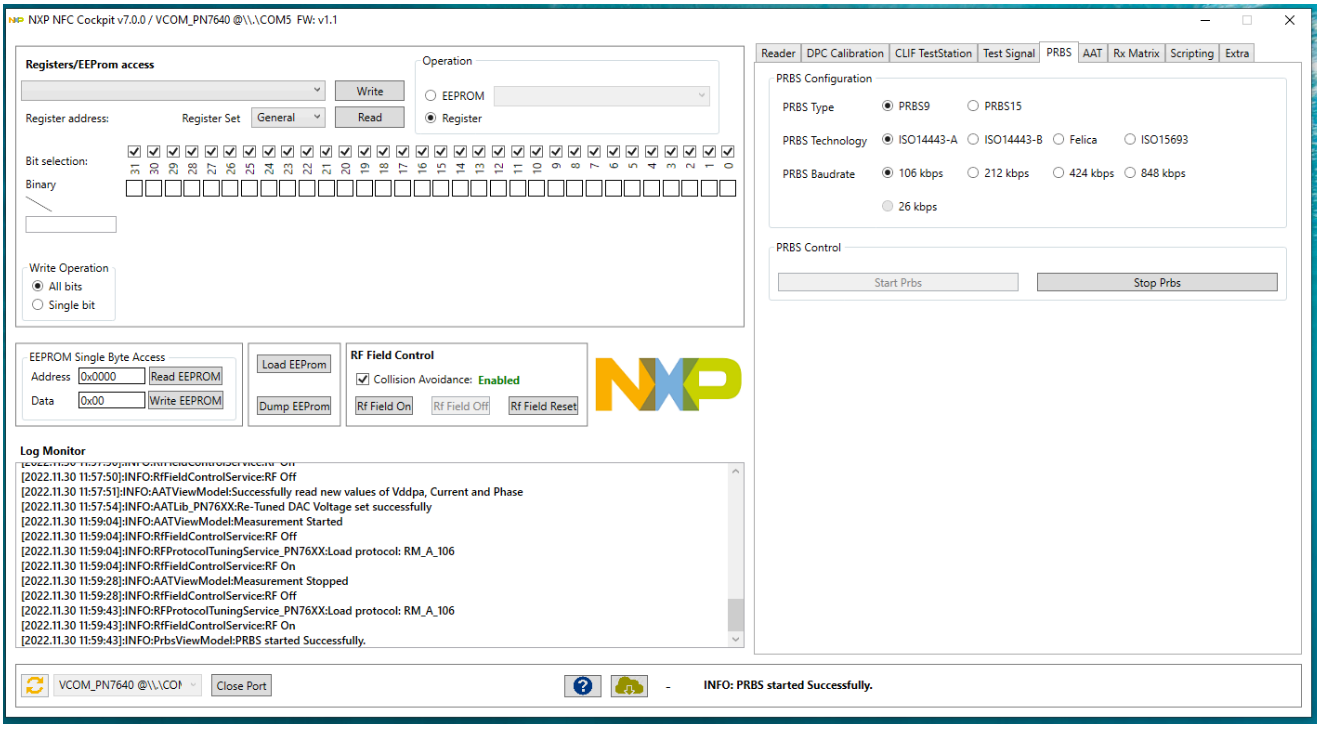



Figure 106. PRBS-Configuration



© 2023 NXP B.V. All rights reserved.

## 4.13 Extra

**Extra: Load** Secondary firmware button once we click on this it loads some of the firmware CLRC663 specific setup firmware (NFC Cockpit firmware) extra (some extra setup Application). These are nothing but the Secondary Application. Once we select the NFC Cockpit firmware it will start to load the Cockpit-specific firmware to LPC. After the warning message we need to do close port and then Open Port.

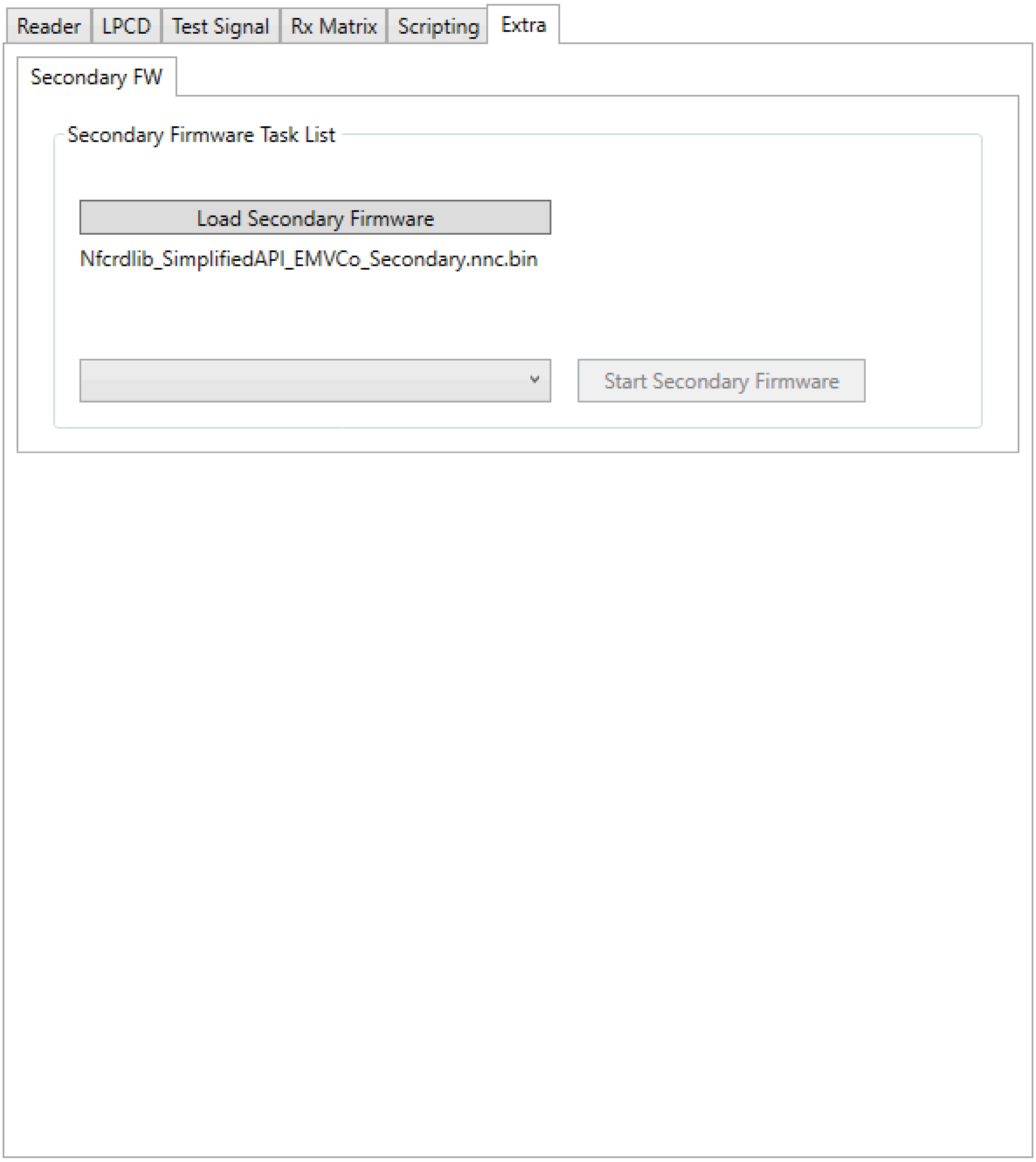



**Figure 107. RC663 Secondary firmware**



© 2023 NXP B.V. All rights reserved.

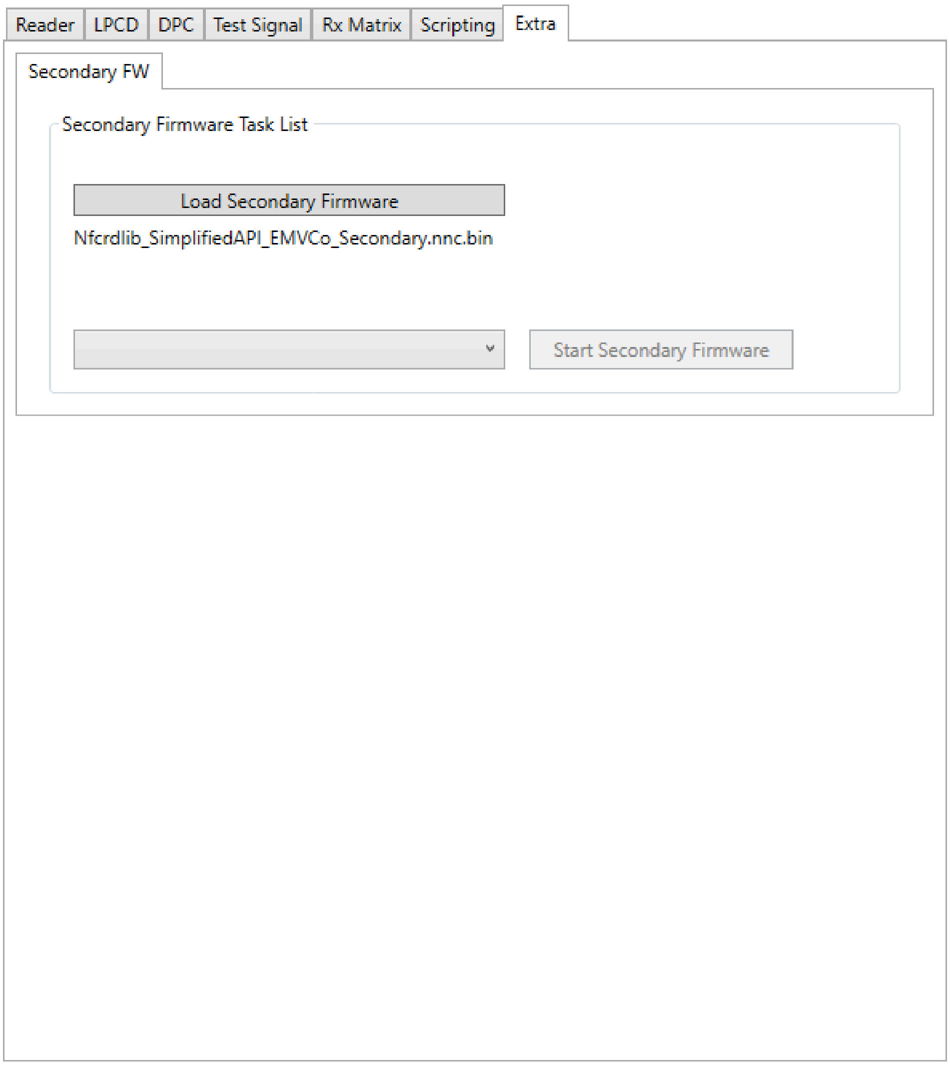



**Figure 108. PN5180 Secondary firmware**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

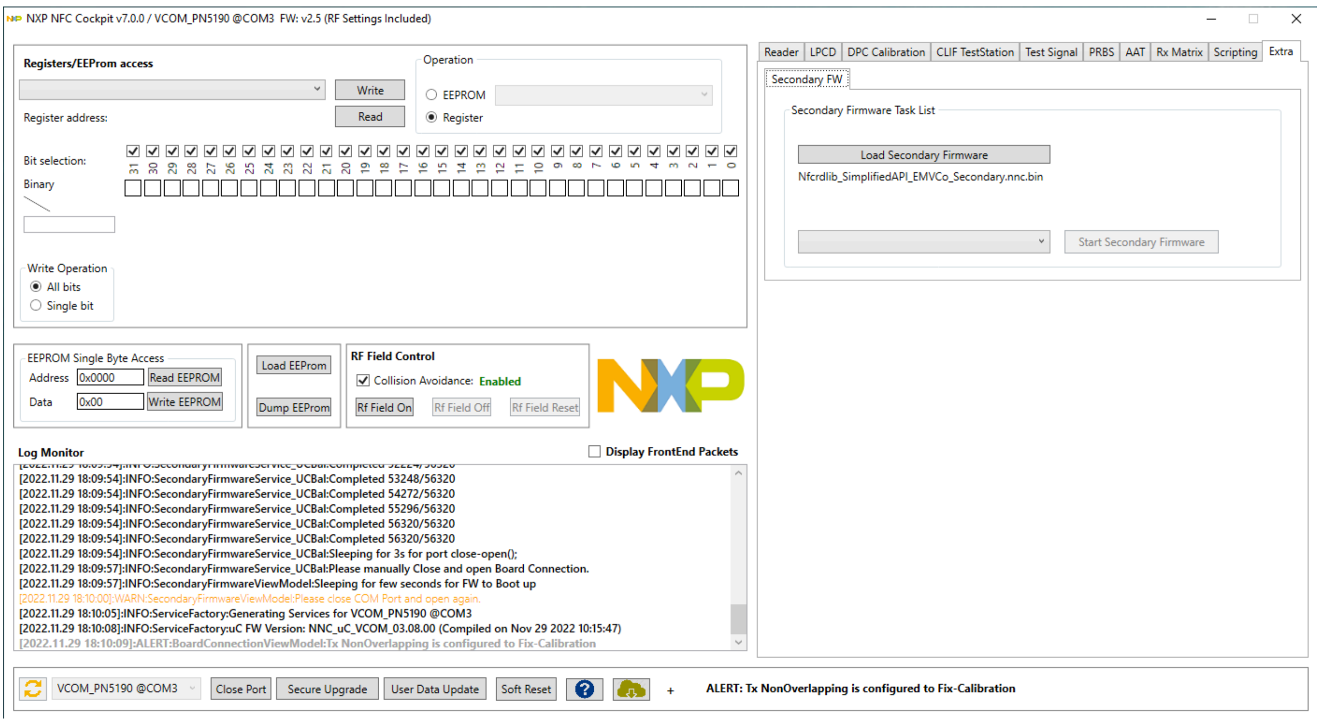


Figure 109. PN5190 Secondary firmware

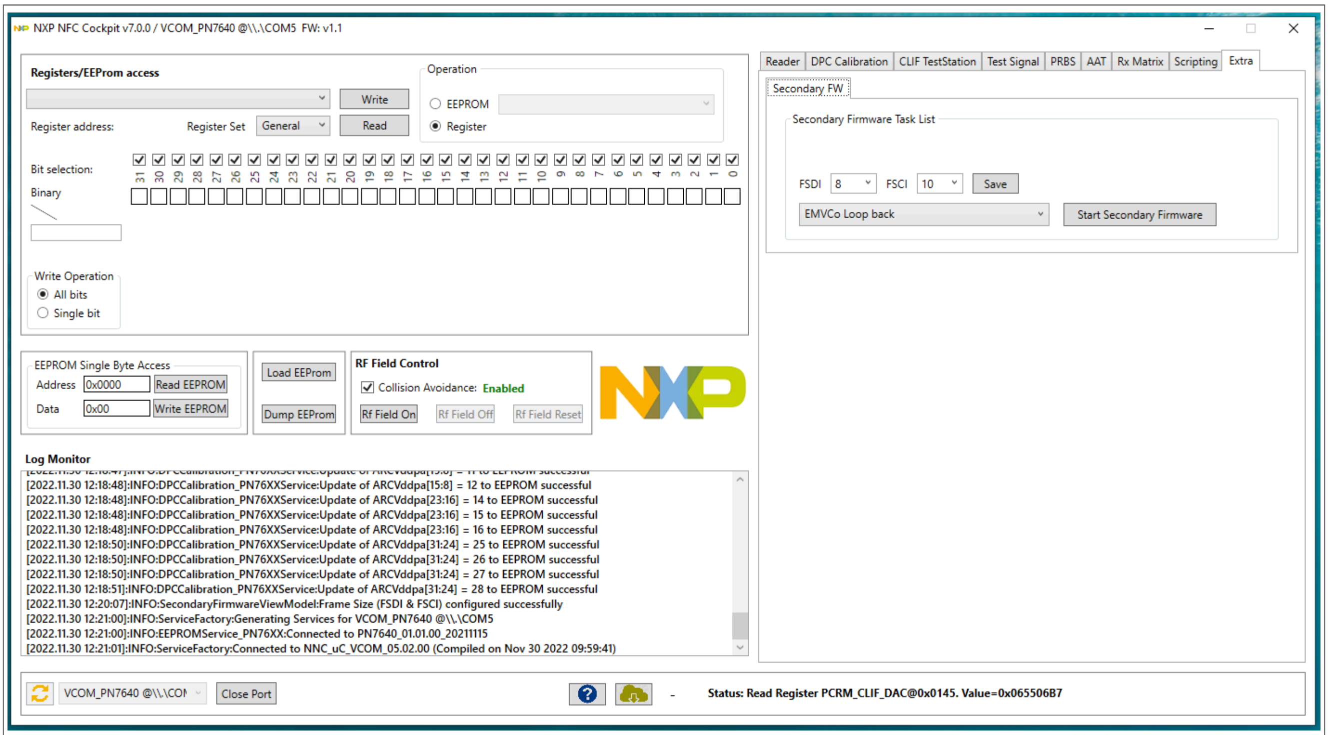


Figure 110. PN7640 Secondary firmware



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

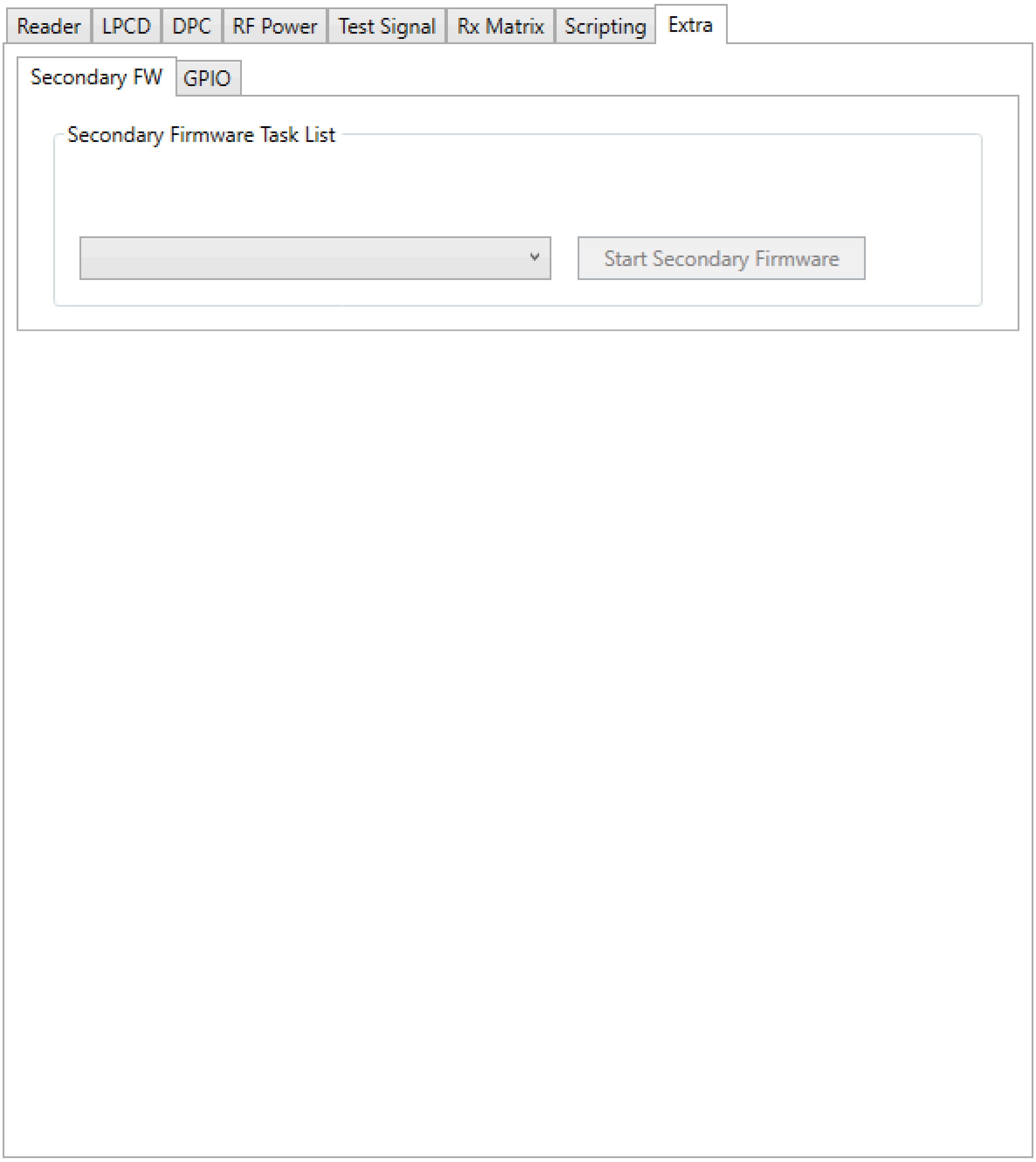



**Figure 111. PN7462 Secondary firmware**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

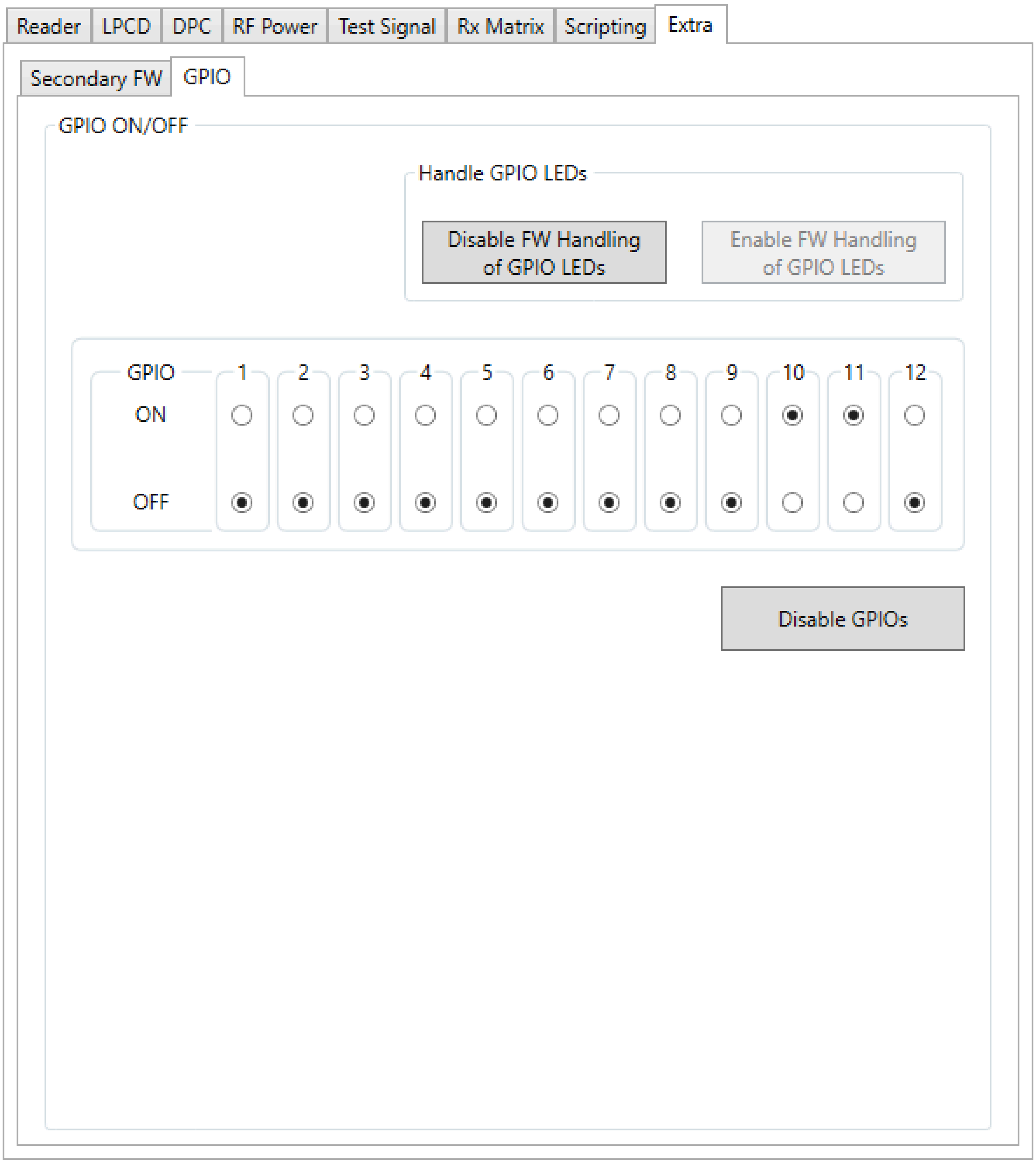



**Figure 112. PN7462 GPIO**



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

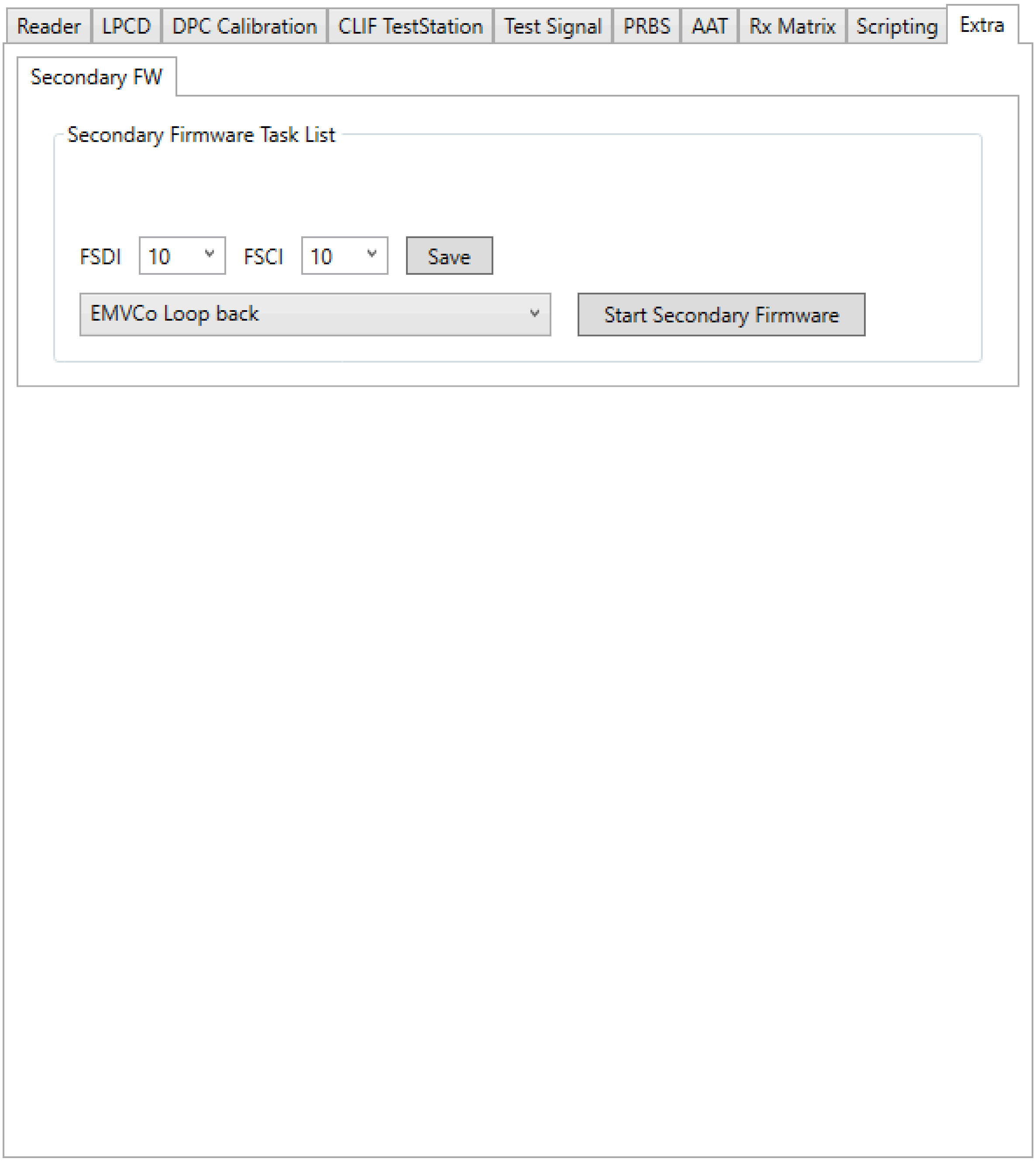



**Figure 113. PN7642-Configuration**

# 5 Scripting



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

## 5.1 Commands

### 5.1.1 Logging commands

Logging commands are meant to enable the users to display a relevant message on log. A message can be some general information to keep track of what operation has been performed, a warning about may go wrong or an error. It may be required to display these messages from time to time.

**Info**

Displays any general information on log. It takes up the string message as a parameter and displays that message on log. General information includes any information that is not a warning, error, or critical issue.

**Usage**: `Info message`

**Warn**

Displays any Warning message on log. A warning should be given whenever there is an issue that can still be handled. This command takes up the string message as a parameter and displays that message as a warning on log.

**Usage**: `Warn message`

**Error**

Displays any Error messages on log. An error is an issue that cannot be insanely handled and therefore it is required to be reported. This command takes up the string message as a parameter and displays that error message on log.

**Usage**: `Error message`

### 5.1.2 Jump commands

Almost every programming and scripting language has control flow mechanism and nncscript is no exception. Jump commands can change the flow of execution of statements. Jumps commonly make use of Labels to achieve the change in control flow. The first statement executed after the jump is the statement immediately following the given label. Depending on the type of jumps, there are different variants of jump commands accepting different set of parameters.

***Note:*** *All the jump commands except `Jump` in nncscript are conditional jumps i.e program flow is diverted only if a condition is true.*

**Jump**

The command jumps to the location specified by the parameter `the Label`

**Usage**: `Jump theLabel`

**JZ - Jump On Zero**

The command accepts 2 parameters `"theVariable"` and `"the Label"`. Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the a label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter `"theLabel"` only if parameter `"theVariable"` is evaluated as zero. JZ performs a conditional jump wherein the condition is that `theVariable` is zero.

**Usage**: `JZ theVariable theLabel`



© 2023 NXP B.V. All rights reserved.

**JNZ - Jump On NonZero**

JNZ will take the execution flow to the location specified by `theLabel` if `"theVariable"` evaluates as a non-zero value.

**Usage**: `JNZ theVariable theLabel`

**JumpIfBitN**

The command takes 3 parameters `"theVariable"`, `"u8theBit Number"` and `"theLabel"`. Calling this command in a script will take the execution flow to the label specified by `"theLabel"` if the bit specified by `"u8theBitNumber"` in `"theVariable"` is set.

**Usage**: `JumpIfBitN theVariable u8theBitNumber theLabel`

**JumpIfNotBitN**

Calling this command in a script will take the execution flow to the label specified by `"theLabel"` if the bit specified by `"u8theBitNumber"` in `"theVariable"` is **not set**.

**Usage**: `JumpIfNotBitN theVariable u8theBitNumber the Label`

**JE - Jump On Equal**

The command accepts 3 parameters `"theVariable"`, `"u32Number"` and `"theLabel"`. Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter `"theLabel"` only if parameter `"theVariable"` is evaluated **equal** to the value specified by parameter `"u32Number"`.

**Usage**: `JE theVariable u32Number theLabel`

**JG - Jump on Greater**

The command accepts 3 parameters `"theVariable"`, `"u32Number"` and `"theLabel"`. Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter `"theLabel"` only if parameter `"theVariable"` is evaluated **greater** than the value specified by parameter `"u32Number"`.

**Usage**: `JG theVariable u32Number theLabel`

**JGE - Jump on Greater or Equal**

The command accepts 3 parameters `"theVariable"`, `"u32Number"` and `"theLabel"`. Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter `"theLabel"` only if parameter `"theVariable"` is evaluated **greater than or equal** to the value specified by parameter `"u32Number"`.

**Usage**: `JGE theVariable u32Number theLabel`

**JL - Jump On Lesser**

The command accepts 3 parameters `"theVariable"`, `"u32Number"` and `"theLabel"`. Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter `"theLabel"` only if



All information provided in this document is subject to legal disclaimers.



© 2023 NXP B.V. All rights reserved.

parameter "`theVariable`" is evaluated **less** than the value specified by parameter "`u32Number`".

**Usage**: `JL theVariable u32Number theLabel`

**JLE - Jump On Lesser or Equal**

The command accepts 3 parameters "`theVariable`", "`u32Number`" and "`theLabel`". Parameter `theVariable` takes up a variable name on which condition check is performed and `theLabel` denotes the label name to which the execution has to be diverted. Execution flow jumps to location specified by the parameter "`theLabel`" only if parameter "`theVariable`" is evaluated **less than or equal** to the value specified by parameter "`u32Number`".

**Usage**: `JLE theVariable u32Number theLabel`

### 5.1.3 EEPROM commands

EEPROM commands facilitate performing operations on Contents of EEPROM like reading, writing, and dumping the contents of EEPROM in a file.

**ReadEEPROM_U8**

This command enables reading the contents of EEPROM at a specified address. The command accepts 2 parameters "`EEAddress`" and "`theVariable`". The command reads the EEPROM contents at the address given by parameter "`EEAddress`" and stores the read values in variable denoted by the parameter "`theVariable`".

**Usage**: `ReadEEPROM_U8 EEAddress theVariable`

**WriteEEPROM_U8**

This command enables performing write operation in EEPROM at a specified address. The command accepts 1 parameter "`EEAddress`" We can write contents of EEPROM at the address specified by "`EEAddress`"

**Usage**: `WriteEEPROM_U8 EEAddress`

**DumpEEPROMToFile**

Using the command we can dump the EEPROM contents in a file. The command accepts 1 parameter "`FileName`" which is used to specify the filename.

**Usage**: `DumpEEPROMToFile FileName`

**LoadEEPROMFromFile**

Using the command we can load the contents of EEPROM importing information from a file. The command accepts one parameter "`FileName`" which is used to specify the filename from which the configurations have to be loaded.

**Usage**: `LoadEEPROMFromFile FileName`

**Note:** For PN7642 hardware, the EEPROM is split into two halves. One is User area and other is Secure Lib area.

**Supported labels are:**

1. NONE.
2. USER_AREA.
3. SECURE_LIB_CONFIG.


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

**To configure use the below:**

1. ReadEEPROM_U8 EEAddress theConfig
2. ReadEEPROM_U8 EEAddress theVariable theConfig

### 5.1.4 Register commands

Register commands facilitate performing operations on Contents of a specific register. These commands provide basic operations like reading, writing the contents of a Register.

**ReadRegister**

The first `ReadRegister` command either takes HEX address of the register or IC-specific valid name of register as a parameter and returns the contents of that register. The other variant "`ReadRegister Register theVariable`" takes 2 parameters. One is the register HEX Address or IC-specific register name and the other parameter is a variable in which the contents of the specified registers is stored.

**Usage**: `ReadRegister`command has 2 variants:

- `ReadRegister Register`
- `ReadRegister Register theVariable`

**WriteRegister**

`WriteRegister` command accepts 2 parameters "`Register`" i.e register HEX Address or IC-specific register and the other parameter is "`ValueOrVariable`" wherein we give the value or the variable name who's values is to be written in the specified register.

**Usage**: `WriteRegister Register ValueOrVariable`

### 5.1.5 Protocol tuning commands

**LoadProtocol**

`LoadProtocol` command loads the specified RF Protocol among the following:

- RM_A_106
- RM_A_212
- RM_A_424
- RM_A_848
- RM_B_106
- RM_B_212
- RM_B_424
- RM_B_848
- RM_F_212
- RM_F_424
- RM_I15693_Tx26_Rx26_ASK10
- RM_I15693_Tx26_Rx26_ASK100
- RM_I15693_Tx26_Rx53_ASK10
- RM_I15693_Tx26_Rx53_ASK100
- RM_I180003m3_TX_TARI_18_88us_RX_Manch424_4_106
- RM_I180003m3_TX_TARI_18_88us_RX_Manch424_2_212
- RM_I180003m3_TX_TARI_18_88us_RX_Manch848_4_212
- RM_I180003m3_TX_TARI_18_88us_RX_Manch848_2_424


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

- RM_I180003m3_TX_TARI_9_44us_RX_Manch424_4_106
- RM_I180003m3_TX_TARI_9_44us_RX_Manch424_2_212
- RM_I180003m3_TX_TARI_9_44us_RX_Manch848_4_212
- RM_I180003m3_TX_TARI_9_44us_RX_Manch848_2_424

`LoadProtocol` command accepts `"rf_protocol_type"` as its only argument and loads the given protocol.

**Usage**: `LoadProtocol rf_protocol_type`

**Ping**

`Ping` command does not require any argument. `LoadProtocol` is performed before calling `Ping` in a script and accordingly `Ping` command pings for the corresponding type for which `LoadProtocol` has been performed. For instance if we do the following:

Ping command returns the card response if the card is present in the field of antenna and returns "----" if no card is present.

**Usage**: `Ping`

Example:

```
LoadProtocol RM_A_106
```

```
Ping
```

Here `Ping` command performs ping for Type A.

### 5.1.6 Variables

Variables are containers of values. Variable names must be prefixed with '$' sign. In nncscript, variables must declared before being assigned or modified. We need not specify the type of variable while declaration. The names of the variables are Case Sensitive and valid characters for variable names are A-Za-z0-9. Given below are the methodologies to declare and initialize variables.

**Var**

`Var` keyword is used to declare a variable e.g `Var count`. `Var` is actually a command which accepts variable name as a parameter and creates a variable of that name. While declaring a variable, it is not prefixed with '$' sign.

**Usage**: `Var theVariable`

**Set**

`Set` keyword is used to initialize a variable with a value. We use $ sign prefixed with variable name whenever accessing, setting, or modifying a variable value. e.g `Set $count 10`.

**Usage**: `Set theVariable theValue`

***Note:*** *A variable must be declared before it is initialized.*

#### 5.1.6.1 Operations on Variables

**Increment**

`Increment` keyword is used to increment a variable value i.e increasing the current value of a variable by 1. But after 0xFFFFFF FF, it rolls over the value to 0. `Increment` takes 1 argument `the Variable` which is the name of the variable to be incremented.


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

**Usage**: `Increment theVariable`

**Decrement**

`Decrement` keyword is used to decrement a variable value i.e increasing the current value of a variable by 1. But after 0, it rolls over to 0xFFFFFFFF. `Decrement` operation takes 1 argument `the Variable` which is the name of the variable to be decremented.

**Usage**: `Decrement theVariable`

**Mask**

`Mask`operation is used to logically Mask a variable's value. Masking with 0 would result in `theVariable` to be set to 0 and masking with 0x FFFFFFFF would not change the value of the variable. `Mask` takes 2 parameters `theVariable` i.e the name of the variable to be masked and `theMask` i.e the mask value to be used.

**Usage**: `Mask theVariable theMask`

**ShiftLeft**

`ShiftLeft` keyword is used to perform a right shift on a variable value by n bits. `ShiftLeft` takes 2 argument `theVariable` which is the name of the variable and `nbits`, which is the number of bits by which the variable value is to be left shifted.

**Usage**: `ShiftLeft theVariable nBits`

**ShiftRight**

`ShiftRight` keyword is used to perform a left shift on a variable value by n bits. `ShiftRight` takes 2 argument `theVariable` which is the name of the variable and `nbits`, which is the number of bits by which the variable value is to be right shifted.

**Usage**: `ShiftRight theVariable nBits`

### 5.1.7 Device commands

Device commands comprises of simple commands involving Opening and Closing of device i.e the board.

**Open**

`Open` command is used to open or establish connection with the stated device. This command has one string parameter `TheDevice` to give the board/device name which is to be connected.

**Usage**: `Open TheDevice`

**Close**

`Close` command also takes the board name as its argument to close the connection with the stated device. This command takes one string parameter `TheDevice` to give the board/device name which is to be closed.

**Usage**: `Close TheDevice`

### 5.1.8 RF field commands

RF field commands help the user to simply turn the RF field on or off.



© 2023 NXP B.V. All rights reserved.

**RFField**

`RFField` commands just take one argument "`On_Off_Reset`" to notify whether the RF field should be turned On or Off. `RFField On` turns the field on, similarly `RFField Off` turns the RF field off.

Usage: `RFField On_Off_Reset`

## 5.2 Samples

### 5.2.1

Here are some sample scripts to demonstrate how specific tasks can be performed using scripting in NxpNfcCockpit.

**rc663_sample_1.nncscript**

The script demonstrates read/write operation on specific Registers.

```
; Establish the connection with RC663 board.
Open RC663

; Perform Field On / OFF
RFField On
Sleep   10                      ; Sleep 10 milli seconds
RFField Off

;ReadRegister by specifying Register Address or Name
ReadRegister  0x5f              ;FABCAL_REG
ReadRegister  VERSION_REG       ;0x7f

;WriteRegister by specifying Register Address or Name
WriteRegister 0x00 0x00         ;COMMAND_REG
WriteRegister COMMAND_REG 0x00  ;0x00

; If it's hexadecimal, prefix with 0x. Else No prefix for decimal
ReadEEPROM_U8 0xC0
ReadEEPROM_U8 192

;Close the connection with the board.
Close
```

**ReadRegisterInLoop.nncscript**

The script loops for specified number of times and performs RFField ON-OFF continuously

```
; Establish the connection with RC663 board.
Open RC663

; Declare a variable.
Var LoopCount

; Set the value for the variable.
Set $LoopCount 15

; Demonstration of continous RF Filed On / OFF inside a LOOP.
```



© 2023 NXP B.V. All rights reserved.

```
            :LOOP_ENDLESS
                RFField On
                Sleep $LoopCount
                RFField Off
                Sleep 300

                Decrement $LoopCount
                JNZ $LoopCount :LOOP_ENDLESS

            ; Establish the connection with RC663 board.
            Close
```

**rc663_PingA.nncscript**

The script loops for specified number of times and sends Ping Request for Type A

```
            ; Establish the connection with RC663 board.
            Open RC663

            ; Load Protocol RM_A_106
            LoadProtocol RM_A_106

            ; Declare a variable.
            Var Loop_count

            ; Set the value for the variable.
            Set $Loop_count 100

            ; Demonstration of continous REQA command inside a LOOP.
            :LOOP_ENDLESS
               RFField On
               Ping
               RFField Off

               Decrement $Loop_count
               JNZ $Loop_count :LOOP_ENDLESS

            ; Establish the connection with RC663 board.
            Close
```



© 2023 NXP B.V. All rights reserved.

# 6 Secondary firmware

## 6.1 Introduction

Secondary firmware is the application that runs as an RTOS task. Within the microcontroller host, where ever applicable, secondary applications like EMVCo loop back, etc. can be implemented. These secondary applications are treated as RTOS tasks and can be started and stopped from the GUI.

## 6.2 Files

### 6.2.1 Port specific files

These files change from controller to controller.

### 6.2.2 Implementation specific files

These files change based on top-level application, e.g. top-level application may choose to initialize RTOS.

### 6.2.3 Portable files

Generally, these file should not undergo any change.

## 6.3 Porting

Porting to your own microcontroller

### 6.3.1 VCOM Porting

VCOM Porting Guideline

This part of the document gives an overview on the steps needed to use/port this application on different platforms other than the default LPC1769.

**Pre-Conditions**

**phPlatform / OSAL / BAL Porting**

It is mandatory that phPlatform and relevant OSAL and BAL porting has already been performed onto the target platform. Apart from the process of receiving Command Frame from the PC Host and sending Response Frame back to the PC Host, this application directly uses the ``phPlatform`` porting layer of NxpNfcRdLib to perform low-level operations.

**VCOM/RS232 Serial Interface**

It is mandatory that the microcontroller Host has some method/approach to expose either a USB VCOM CDC Class interface to the PC Host, or at the minimum a serial port interface.

**Communication between microcontroller Host and DUT**

If the Pre-Conditions of phPlatform / OSAL / BAL Porting has already been met, and reference examples provided with NxpNfcRdLib are already running on the target platform, the communication between microcontroller Host and Device Under Test (DUT) should already be fully functional. Nothing special should be required for porting from the GUI.

**Communication between PC Host and microcontroller Host**


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

The way VCOM/RS232 is implemented is specific to the microcontroller and not explained in this document. The end user is expected to port APIs of the group as listed in VCOM Interface Group.

To verify whether such porting for Communication between PC Host and microcontroller Host is complete, Loop Back (For Testing) are already implemented inside this application. A separate test application to be run on PC is provided with NXP NFC Cockpit: Microcontroller BAL - PC Test Application

**Microcontroller BAL - PC Test Application**

``UcBalPCTestApp.exe`` is a standalone application implemented to verify the porting of Communication between PC Host and microcontroller Host. As of writing of this document, the following commands are available for testing.

| General Commands | Description |
|---|---|
| /Usage | Prints the Usage |
| /help | Same as /Usage |

| GPIO Commands | Description |
|---|---|
| /GetBusy | Get Value of Busy PIN |
| /GetIRQ | Get Value of IRQ PIN |
| /SetDWL=1 | Set Download Pin to 1 |
| /SetDWL=0 | Set Download Pin to 0 |

| Tx/Rx Commands | Description |
|---|---|
| /Echo | Send a Dummy frame from PC to ucHost and check if it is Echoed back |
| /TxAscending | Send a Dummy frame from PC to ucHost, in ascending order |
| /RxAscending | Get a Dummy frame from ucHost to PC, in ascending order |
| /RxDescending | Receive a Dummy frame from ucHost to PC, in descending order |

| Secondary FW Commands | Description |
|---|---|
| /GetTaskCount | Get Secondary FW Task Count |
| /GetTaskNames | Get Secondary FW Task Names |
| /StartTask=0 | Start Task[0], if present |
| /StartTask=1 | Start Task[1], if present |
| /StopTask | Stop Task, if running |

| Version Commands | Description |
|---|---|
| /GetRdMajorVer | Get Nxp Nfc Reader Library Major Version |
| /GetRdMinorVer | Get Nxp Nfc Reader Library Minor Version |
| /GetRdDevVer | Get Nxp Nfc Reader Library Development Version |


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

| Version Commands | Description |
|---|---|
| /GetRdString | Get Nxp Nfc Reader Library Complete Version string |
| /GetuCMajorVer | Get uC (Host Controller) firmware Major Version |
| /GetuCMinorVer | Get uC (Host Controller) firmware Minor Version |
| /GetuCDevVer | Get uC (Host Controller) firmware Development Version |
| /GetuCString | Get uC (Host Controller) firmware version in ASCII string format. |
| /GetuCDateTime | Get uC (Host Controller) firmware Compiled DateTime in ASCII string format. |
| /GeFrontEnd | Get the Reader IC type (RC663, PN5190, PN5190, etc...) |

Note: The above table may not be up-to-date. Running ``UcBalPCTestApp.exe`` (without any parameters) or ``UcBalPCTestApp.exe /Help``, prints the latest implemented Commands and description by the version of ``UcBalPCTestApp.exe`` supplied in this package.

``UcBalPCTestApp.exe`` is written to incrementally confirm and check what is working and what is not working between the PC Host and microcontroller Host, without running the complete GUI.

**VCOM Interface Porting**

Separate VCOM interface block for porting between PC Host and microcontroller Host. For VCOM to be ported, below API's need to be implemented.

- **phUcBal_VCOM_Init**
  Initialize connection between PC Host and microcontroller Host.
- **phUcBal_VCOM_IsConnected**
  Is PC Host and microcontroller Host connected? Returns: Connection status Return values:: 0 Not connected | 1 Connected
- **phUcBal_VCOM_Write**
  Send RxBuffer from microcontroller Host to PC Host Returns: Number of bytes written phUcBal_VCOM_Read Get data from PC Host into microcontroller Host.
- **phUcBal_VCOM_Read**
  Get data from PC Host into microcontroller Host. Returns: Number of bytes read

### 6.3.2 IAP (In Application Programming)

In-Application-Programming(IAP) is supported with phUcBal with the help of a bootloader. It allows to write user's firmware from the already running bootloader.

**Secondary FW Upgrade**

Performing Secondary FW Upgrade, IAP (In Application Programming), build system changes to support required memory map, strategy for the bootloader to ensure validity of the secondary application during upgrade mode, hard/soft reboot post download, vector remapping in case of moving from primary to secondary application, etc. are advanced topics in themselves and therefore not covered in this document. The design of Secondary Firmware Upgrade is not mandatory for actually running Secondary Firmware tasks from the GUI. The implementation and architecture of Secondary Firmware Upgrade is shared in phNncBootloader and is out of scope of this document.

**Note:** IAP supported for RC663, PN5180, and PN5190 only.


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

## 6.4 Protocol

**Protocol Overview**

Overview of the Protocol/Frame Format between PC and microcontroller.

Since NxpNfcCockpit is designed for RF Tuning of NFC controller, the semantics of the protocol between PC and the microcontroller is inspired form ISO-7816 APDU Structure with a forward looking idea that the the users who are interested in understanding this implementation would be more or less aware of the ISO-7816 APDU Structure.

Note: This implementation is inspired from ISO-7816 APDU Structure but only uses its semantics for simplicity. It's neither full nor even partially compatible to ISO-7816's APDU implementation.

**Command Frame**

All the commands follow the the following structure.

| CLA | INS | P1 | P2 | Lc | (Command) Payload |
|---|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 1 byte | 2 bytes | Lc bytes |

**Response Frame**

The response follows the following structure.

| CLA | INS | S1 | S2 | Lr | (Response) Payload |
|---|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 1 byte | 2 bytes | Lr bytes |

Note: If you compare ISO 7816 and the above frames, you can see that Le, SW1 SW2 are missing.

**Description of fields in Command Frame and Response Frame**

The below table provides description of each field.

| Field | Description |
|---|---|
| CLA | The main group of command. e.g. Trans Receive , GPIO Control, etc. |
| INS | The instruction for that group |
| P1 | Parameter 1 |
| P2 | Parameter 2 |
| Lc | Length of Command Payload, 2 Bytes, LSB First |
| Lr | Length of Response Payload, 2 Bytes, LSB First |
| S1 | Status 1: API Status |
| S2 | Status 2: Component Code |
| S1S2 | For some of the Commands Status 1 and Status 2 may be merged, and represent values from phStatus_t of NxpRdLib APIs. |

Within the framework of this protocol, different Command Groups are sent and received between the PC and the microcontroller.



© 2023 NXP B.V. All rights reserved.

## 6.5 Features / functionalities

Below are the set of functionalities / commands between application (NxpNfcCockpit) and microcontroller.

- TransReceive
- GPIO
- Configuration
- Secondary Task Management
- Versioning
- Loop Back

### 6.5.1 Transreceive

Send (Transmit) and receive commands/data between microcontroller and the NFC Device. The APIs/ Commands in this module itself have no notion/idea about the connection between microcontroller Host and the Device Under Test (DUT). On the contrary, it depends on the platform that is initialized and relevant porting. **Class (CLA) value is 0x01**.

There are three variants of Transceive command. There are

| | |
|---|---|
| **Transition (INS: 0x05)** | Sends the command to microcontroller and microcontroller does a direct exchange to Reader IC. |
| **Reception (INS: 0x0E)** | Sends a command to microcontroller to read the data given by Reader IC to microcontroller. |
| **Transmission and Reception (INS: 0xFD)** | Sends the command to microcontroller and microcontroller does a direct exchange to Reader IC.<br>Sends a command to Microcontroller to read the data given by Reader IC to microcontroller. |

**Table 1. Format of Command**

| CLA | INS | P1 | P2 | LC | Payload |
|---|---|---|---|---|---|
| CLA_TransReceive | TransReceive_INS_Tx | 0 | 0 | Len of Payload | Payload Data |
| | TransReceive_INS_Rx | 0 | 0 | 0 | Not Applicable |
| | TransReceive_INS_TRx | 0 | 0 | Len of Payload | Payload Data |

**Table 2. Format of Response**

| CLA | INS | S1 | S2 | LR | Payload |
|---|---|---|---|---|---|
| CLA_TransReceive | TransReceive_INS_Tx | 0 | 0 | 0 | Not Applicable |
| | TransReceive_INS_Rx | 0 | 0 | Len of Payload | Payload Data |
| | TransReceive_INS_TRx | 0 | 0 | Len of Payload | Payload Data |

### 6.5.2 GPIO control

GPIO Control module helps manage GPIOs of microcontroller Host between the microcontroller Host and Device Under Test (DUT). The APIs/Commands in this module internally depend on ``phPlatform`` to check the values of the GPIOs and set/get their values. **Class (CLA) value is 0x10**.

p

***Note:*** *During platform initialization, the relevant GPIOs should already have been initialized and set as expected.*

Below are the variants of GPIO commands.



All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

| | |
|---|---|
| **GetV (INS: 0x0E)** | Reads the GPIO value.<br>***Note:*** *For PN5180, returns value using IRQ_STATUS Register IRQ strategy. Else, returns stored IRQ value, which was saved earlier using phUcBal_Config_StoreIRQPinValue* |
| **SetV (INS: 0x05)** | Checks if the requested PIN is valid and sets the PIN. |
| **WaitForHigh (INS: 0xA1)** | Wait for requested GPIO to go High. |
| **WaitForLow (INS: 0xA0)** | Wait for requested GPIO to go Low. |

Table 3. Format of Command

| CLA | INS | P1 | P2 | LC | Payload |
|---|---|---|---|---|---|
| CLA_GPIO | GPIO_INS_SetV | GPIO | Value | 0 | Not Applicable |
| | GPIO_INS_GetV | GPIO | 0 | 0 | Not Applicable |
| | GPIO_INS_WaitForHigh | GPIO | 0 | 0 | Not Applicable |
| | GPIO_INS_WaitForLow | GPIO | 0 | 0 | Not Applicable |


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 4. Format of Response

| CLA | INS | S1 | S2 | LR | Payload |
|---|---|---|---|---|---|
| CLA_GPIO | GPIO_INS_SetV | 0 | 0 | 0 | Not Applicable |
| | GPIO_INS_GetV | 0 | 0 | 1 | Value |
| | GPIO_INS_WaitForHigh | 0 | 0 | 0 | Not Applicable |
| | GPIO_INS_WaitForLow | 0 | 0 | 0 | Not Applicable |

Table 5. GPIO names and its values

| GPIO | RESET | IRQ | BUSY | Download |
|---|---|---|---|---|
| Number | 1 | 2 | 5 | 6 |
| Value | 00 or 01 | | | |

### 6.5.3 Configuration

Configuration module is basically used to manage runtime configurations. For example, Depending on the state of PN5180 (Device Under Test (DUT)) and it's (Normal mode vs Secure FW Upgrade mode) the IRQ pin handling would change between the Micro Controller Host and Device Under Test (DUT).

This component ensures that such handling for the current case (and more complex future scenarios) is managed in a specific module.

**Provided Configurable Parameters**

- **phUcBal_Config_WaitBeforeRX_Strategy/phUcBal_Config_WaitBeforeTX_Strategy**
  It helps in configuring TX/RX Strategy during Transrecieve with one of the three options
  - WaitBefore_Immediate : Just go and do TX/RX.
  - WaitBefore_WaitForIRQHigh : Wait for the IRQ Pin to go High.
  - WaitBefore_WaitForBusyLow : Wait for the Busy Pin to go Low.
- **phUcBal_Config_ConfigIRQPollStrategy**
  It helps in configuring IRQ Pin for PN5180. If IRQ Pin is used for Test Bus, we cannot use interrupts. Hence we use the following strategies for IRQPolling.
  - IRQHandling_CheckTestBus : Check if TestBus is enabled and take decision dynamically to switch between ReadRegsiter or UseInterrupts strategy
  - IRQHandling_ReadRegsiter : Invoke read register of IRQ_STATUS Register, when IRQ pin is used for TestBus because interrupts cannot be used
  - IRQHandling_UseInterrupts : Return to normal usage of interrupts
- **phUcBal_Config_IRQIsrHandling**
  It helps in configuring CLIF IRQ. Only when Secondary Firmware is running, HAL on MicroController Host would need this information. Else, this IRQ is not of any purpose to hal on Micro Controller Host. By default only share this with HAL Running on PC Host.
  - IRQIsrHandling_Consume : Consume CLIF IRQ and do not expose it to PC Host
  - IRQIsrHandling_Share : Share CLIF IRQ with HAL on PC Host
- **phUcBal_Config_StoreIRQPinValue**
  NxpNfcRdLib and phPlatform layer stores information of IRQ Pin uniquely. By design, IRQ Pin value denotes whether an IRQ is yet to be processed and not whether the IRQ Pin is high or low. Since Device Under Test (DUT) HAL is instantiated both in PC Host and Micro Controller Host, we need to store it in case the PC Host asks about that information
- **phUcBal_Config_StoreICInitFailed**
  If IC Initialization failed, store the state that the IC Init has failed and share when asked for this information.
- **phUcBal_Config_I18000p3m3_Commands**


All information provided in this document is subject to legal disclaimers.
© 2023 NXP B.V. All rights reserved.

phUcBal also offer to configure I18000p3m3 commands since I18000p3m3 is very time critical that PC to microcontroller delay is not tolerated.
- phUcBal_Config_SelectCommand
- phUcBal_Config_SelectCommandLength
- phUcBal_Config_NumValidBitsinLastByte
- phUcBal_Config_BeginRoundCommand
- phUcBal_Config_TSprocessing

- **phUcBal_Config_GetIrqType**
  This helps in knowing what type of IRQ implementation is being followed for the communication between MicroController and Device under test. IRQ can be RISING EDGE/FALLING EDGE/EITHER EDGE

### 6.5.4 Secondary Tasks management

Within the Micro Controller Host, where ever applicable, Secondary Applications like EMVCo loop Back, etc. can be implemented. In the current implementation, these secondary applications are treated as RTOS Tasks and can be started and stopped from the GUI. At a time, only one secondary application is allowed to be run.

Entering Secondary FW Upgrade Mode and allowing over-write of Secondary FW from the PC is an optional feature. An application can be built that does not support Secondary FW Upgrade, but still such an application can be downloaded on Micro Controller Host via relevant debugger/programmer, and secondary tasks can be triggered/stopped from the GUI.

**Provided Functions**

- **phUcBal_SECFw_GetTaskCount**
  Gives the number of tasks implemented by the Secondary Application. This API depends on the structure gkphUcBal_SECFw_Tasks that is to be filled in statically by the Secondary Application.
- **phUcBal_SECFw_GetTaskName**
  Give the task name of tasks implemented by the Secondary Application. Before calling this API, ensure phUcBal_SECFw_GetTaskCount is invoked. Input: **P1** holds the task number. This API depends on the structure gkphUcBal_SECFw_Tasks that is to be filled in statically by the Secondary Application.
- **phUcBal_SECFw_StartAppTask**
  Start the secondary Application. Before calling this API, ensure phUcBal_SECFw_GetTaskCount and phUcBal_SECFw_GetTaskName is invoked.
- **phUcBal_SECFw_StopAppTask**
  Stop the previous running RTOS Task
- **phUcBal_SECFw_CanUpgrade**
  This API checks if it is possible to upgrade this app through secondary FW Upgrade mechanisam. If this feature is avaialble, the GUI can download a new Binary can be uploaded to the controller without use of debugger/programmer. This feature is only available for PN5180 + LPC1769 and RC663 + LPC1769 reference boards.

### 6.5.5 Versioning

Version information of the running firmware. This allows the PC Application to fetch version information of the pre-compiled binary running on the microcontroller Host at runtime. **Class (CLA) value is 0x0E**.

phUcBal Version can be used extract version from the NxpRdLib and also microcontroller. This can be achieved by loading the INS value of the command frame with one of the below instructions.

Below are the variants of GPIO commands.

**NxpNfcRdLib: NXP NFC Reader Library version information**

| | |
|---|---|
| **RD_Major (INS: 0x01)** | Returns the major version of NxpNfcRdLib |


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

| | |
|---|---|
| **RD_Minor (INS: 0x02)** | Returns the minor version of NxpNfcRdLib |
| **RD_Dev (INS: 0x03)** | Returns the development version of NxpNfcRdLib |
| **RD_String (INS: 0x04)** | Returns the complete ASCII version string of NxpNfcRdLib |

**uC: Microcontroller (can be LPC1769, K82, or user-specific Host Controller) version information**

| | |
|---|---|
| **uC_Major (INS: 0x11)** | Returns the major version of uC firmware |
| **uC_Minor (INS: 0x12)** | Returns the minor version of uC firmware |
| **uC_Dev (INS: 0x13)** | Returns the development version of uC firmware |
| **uC_String (INS: 0x14)** | Returns the complete ASCII version string of uC firmware |
| **uC_Date (INS: 0x15)** | Returns the complete ASCII Compiled date and time string of uC firmware |
| **Frontend (INS: 0x20)** | Returns the Reader IC type (CLRC663, PN5190, PN5190, etc...) |

Table 6. Format of Command

| CLA | INS | P1 | P2 | LC | Payload |
|---|---|---|---|---|---|
| CLA_Version | RD_Major | 0 | 0 | 0 | Not Applicable |
| | RD_Minor | 0 | 0 | 0 | Not Applicable |
| | RD_Dev | 0 | 0 | 0 | Not Applicable |
| | RD_String | 0 | 0 | 0 | Not Applicable |
| | uC_Major | 0 | 0 | 0 | Not Applicable |
| | uC_Minor | 0 | 0 | 0 | Not Applicable |
| | uC_Dev | 0 | 0 | 0 | Not Applicable |
| | uC_String | 0 | 0 | 0 | Not Applicable |
| | C_DateTime | 0 | 0 | 0 | Not Applicable |
| | FrontEnd | 0 | 0 | 0 | Not Applicable |

Table 7. Format of Response

| CLA | INS | S1 | S2 | LR | Payload |
|---|---|---|---|---|---|
| CLA_Version | RD_Major | 0 | 0 | Len of Payload | Payload |
| | RD_Minor | 0 | 0 | Len of Payload | Payload |
| | RD_Dev | 0 | 0 | Len of Payload | Payload |
| | RD_String | 0 | 0 | Len of Payload | Payload |
| | uC_Major | 0 | 0 | Len of Payload | Payload |
| | uC_Minor | 0 | 0 | Len of Payload | Payload |
| | uC_Dev | 0 | 0 | Len of Payload | Payload |
| | uC_String | 0 | 0 | Len of Payload | Payload |
| | C_DateTime | 0 | 0 | Len of Payload | Payload |
| | FrontEnd | 0 | 0 | Len of Payload | Payload |

### 6.5.6 Loopback

Loopback commands can be used to test Loop-back between the PC Host and microcontroller Host.

This can be achieved by loading the INS value of the command frame with one of the below instructions.

**phUcBal_Loopback**

- LoopBack_INS_Out_Ascending -> [0x0A] Return an array in ascending order.


All information provided in this document is subject to legal disclaimers.
© 2023 NXP B.V. All rights reserved.

- LoopBack_INS_Out_Descending -> [0x0D] Return an array in descending order.
- LoopBack_INS_In_Ascending -> [0x1A] Return the length of received packet, expecting ascending order of input data.
- LoopBack_INS_ECHO -> [0xE0] Send same data back.

## 6.6 Execution flow

1. Initialize USB CDC Library so that device can be exposed as a USB Serial Device. This happens on boot up.
2. If PH_UCBAL_MAINTASK_PERFORM_RFONOFF_ON_BOOTUP is enabled, perform RfOnOff so that there can be a quick feedback of the boot up of the FW and a working connection between the microcontroller Host and Device Under Test (DUT).
3. Enter an Infinite loop. Wait for packet from PC Host and process it.
4. If USB is connected, then process, else WFI().
5. Receive a command buffer from the PC.
6. Check if packet is as large as the header as shown in Command Frame
7. For a valid frame, dispatch Command Frame to respective Command Groups
8. Give data back to the PC
9. Wait for the next command


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

# 7 Abbreviations

**Table 8. Abbreviations**

| Acronym | Description |
|---|---|
| AAT | automatic antenna tuning |
| APDU | Application Protocol Data Unit |
| API | application programming interface |
| ARC | Adaptive Receiver Control |
| AWC | automatic wave control |
| AWG | Automatic Wave Generator |
| CLIF | Contactless Interface |
| CRC | cyclic redundancy check |
| CTIF | Contact Interface |
| CTS | CLIF TestStation |
| DPC | dynamic power control |
| DUT | device under test |
| EEPROM | electrically erasable programmable read-only memory |
| FSDI | Frame Size for Proximity Coupling Device |
| FSCI | Frame Size for Proximity Card Integer |
| FW | Firmware |
| GPIO | peneral-purpose input/output |
| HAL | Hardware Abstraction Layer |
| HP | High Power |
| HSU | High-Speed UART |
| HW | hardware |
| ICMFG | Integrated Chip Manufacturing Code |
| LDO | low dropout |
| NFC | near-field communication |
| LP | Low Power |
| LPCD | Low Power Card Detection |
| PAL | Protocol Abstraction Layer |
| PCB | printed-circuit board |
| PCRM | Power Clock and Reset Module |
| PMU | power management unit |
| PRBS | pseudo random binary stream |
| PSP | Product Support Package |
| P2P | Peer to Peer |
| RF | radio frequency |


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 8. Abbreviations*...continued*

| Acronym | Description |
|---|---|
| ROM | read-only memory |
| RTOS | real-time operating system |
| SAM | secure access module |
| SDA | serial data |
| SPI | serial peripheral interface |
| SPIM | SPI controller interface |
| SRAM | static random-access memory |
| SW | software |
| SWD | serial wire debug |
| TXLDO | Transmitter Low Drop Out |
| ULPCD | Ultra Low Power Card Detection |
| USB | universal serial bus |


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

# 8 References


[UM10883] **PN7462AU quick start guide**
http://www.nxp.com/docs/en/user-guide/UM10883.pdf

[AN11706] **PN7462 antenna design guide**
http://www.nxp.com/docs/en/application-note/AN11706.pdf

[AN11022] **CLR663 evaluation board quick start guide**
http://www.nxp.com/docs/en/application-note/AN11022.pdf

[AN11744] **PN5180 evaluation board quick start guide**
http://www.nxp.com/docs/en/application-note/AN11744.pdf




All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

# 9 Legal information

## 9.1 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.

## 9.2 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.

**Limiting values** — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**No offer to sell or license** — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

**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.

**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.

**NXP B.V.** - NXP B.V. is not an operating company and it does not distribute or sell products.

## 9.3 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.

## 9.4 Trademarks

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



© 2023 NXP B.V. All rights reserved.

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

**EdgeVerse** — is a trademark of NXP B.V.

**FeliCa** — is a trademark of Sony Corporation.

**ICODE and I-CODE** — are trademarks of NXP B.V.

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

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

**MIFARE Plus** — is a trademark of NXP B.V.

**MIFARE Ultralight** — is a trademark of NXP B.V.

**NTAG** — is a trademark of NXP B.V.



© 2023 NXP B.V. All rights reserved.

# Tables



# Figures




All information provided in this document is subject to legal disclaimers.
© 2023 NXP B.V. All rights reserved.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

## Contents





All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V. All rights reserved.

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

Date of release: 10 May 2023
Document identifier: UG10025