AN14513

NTAG X DNA - Dual Interface

Rev. 2.0 — 10 November 2025 Application note

Document information

Information	Content
Keywords	NTAG X DNA, Dual Interface, I²C, GPIO, NFC
Abstract	This document provides assistance in NFC system development and describes the main features of NTAG X DNA Dual Interface - I²C, NFC and GPIO coexistence.

1 Introduction

This document focuses on showing how to use NTAG X DNA Dual Interface feature.

Use this document in addition to the NTAG X DNA data sheet ref.[4]. Read the mentioned data sheet in advance.

Note: *This application note does not replace any of the relevant functional specifications, data sheets, or design guides.*

2 Target applications

NTAG X DNA offers a wide area of implementations due to rich features and sophisticated, simple high security. Including, but not limited to, see the following examples:

Accessories

- Cell phones
  - Cases/covers
  - Game controllers
  - Ear pods
  - Battery packs
  - Cables
  - Headsets
  - Camera attachments
  - more
- Wall chargers
- USB-C protocol based
- Input devices (for example, tablet pens)

IoT security

- Smart speakers
- Access points
- Smart thermostats
- Security cameras
- Video doorbells
- Home door locks
- Smart appliances
- Energy storage
- eBikes
- Industrial controllers
- Smart meters

Medical

- Smart injectors
- Authentication of vials

Components

- Rechargeable Batteries

3 Types of applications

3.1 Contactless only

The contactless supply use case is the typical use case of a passive RFID card. The IC is supplied via a 13.56 MHz RF field and communicates via the ISO/IC14443 (Layer 4) interface over LA and LB. All other pins are not used for this use case. All other unused pins shall be either not connected or, for example, as in a flip-chip assembly - shorted/connected.

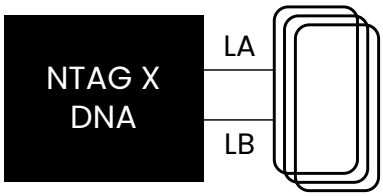

Figure 1. Block diagram - NFC interface only

- Interfaces: NFC
- Possible applications (not limited to):
  - Accessory authentication (Phone Case/Cover/etc.)
  - Consumables authentication
  - NFC Digital Product Passport (NDPP)
- Protocols and crypto: NFC (Type4Tag), ECC Signature Secure Dynamic Messaging, ISO7816-4 incl. mutual auth

Example applications:

Below diagrams show typical NFC only NTAG X DNA applications:

- NFC mobile app-less (Android, iOS) cloud-based Secure Dynamic Messaging based authentication and ECC signature verification Figure 2

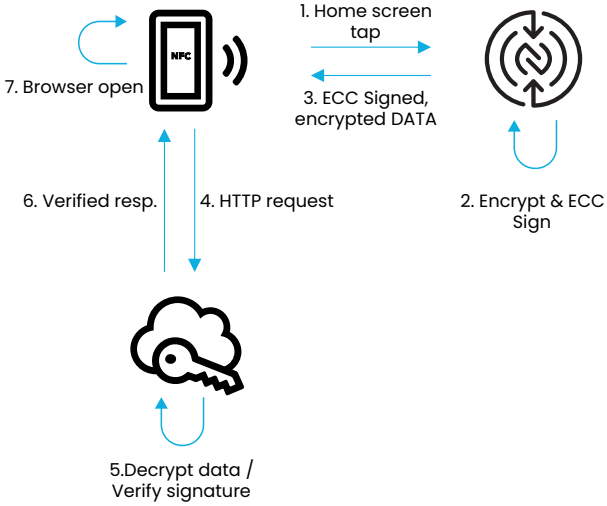

Figure 2. Example for NFC interface only: No NFC mobile application (Home screen NFC reading) - Cloud-based (automatic) system flow

- The diagram below shows a typical NFC-mobile-application-based system. In this case NFC mobile can trigger standard or custom commands to fetch (sensitive) data from NTAG X DNA Figure 3

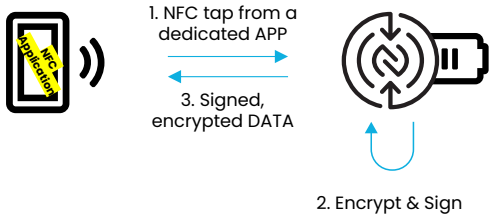

Figure 3. Example for NFC interface only: NFC-mobile-application-based NDPP example

In Figure 4 an example schematic of an NFC only tag is shown.

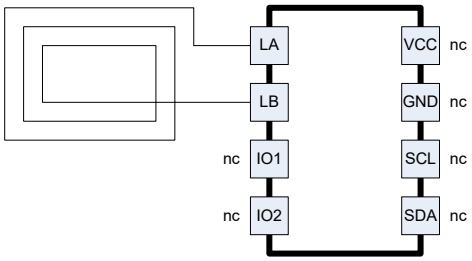

Figure 4. Example schematics - NFC interface only

3.2 Contact-based supply

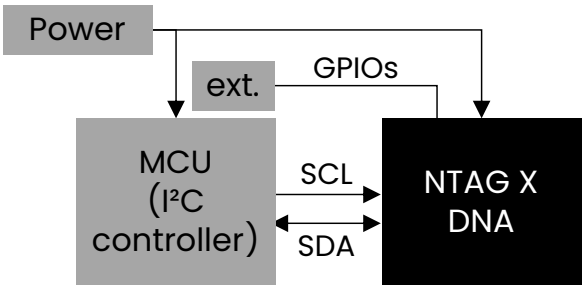

Figure 5. Example schematics - contact-based supply only

In contact-based supply operation the IC is powered via the VCC pin and only communicating via the I²C interface (SCL, SDA pins). The two general purpose IO pins GPIO1 and GPIO2 can be used for arbitrary input and output functionality (for example, sensing a digital input, putting signals as output etc.). Supply voltage range is 1.8 V typical (maximum 2.0 V). I²C communication speed is up to Fast-mode Plus (1 Mbit/s). All pads are directly supplied via the VCC supply pin.

Typical applications are authentication of accessories, storing and using secret key material in a high-secure IC while the connected host (microcontroller) has only a functional application. Current consumption in contact-based application is up to 15 mA (from the VCC pin).

Deep Power-down mode can be entered with a command or timeout. There are configurable wake-up conditions (configurable SDA pulldown, correct I^2C follower address, existing RF field).

General design recommendations:

Power-on reset

- Host controller to control NTAGs V_{CC} (to perform a power-on reset).
- It is possible to supply NTAG X DNA via the MCU/MPU GPIO. The GPIO shall be able to deliver current up to 15 mA.

NTAG X DNA can trigger a Reset via T=1‘ over I^2C protocol.

- Proprietary NXP S-Blocks SE chip reset request/response ref.[3]

NTAG X DNA enables Deep Power Down via T=1‘ over I^2C protocol.

- Proprietary NXP S-Block Deep Power Down request/response ref.[3]

Example applications:

- Accessory authentication: for example, cables, adapters, headsets, docking stations, HID (game controller, keyboard, mouse), printer cartridges, etc.
- Cryptocoprocessor for the host

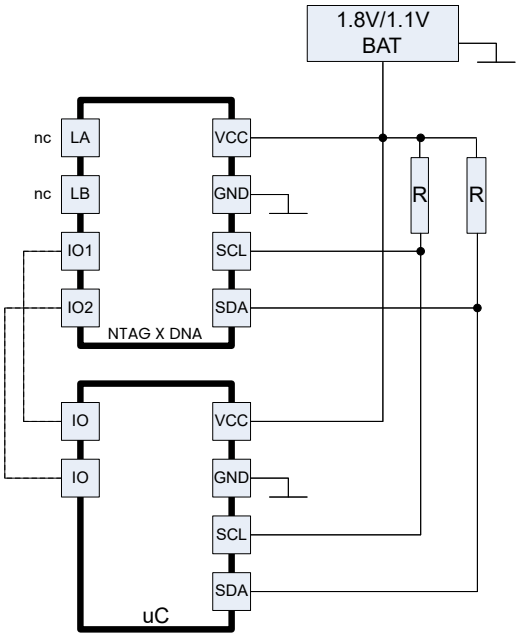

Figure 6. Example schematics - Contact-based supply

3.3 Dual-interface supply

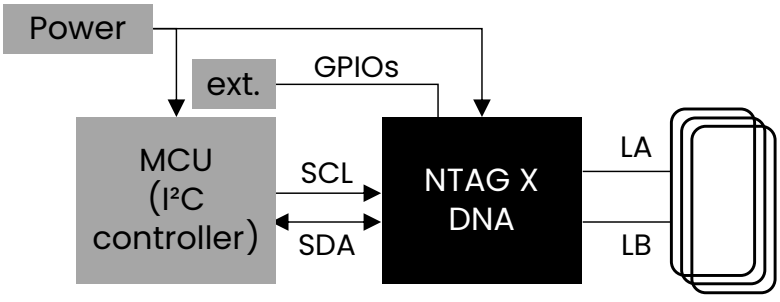

Figure 7. Block diagram: Accessory authentication - VDD powered

In the dual-interface supply mode the IC has a permanent direct connection of the supply from a permanent power supply (for example, battery) via the V_{CC} pin, within the given specification limit (1.0 V to 2 V). In this mode, the NTAG X DNA IC is connected on one side to an I²C controller (for example, a microcontroller) and on the contactless interface communicating with an NFC reader. NTAG X DNA provides mechanisms to control data flow either from MCU → NFC or vice versa. "NFC Pause" Section 4.5 is one of the unique features that NTAG X DNA supports, and only possible in NFC Forum T4T.

A state diagram of NTAG X DNA can be found in chapter Section 6.1.

The NTAG X DNA can be put into a deep Power-down mode (5 µA current consumption). The wake-up can happen either through I²C activity or from an ISO/IEC14443 field.

Dual Interface also covers NFC and NTAG X DNA's GPIO co-existence.

- Interfaces: NFC (antenna connected to LA, LB), I²C, and GPIO
- Possible applications (not limited to):
  - Monitoring/Checking Status (i.e. Smart metering)
  - Activating/Provisioning credentials (i.e. activate new function)
  - Secure MCU firmware update via NFC (instead of BLE)
- Protocols and cryptography: Global Platform T=1 over I²C + NFC (Type4Tag), ECC Signature

Example applications:

- IoT
- Medical wearable devices (for example, a glucose meter)

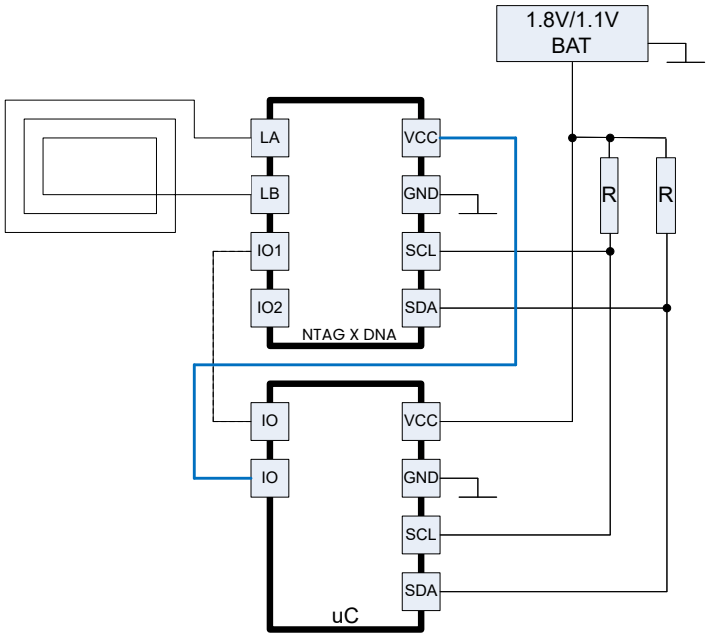

Figure 8. Example schematics - Dual-interface-based supply

3.4 Special use case: Energy harvesting

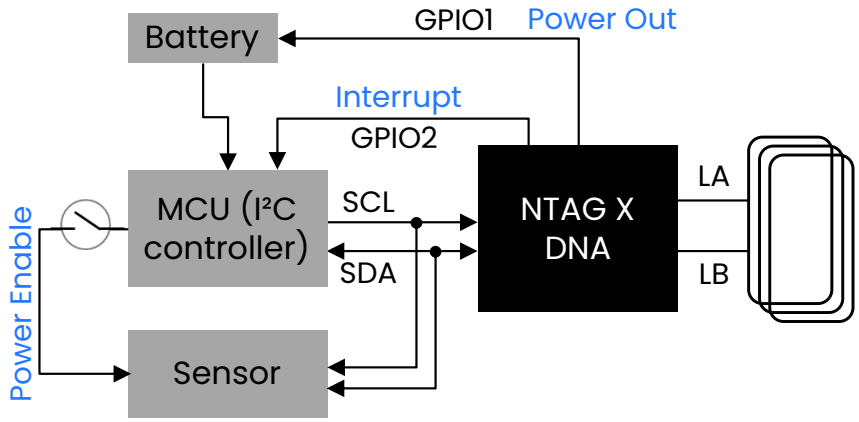

Figure 9. Example block diagram - Energy harvesting

A circuit example is shown in the following figure Figure 10. RF field is coming via LA/LB interface, while the default state of the power harvesting feature is OFF at startup. After a dedicated command (Cmd.ManageGPIO, Operation byte Bit 1 - "MEASURE: execute measurement"), a potential energy yield can be measured. After that, the energy harvesting feature can be enabled, which puts out the regulated voltage on the GPIO1 pin. It can be assumed that an enough large capacitance is connected at the output of the GPIO1 pin to stabilize ISO14443 PCD → PICC Miller pauses.

In case there is no further activity on any of the I²C pins or back supply from VCC, the energy harvesting output is kept enabled until a command disables it or if the RF field gets disabled by the reader. This configuration is always done via software.

The principle of this use case is that a part of the power that is transmitted via the RF field and that is not needed for the internal operation of the NTAG X DNA can be output to a GPIO1 pin and can be used for supplying directly another device, charging a battery or capacitance.

The power output can reach ~10 mW, which at a maximum output voltage of 1.8 V (max. up to 2.0V) results in a current of ~5.5 mA. Note that the available field strength must be large enough (for example, if a Class 1 antenna is targeted, field strength shall be over 3.5 A/m).

For applications where more power is required, "alternative NFC field harvesting" is recommended through NXP wireless charging products or passive components with additional rectifier.

- Interfaces: NFC, I²C, and GPIO
- Possible applications (not limited to):
  - Medical wearables (smart diagnostics)
  - Monitoring/Checking Status (i.e. Smart metering)
  - Activating/Provisioning credential (i.e. activate new function)
  - Firmware update via NFC (instead of BLE)
- Protocol: NFC (Type4Tag) + ECC Signature Secure Dynamic Messaging, ISO7816-4 incl. mutual auth, GPIO Interrupts

Example applications:

- Mobile cases with LEDs (with MCU or without MCU)
- Charging peripheral system

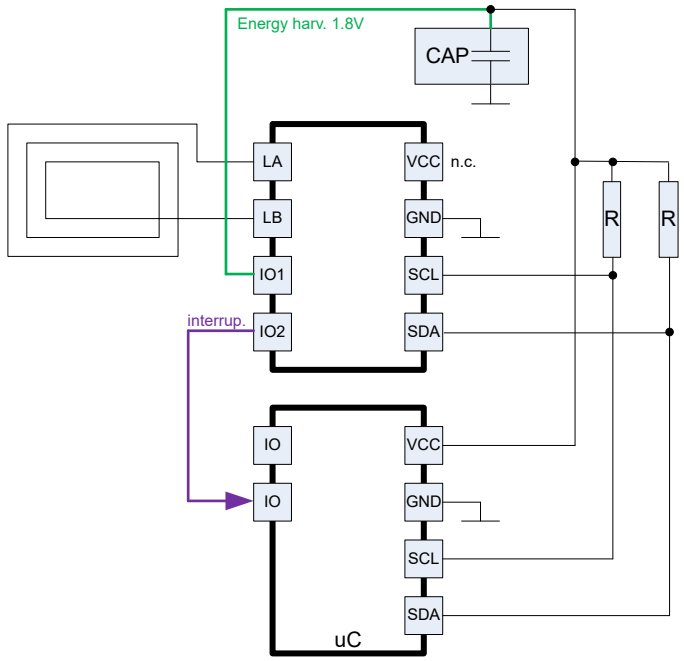

Figure 10. Example schematics - Energy harvesting

Note: *More detailed information can be found in ref.[6]*

3.5 Special use case: IO pin control from contactless (NFC / ISO14443) operation

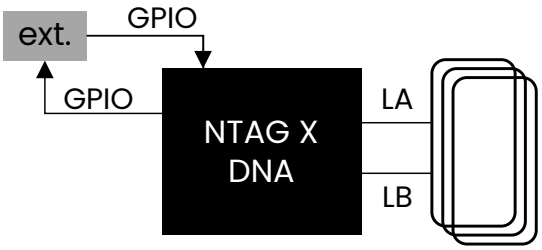

Figure 11. Block diagram: Block diagram: GPIO control from contactless

In this use case, the supply is only available via the contactless interface. The supply pins (VCC, GND) and the I²C pins (SCL, SDA) are not connected and not supplied internally. The two additional pins GPIO1 and GPIO2 can be used in this configuration as GPIOs.

An application where the IOs are used as output would be an LED that is placed at the output of the pin.

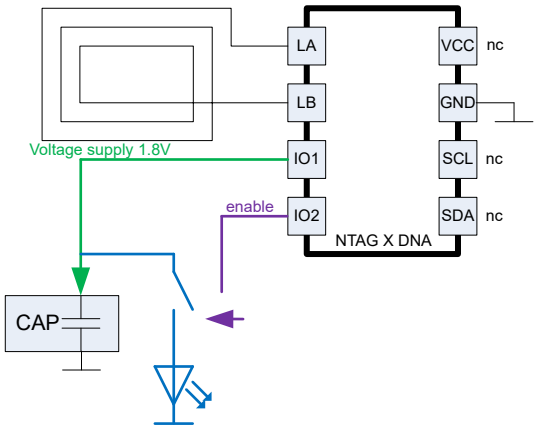

Figure 12. Example schematics - GPIO pin control from NFC, GPIO1 as energy source, GPIO2 as Output

Typical use cases are, where a pin is configured as input is for tag tamper detection or detection of pressing a button.

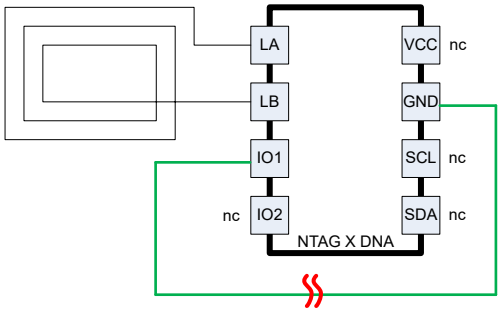

Figure 13. Example schematics - GPIO1 as Input, TagTamper detection

- Interfaces: NFC (LA/LB) + GPIO
- Possible applications (not limited to):
  - NTAG with status detection
  - NFC with action buttons
- Protocols and crypto: NFC (Type4Tag), ECC Signature Secure Dynamic Messaging, Mutual auth.

4 Main features examples

4.1 ManageGPIO

The NTAG X DNA GPIOs can be controlled (set, clear, toggle) by use of the ManageGPIO command from either the I²C or NFC interface. The command may require an authentication and CommMode.FULL, depending on SetConfig settings.

If a GPIO is configured for output, after a power-on reset, the GPIO will be initialized with the state as configured by the GPIOXConfig from SetConfiguration Option 0x11 on the first command after activation. This is independent of the state from a previous activation. Immediately after the PoR, output GPIOs will be in high-impedance (High-Z) state.

The following example shows how only an Authentic NFC reader/writer device can toggle GPIO1. Authentic NFC reader/writer device must perform mutual authentication (asymmetric SIGMA-I or symmetric AES-128/256) prior to sending the Cmd.ManageGPIO command, to create a secure NFC communication channel. Note that the CommMode of the Cmd.ManageGPIO command can be changed by Cmd.SetConfig, Option byte 0x11 (for example, can be set in a way that no authentication is required for toggling GPIOx).

Prerequisites:

1. *Successful mutual authentication. For the example below the following values are assumed:*
   - *$K_{SesAuthMAC}$ = 72 12 CB E0 03 2E FA 8F 7F B0 5F A1 B4 9E B6 E3 (session key generated during AES-128 mutual authentication ref.[5])*
   - *TI = 50 E2 8E 06*
   - *CmdCtr = 02 00*
2. ManageGPIOAccessCondition for Cmd.SetConfig, GPIO Management option is set to:
   - CommMode: 0x1 (communication channel is integrity protected with MAC signatures)
   - AccessCondition: 0x0 (only Key0x00 has access rights for the Cmd.ManageGPIO command).

Table 1. Cmd.ManageGPIO

Step	Command		Data
1	Cmd (INS) of Cmd.ManageGPIO	=	42
2	CommMode	=	CommMode.MAC
3	$K_{SesAuthMAC}$	=	7D 97 4D 25 C6 07 81 88 92 EE 9E 7C 39 C9 07 16
4	GPIONo (GPIO0)		00
5	Operation (SET: set the GPIO State to HIGH (driven)		01
6	CmdHeader (Step 5 || Step 6)		00 01
7	CMD || CmdCtr || TI [|| CmdHeader] [|| CmdData]	=	42 02 00 50 E2 8E 06 00 01
8	MAC($K_{SesAuthMAC}$; CMD || CmdCtr || TI [|| CmdHeader] [|| CmdData])	=	28 D6 3C F6 81 65 B4 F2 19 88 EC 8D 16 56 83 5E
9	MACt	=	D6 F6 65 F2 88 8D 56 5E
10	APDU	>	90 42 00 00 0A 00 01 D6 F6 65 F2 88 8D 56 5E 00
11	R-APDU	<	21 D1 F0 DB C3 1F 16 FC

Table 1. Cmd.ManageGPIO...*continued*

Step	Command		Data
			91 00
12	PICC's MAC	=	21 D1 F0 DB C3 1F 16 FC
13	SW1 SW2	=	91 00
14	Verify Response MAC RC (SW2) || CmdCtr++ || TI [|| RespData]	=	00 03 00 50 E2 8E 06
15	MAC($K_{SesAuthMAC}$; RC || CmdCtr || TI [|| RespData])	=	AE 21 3D D1 F2 F0 CD DB 99 C3 21 1F 96 16 9C FC
16	MACt	=	21 D1 F0 DB C3 1F 16 FC
17	Compare MACt-s (from step 10 and step 15)	=	21 D1 F0 DB C3 1F 16 FC == 21 D1 F0 DB C3 1F 16 FC

Figure 14 shows response on Cmd.ManageGPIO in non-authenticated and authenticated state. GPIO1 is configured (Cmd.SetConfig) as AccessCondition: 0x0 (only Key0x00 has access rights for the Cmd.ManageGPIO command).

- First Cmd.ManageGPIO does not succeed to toggle GPIO1, because the NFC reader did not perform mutual authentication first.
- Second Cmd.ManageGPIO is successful, because of successful mutual authentication upfront with Key 0x00 (because of AccessCondition: 0x0 for GPIO1).

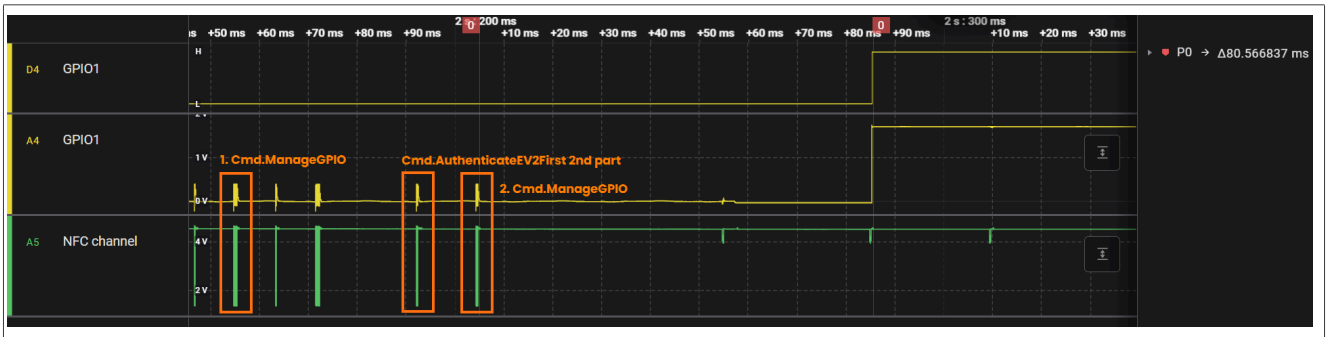

Figure 14. Cmd.ManageGPIO trace

Note: When executing a Cmd.ManageGPIO SET or MEASURE operation, or the combination of both, NTAG X DNA always returns a WTX over the NFC interface, before executing the operation. WTXes can be seen on Figure 14 as small ripples on the "NFC channel" until the GPIO1 toggle command is in process.

4.2 ReadGPIO

Cmd.ReadGPIO may be used to read the current status of the GPIOs. Status can be read of GPIOs configured either as an Output or Input.

Following example shows how an NFC reader/writer device can read the status of (all) GPIOs. Optionally, NFC reader/writer device can perform mutual authentication (SIGMA-I or AES-128/256) prior to sending the Cmd.ReadGPIO command, to create a secure NFC communication channel. Note that the CommMode of the Cmd.ReadGPIO command can be changed by Cmd.SetConfig, Option byte 0x11 (for example, can be set in a way that no authentication is required for toggling GPIOx).

Prerequisites:

1. GPIO1Conf, GPIO2Conf
2. ReadGPIOAccessCondition for Cmd.SetConfig, GPIO Management option is set to:
   - CommMode: 0x0 (communication channel is in PLAIN)
   - AccessCondition: 0xE ("free" access rights for Cmd.ReadGPIO command)

Table 2. Cmd.ReadGPIO

Step	Command		Data
1	Cmd (INS) of GetKeySettings	=	43
2	CommMode	=	CommMode.PLAIN
3	APDU	>	90 43 00 00 00
4	R-APDU	<	49 48 48 91 00
5	Response	=	49 48 48
6	GPIOByte0 (0x49 is N/A)	=	49
7	GPIOByte1 (0x48 is High)	=	48
8	GPIOByte2 (0x48 is High)	=	48

4.2.1 GPIO status is mirrored in the NDEF file (message)

The status of GPIOs can be automatically projected (acronym: mirrored within NDEF file) by NTAG X DNA automatically. This feature makes any NFC Forum device (for example cell phone) that it gets values/statuses of GPIOs with a simple tap from a Home screen (no dedicated application running needed). The whole data payload is in the form of NDEF message (for example, URL). The NFC mobile then makes a request to the verification server automatically, where data can be decrypted, checked for authenticity and integrity. The feature is called SDM (Secure Dynamic Messaging), also known as Secure Dynamic NDEF (SUN). More details can be found in ref.[5].

Below is the example of NDEF message of URI record structure and how GPIO status is mirrored into the NDEF message automatically.

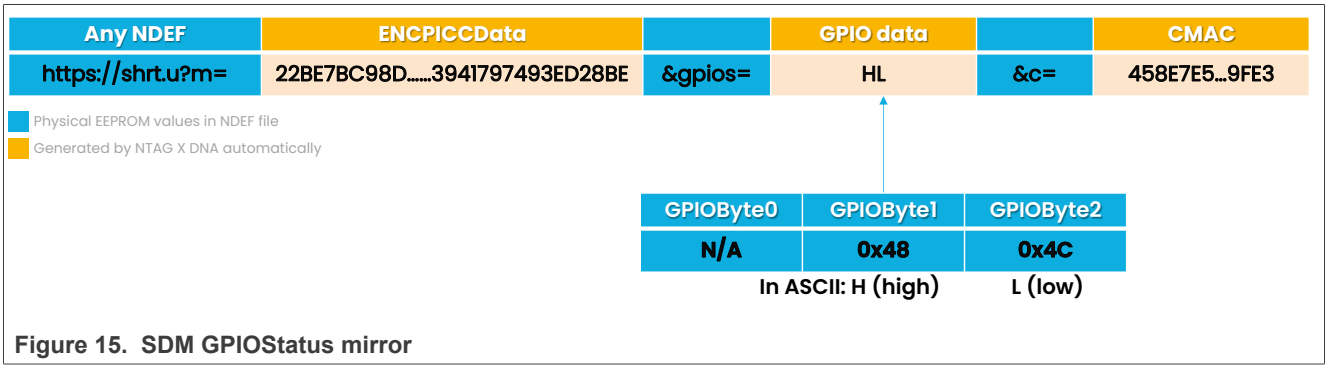

Figure 15. SDM GPIOStatus mirror

Note: *The GPIO status (mirror) can be optionally encrypted as well, using the SDMENCFileData encryption option.*

4.3 Authentication notification

GPIOs can be toggled (set, clear) automatically by the IC, when successful mutual authentication is performed.

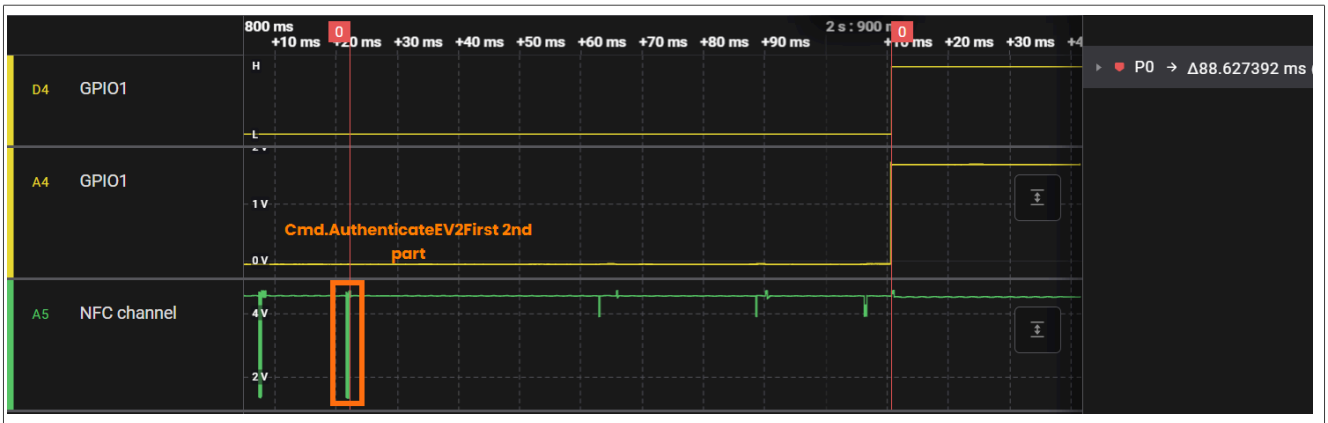

Figure 16. Successful Authentication notification

4.4 ISOSelectFile NDEF notification

GPIOs can be toggled (set, clear) automatically by IC, when the NFC host performs successful T4T NDEF Application selection (Cmd.ISOSelectFile of NDEF App - D2760000850101h). This is done by an NFC Forum compatible device automatically.

In the following example NTAG X DNA is configured as follows:

- GPIO1:
  - GPIO1Mode: 0x02 (output)
  - GPIO1Config: 0x00 (Initial state after power off cycle: LOW)
  - GPIO1Notif: 0x01 (enable authentication notification)
- GPIO2:
  - GPIO2Mode: 0x02 (output)
  - GPIO2Config: 0x00 (Initial state after power off cycle: LOW)
  - GPIO2Notif: 0x02 (enable NFC field notification)

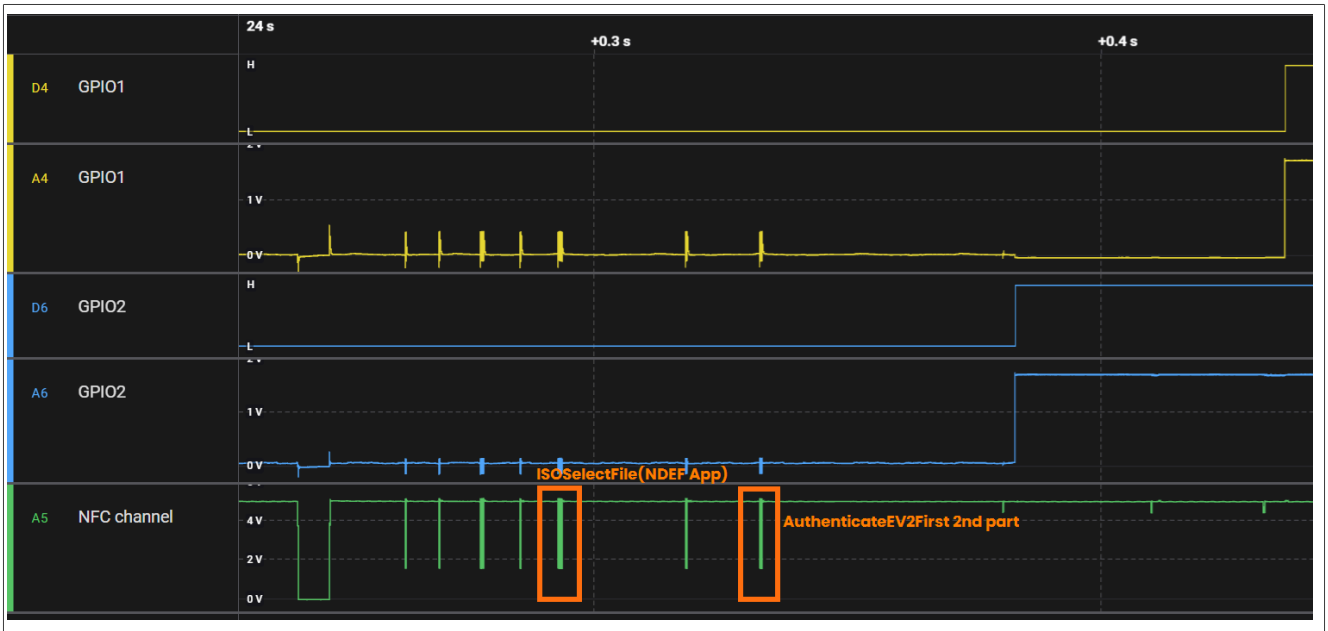

Figure 17. Cmd.ISOSelectFile and Cmd.AuthenticateEV2First notification

4.5 NFCPause examples

4.5.1 NFC pause - NFC reads data

The example of "NFCPause" communication, when NFC reads the data from NTAG X DNA, can be seen in the following flow.

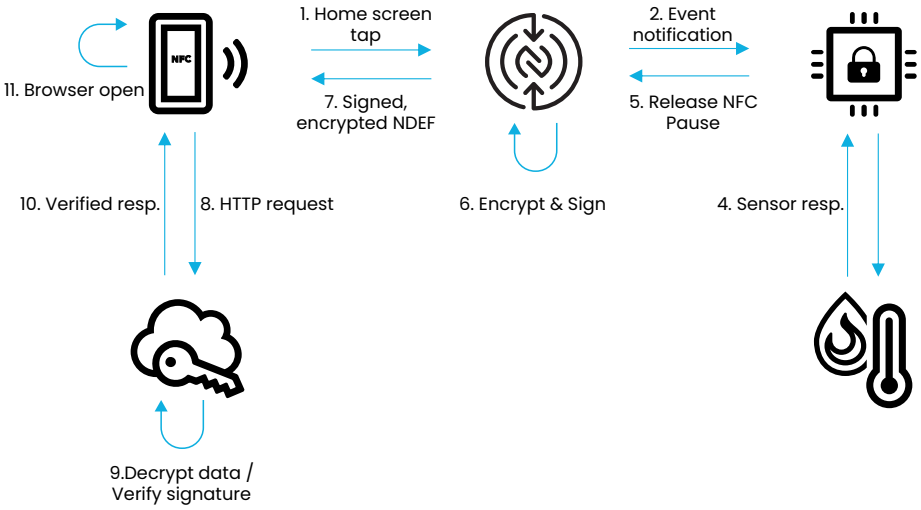

Figure 18. Example of sensor data read out - communication flow

A simple NFC tap with NFC mobile from home screen, will give the sensor value.

1. Home screen tap - tap the NFC application with NFC enabled cell phone
2. Event notification - MCU is notified about NFC device is reading NDEF File through NTAG X DNA's GPIO
3. Sensor query - MCU triggers command to fetch temp. data from P3T17 over I²C interface
4. Sensor response - sensor responds with requested data
5. MCU writes the temperature value into NTAG X DNA's NDEF file to the exact position (offset 30d or 0x1E). NDEF record used in this example is mime/text type. NDEF record can be any NFC Forum record type (e.g. URL, vCard, etc.).
   Release NFC Pause (S(WTX)) - MCU triggers ManageGPIO command. This toggles targeted GPIO on NTAG X DNA and releases WTX (NFC Pause) on NFC interface
6. Encrypt & Sign - NTAG X DNA automatically (optionally) encrypts and signs the data (NDEF)
7. Signed, encrypted NDEF is returned as a response to NFC enabled cell phone read command
8. HTTP request - NDEF, which came from the NTAG X DNA, is passed through cell's browser to the verification server automatically as HTTP request
9. Decrypt data and Verify signature - verification server can verify the date
10. Tailored Verified response - after successful verification on verification server, HTTP response can be passed
11. Browser opens - on the host cell phone to show the landing page, as shown in Figure 19

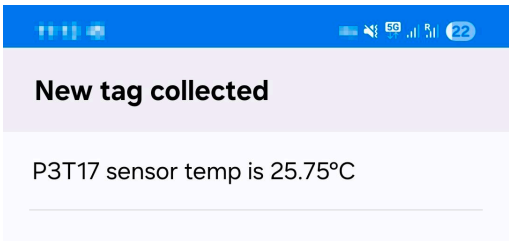

Figure 19. NFC Tap from Home screen outcome - Android screenshot

NDEF file (0xE104) memory layout of reading:

```
[00] 02 65 6E 50 33 54 31 37 20 73 65 6E 73 6F 72 20 |#enP3T17#sensor#|
[10] 74 65 6D 70 20 69 73 20 32 35 2E 38 31 C2 B0 43 |temp#is#25.81..C|
```

Detailed example sequence flow - NFC Pause on reading

Prerequisite:

1. NDEF message containing 1 record of [00 24 D1 01 20 54 02 65 6E 50 33 54 31 37 20 73 65 6E 73 6F 72 20 74 65 6D 70 20 69 73 20 32 35 2E 36 32 C2 B0 43] is programmed to NDEF File (FileNo. 0x02):
   00 24 - NDEF Length
   D1 01 20 54 02 - NDEF header
   65 6E - en
   50 33 54 31 37 20 73 65 6E 73 6F 72 20 74 65 6D 70 20 69 73 20 32 35 2E 36 32 C2 B0 43 - from HEX to ASCII: "P3T17 sensor temp is 25.62 °C"

Hint: Use NXP TagWriter application to simplify NFC programming of URL.

1. RF field is switched ON by an NFC reader/writer device and NTAG X DNA boots
2. The NFC activation (as per ref.[1]) is done by an NFC reader/writer device
3. NTAG X DNA returns its UID and SAK
4. The NFC activation (as per ref.[2]) is done by an NFC reader/writer device
5. NTAG X DNA returns ATS and PPS. Additionally, NFC reader/writer device can perform the switch to HBR
6. NFC reader/writer device Selects NDEF Application (Additionally, NFC reader/writer device can select a CC file before and reads its contents)
7. NFC reader/writer device Selects NDEF File (within selected NDEF Application)
8. NFC reader/writer device issues read command ISOReadBinary (could be ReadData) - both trigger NFCPause
9. NTAG X DNA sends WTX back to NFC reader/writer device. It will keep sending S(WTX) until released by Cmd.ManageGPIO (step 17)
10. NTAG X DNA automatically toggles its GPIO to notify MCU (e.g. interrupt) that NFC reader/writer device is done accessing NTAG X DNA flash memory (NDEF file)
11. NFC Host waits for time *n*FWT* to pass. When time will pass, NFC host will retry the communication so long as no S(WTX) will be coming from NTAG X DNA
12. MCU fetches the data from P3T17
13. P3T17 returns temperature values
14. MCU interpretates the data
15. MCU writes the temperature values into NDEF File (File No. 0x02) at offset 0xE1, length of 0x05.
16. NTAG X DNA returns ACK
17. MCU triggers Cmd.ManageGPIO to toggle the GPIO back and to stop NTAG's auto S(WTX)
18. NDEF File contents are returned as a response to step 8

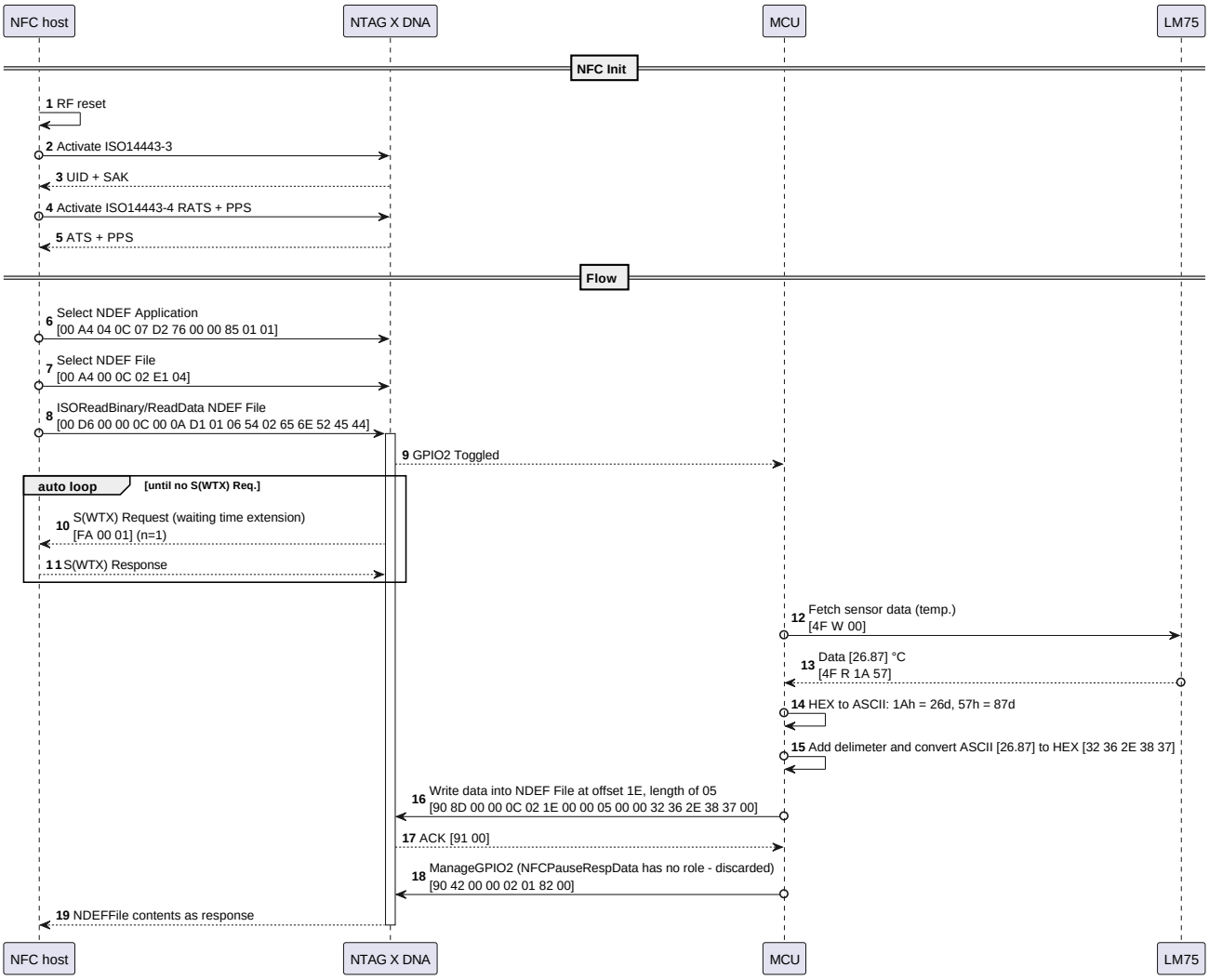

Figure 20. Example sequence diagram flow - NFC Pause on reading

4.5.2 NFC Pause - NFC read data with energy harvesting

Following example shows how NFC frontend (for example, NFC mobile) can read integrity protected, authentic, secure sensor data, with the help of NTAG X DNA and its capability of NFC field harvesting.

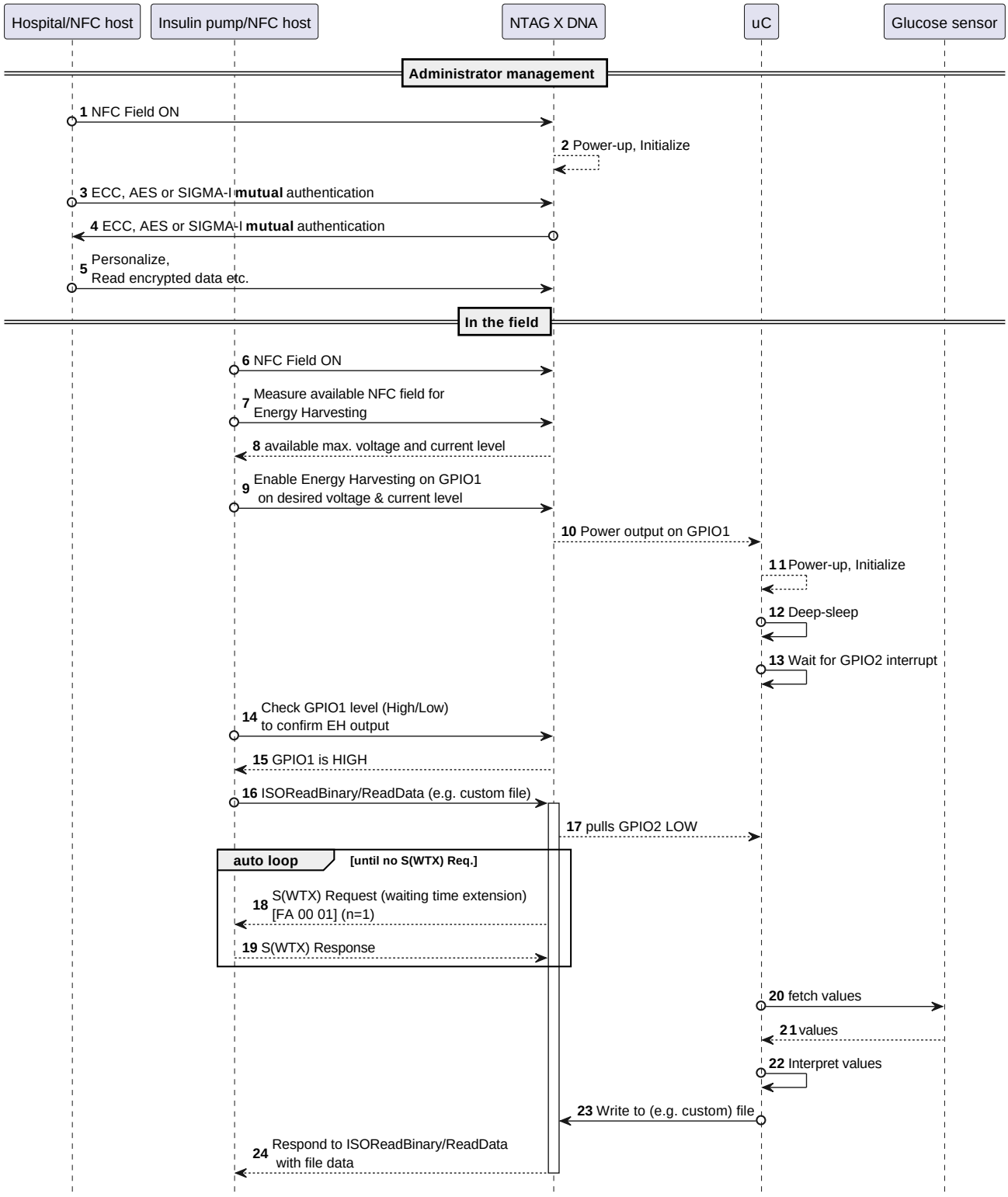

Figure 21. Example sequence diagram flow - NFC Pause on reading with energy harvesting - Body glucose meter and insulin doser application example

4.5.3 NFC pause - NFC writes data

An example of "NFCPause" communication can be seen in the following flow. NFC Host will write some data to the NTAG X DNA and that data will be read out by MCU.

Note that writing from an NFC mobile device can be achieved with a dedicated app only.

The following steps are shown in the example flow below:

1. RF field is switched ON by an NFC reader/writer device and NTAG X DNA boots
2. The NFC activation (as per ref.[1]) is done by an NFC reader/writer device
3. NTAG X DNA returns its UID and SAK
4. The NFC activation (as per ref.[2]) is done by an NFC reader/writer device
5. NTAG X DNA returns ATS and PPS. Additionally, NFC reader/writer device can perform the switch to HBR
6. NFC reader/writer device Selects NDEF Application (Additionally, NFC reader/writer device can select a CC file before and reads its contents)
7. NFC reader/writer device Selects NDEF File (within selected NDEF Application)
8. NFC reader/writer device writes to NDEF file
9. NFC reader/writer device triggers ManageGPIO command (Optionally GPIO can be toggled automatically by Cmd.ISOReadBinary, if configured)
10. NTAG X DNA sends S(WTX) back to NFC reader/writer device
11. NTAG X DNA automatically toggles its GPIO to notify MCU (for example, through interrupt) that NFC reader/writer device is done accessing NTAG X DNA flash memory (NDEF file)
12. NFC Host waits for time *n*FWT* to pass. When time will pass, NFC host will retry the communication so long as no S(WTX) will be coming from NTAG X DNA
13. MCU reads NDEF file contents (note: no need to reselect NDEF Application by I²C controller / MCU)
14. NTAG X DNA returns data requested in step 13.
15. MCU triggers Cmd.ManageGPIO to toggle the GPIO back and provide additional custom NFCPauseResp data (up to 240 Bytes) to the NFC reader/writer device. Custom NFCPauseResp data can be for example "message from MCU that it has successfuly/unsuccessfully enabled LEDs".
16. NTAG X DNA stops responding with S(WTX) automatically and provides Custom NFCPauseResp data as response to step 9
17. MCU: Interpret NDEF data
18. MCU: Trigger GPIO / drive LED matrix

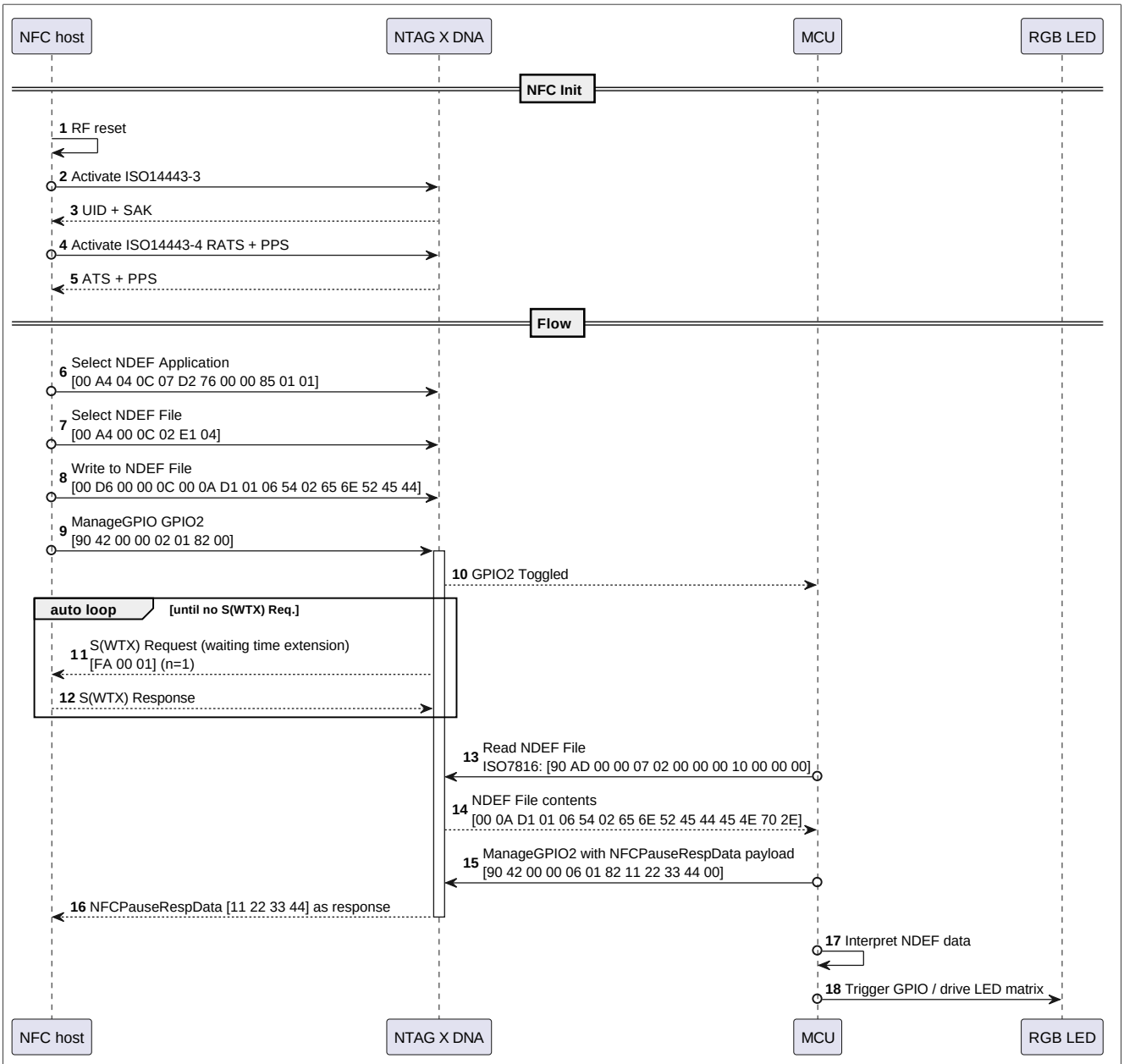

Figure 22. Example sequence/application diagram flow - NFC Pause

Note: *Additionally, mutual authentication (SIGMA-I or AES) may be freely included before step 6 , so only the authentic NFC reader/writer is allowed to continue the flow.*

5 NX Middleware examples

For the NX Middleware stack, demos, and information on how to install it, refer to https://github.com/NXP/nxmw.

In the following examples, MS Windows platform is used to run the examples. For exact details (first time) how to set up the environment, follow the Quick startup guide ref.[7] to create development environment on local PC. Extracts are included below.

5.1 Prerequisites

1. Visual studio installed (>= 2015 version, or higher)
2. Python installed (version 3.10 or above). Please follow the guidelines at Python Download
3. Git installed. Please follow the guidelines at git install
4. West installed (version 1.2.0 or above). Please follow west setup to install west
5. CMake installed (version 3.30.0 or above). Preferably at C:/opt/cmake/bin/cmake.exe
6. MCXA153 (or any other supported board) flashed with VCOM, more detailed info in ref.[7]

5.2 Getting the NX Middleware source

Use the command:

```
west init -m https://github.com/NXP/nxmw.git --mf mcu_sdk/west.yml workspace
cd workspace
west update
```

Note: *The complete setup takes 10-15 minutes to download. If a network error occurrs, you will see a message that the update failed for project. Try* `west update` *again.*

5.3 Create build files

1. Edit (text editor) `nxmw/scripts/create_cmake_projects.py` to reflect tools locations in your system
2. In command line, run:

```
cd nxmw\scripts
env_setup.bat
python create_cmake_projects.py
```

3. Build files are generated in `<nx_mw_root_folder>/nxmw_build`
4. Navigate to `<nx_mw_root_folder>/nxmw_build/se_x86`

5.4 Build example(s) - Tool for configuring NTAG X DNA

For the following demos, users need to configure the NTAG X DNA properly (for example, GPIOx to Input, GPIOx to Output, etc.) by using "nx_tool_setconfig".

To build a single example (nx_tool_setconfig - Navigate to `<nx_mw_root_folder>/nxmw_build/se_x86`) run:

```
cmake --build . --target nx_tool_setconfig
```

Optionally, all projects (demos) can be build using: `cmake --build .`

5.5 GPIO - Output/Input example

The following example demonstrates a GPIO output / input operations using NX MW APIs. It uses:

- GPIO1 as NTAG X DNA's output connected to MCU's GPIO input.
- GPIO2 as NTAG X DNA's input connected to MCU's GPIO output.

Example's description can be found under: https://github.com/NXP/nxmw/tree/main/demos/nx/gpio.

5.5.1 Prerequisites - hardware

NTAG-X-DNA-EVAL should be connected to FRDM-MCXN153 board or similar. For more options take a look into ref.[7].

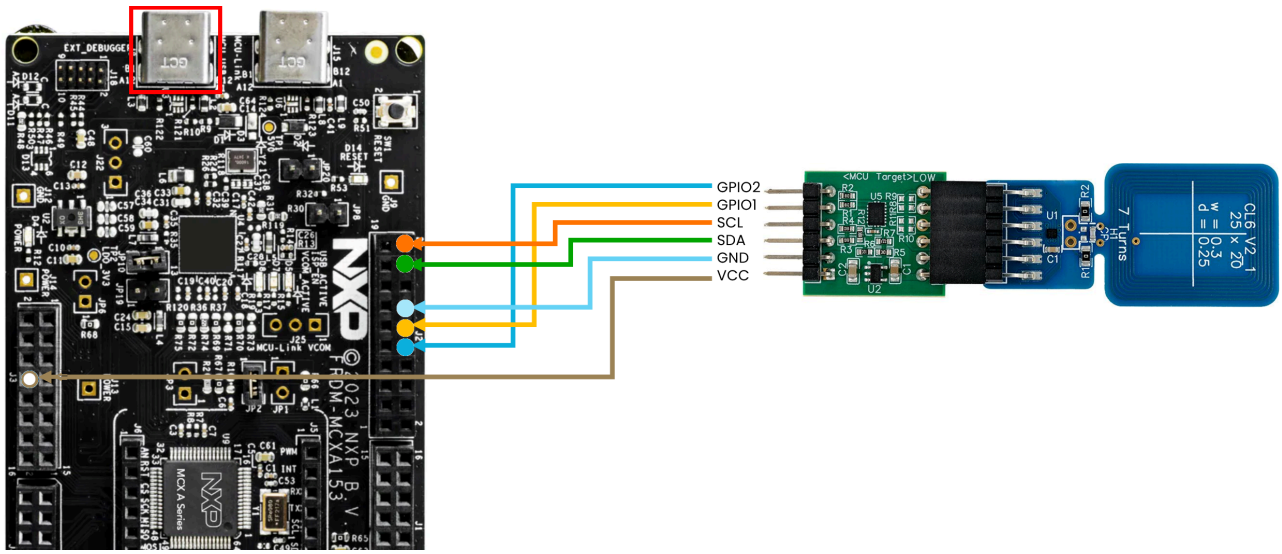

Figure 23. NTAG-X-DNA-EVAL connection to FRDM-MCXA153 development board

Table 3. Connections to FRDM-MCXA153 development board

NTAG-X-DNA-EVAL pin	FRDM-MCXA153 development board pin
GPIO2	J2 → 10 (PTD5) / P2_16
GPIO1	J2 → 12 (PTD7) / P2_12
SCL	J2 → 20 (PTE1) / P1_9
SDA	J2 → 18 (PTE0) / P1_8
GND	J2 → 14 (GND) or J3 → 12, 14 (GND)
VCC	J3 → 8 (+3V3) or 10 (+5V)

5.5.2 Prerequisites - software and NTAG X DNA configuration

NX middleware example for this demo is cross-compiled on MS Windows. Follow https://github.com/NXP/nxmw/tree/main/doc/windows to set up the environment. Build project "nx_tool_setconfig".

Plug in MCXA board into "MCU USB" (not MCU-Link) port. Check in Windows Device Manager which COM port appeared as "NxpNfcCockpit VCOM" device. Use this number in the following command prompt. For this example, COM30 (replace with the one used for the actual application).

NTAG X DNA needs to be configured using **nx_tool_setconfig** tool Section 5.4.

Config:

- GPIO1 should be configured as an output
- GPIO2 should be configured as an input

After successful building of "nx_tool_setconfig" Section 5.4, go to the Build (Debug or Release) directory and lunch following command from the command-line:

```
nx_tool_setconfig.exe -gpio1mode output -gpio2mode input -gpioMgmtCM full -
gpioReadCM full -gpioMgmtAC 0x0 -gpioReadAC 0x0 "COM30"
```

Options description:

- **gpio1mode**: Set GPIO1 to disabled, input, output, input tag tamper or energy harvesting power output
- **gpio2mode**: Set GPIO2 to disabled, input, or output
- **gpioMgmtCM**: Set ManageGPIO communication mode full (CmdMode.FULL - ManageGPIO command is executed in the secure channel - encrypted and mac-ed)
- **gpioReadCM**: Set ReadGPIO communication mode full (CmdMode.FULL - ReadGPIO command is executed in the secure channel - encrypted and mac-ed)
- **gpioMgmtAC**: Set ManageGPIO access condition 0x0 (Key 0x0 Mutually authenticated host only has access to ManageGPIO command)
- **gpioReadAC**: Set ReadGPIO access condition 0x0 (Key 0x0 Mutually authenticated host only has access to the ManageGPIO command)

Console output Figure 24 description after running above command. To decode APDUs use ref.[4].

1. request CIP
2. select NDEF Application
3. Mutual Authentication AES-128. Details can be found in ref.[5]
4. Get NTAG's configuration - Option set to GPIO Management
5. Set NTAG's configuration

Figure 24. nx_tool_setconfig - Console output 1/2

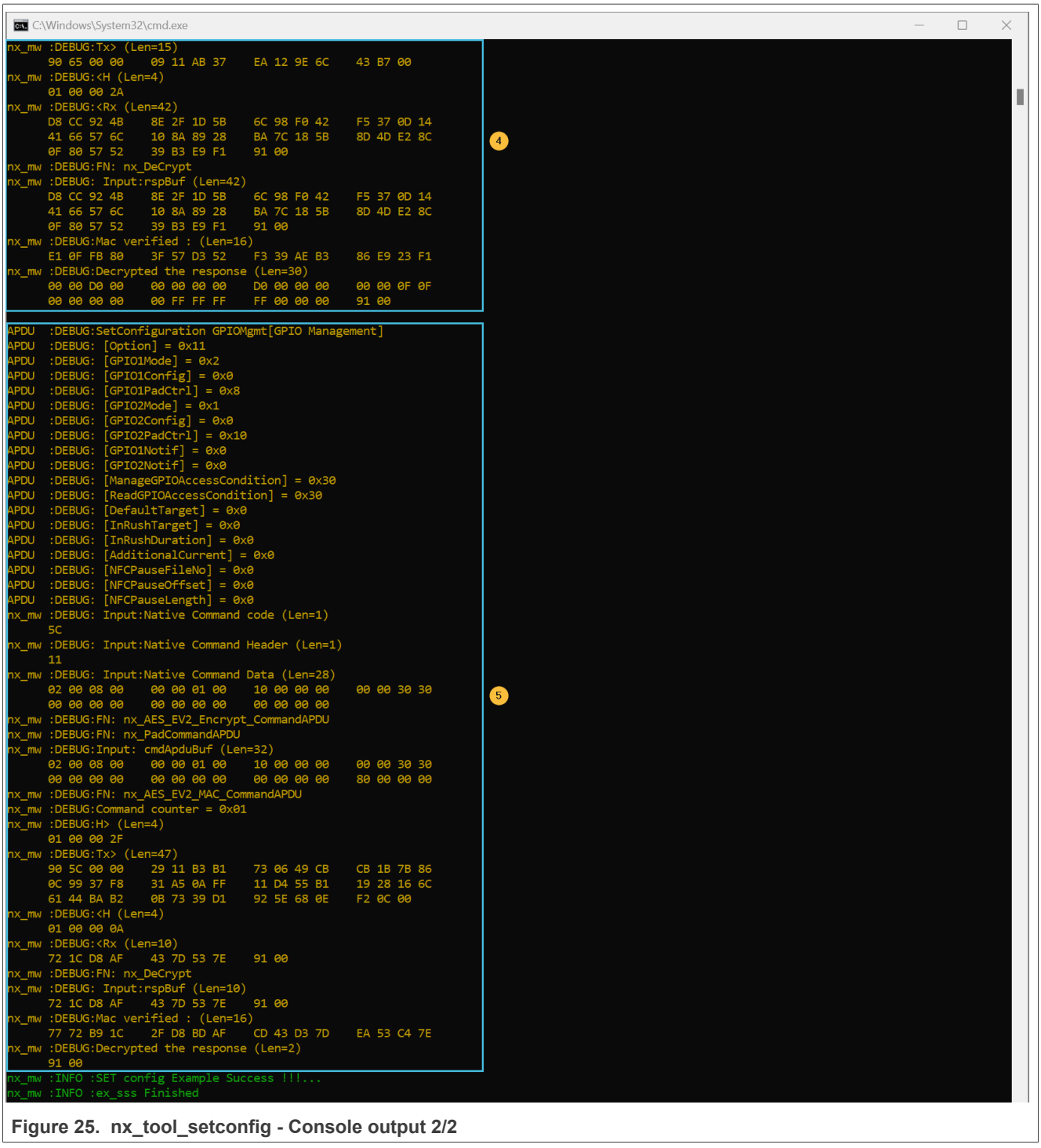

Figure 25. nx_tool_setconfig - Console output 2/2

5.5.3 Building the example - ex_gpio

To build a single example (ex_gpio) run:

```
cmake --build . --target ex_gpio
```

Build files are generated in `<nx_mw_root_folder>/nxmw_build/se_x86/bin/Debug`

5.5.4 Run/Debug the example

To run a single example (ex_gpio) run below command in command-line from `<nx_mw_root_folder>/nxmw_build/se_x86/bin/Debug`. Replace COM number with yours:

```
ex_gpio COM30
```

5.5.5 Console output

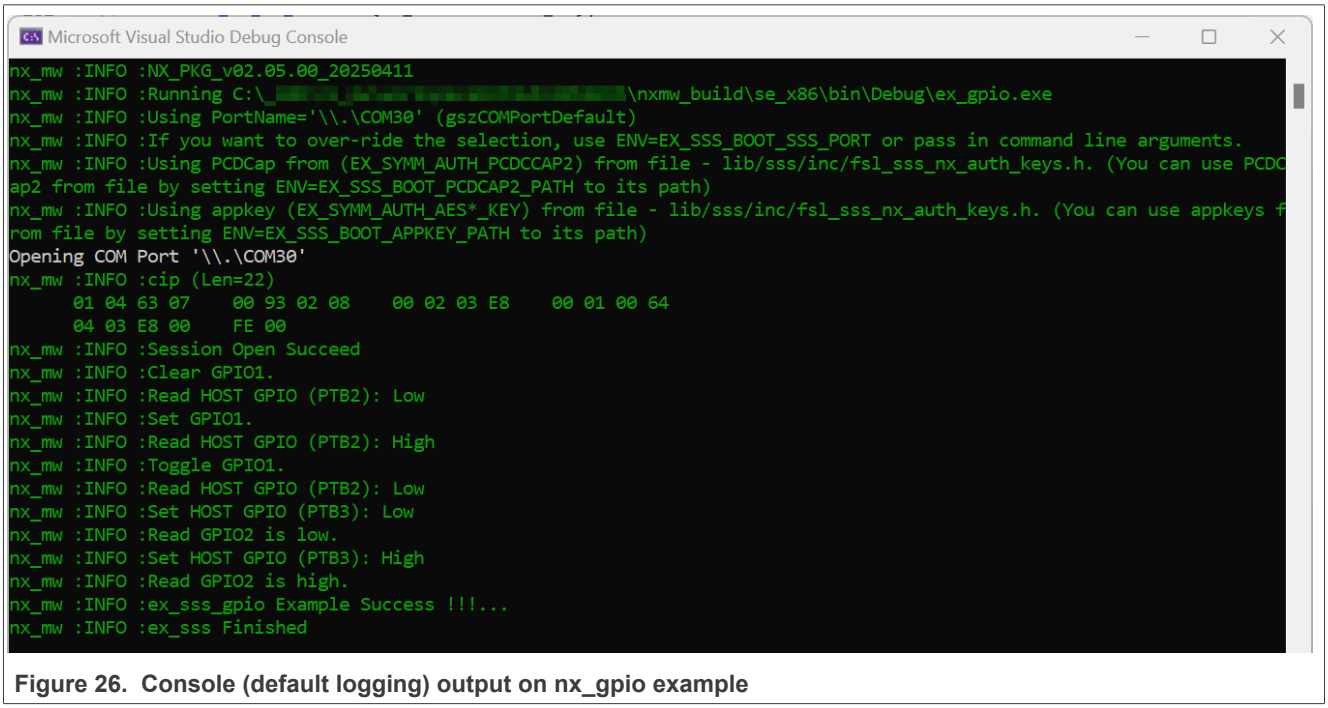

Figure 26. Console (default logging) output on nx_gpio example

Console (verbose logging) output on nx_gpio example:

- (1) Select NDEF Application
- (2) Mutual Authentication: AES-128
- (3) (optional) Get NTAG's configuration - Option set to GPIO Management - NX MW always checks access rights and access conditions before triggering ManageGPIO command
- (4) ManageGPIO - clear GPIO1
- (5) (optional) Get NTAG's configuration - Option set to GPIO Management - NX MW always checks access rights and access conditions before triggering ManageGPIO command
- (6) ManageGPIO - set GPIO1
- (7) ManageGPIO - toggle GPIO1
- (8) ReadGPIO - GPIO2
- (9) ReadGPIO - GPIO2

Shown APDUs are in ISO/IEC 7816-4 command/response pair format.

```
1 nx_mw :INFO :NX_PKG_v02.05.00_20250411
2 nx_mw :INFO :Running C:\nx_mw\workspace\nxmw_build\se_x86\bin\Debug
\ex_gpio.exe
3 nx_mw :INFO :Using PortName='\\.\COM30' (gszCOMPortDefault)
4 nx_mw :INFO :If you want to over-ride the selection, use
ENV=EX_SSS_BOOT_SSS_PORT or pass in command line arguments.
```

```
nx_mw :DEBUG:Using File: C:\nxp\configuration\plain_pcdcap2.txt
nx_mw :INFO :Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/
sss/inc/fsl_sss_nx_auth_keys.h. (You can use PCDCap2 from file by setting
 ENV=EX_SSS_BOOT_PCDCAP2_PATH to its path)
nx_mw :DEBUG:Using PCDCap from (EX_SYMM_AUTH_PCDCCAP2) from file - lib/sss/
inc/fsl_sss_nx_auth_keys.h.
nx_mw :DEBUG:Using File: C:\nxp\configuration\plain_appkey.txt
nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/
sss/inc/fsl_sss_nx_auth_keys.h. (You can use appkeys from file by setting
 ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
nx_mw :DEBUG:Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/sss/inc/
fsl_sss_nx_auth_keys.h.
Opening COM Port '\\.\COM30'
nx_mw :DEBUG:Get CIP (Len=4)
       00 00 00 00
nx_mw :DEBUG:pCip (Len=26)
       00 00 00 16    01 04 63 07    00 93 02 08    00 02 03 E8
       00 01 00 64    04 03 E8 00    FE 00
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0D
nx_mw :DEBUG:Tx> (Len=13)
       00 A4 04 0C    07 D2 76 00    00 85 01 01    00 (1)
nx_mw :DEBUG:<H (Len=4)
       01 00 00 02
nx_mw :DEBUG:<Rx (Len=2)
       90 00
nx_mw :INFO :cip (Len=22)
       01 04 63 07    00 93 02 08    00 02 03 E8    00 01 00 64
       04 03 E8 00    FE 00
APDU  :DEBUG: [KeyNo] = 0x0
APDU  :DEBUG: [pcdCap2Len] = 0x6
APDU  :DEBUG: [PCDCap2] (Len=6)
       00 00 00 00    00 00
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0E
nx_mw :DEBUG:Tx> (Len=14)
       90 71 00 00    08 00 06 00    00 00 00 00    00 00 (2)
nx_mw :DEBUG:<H (Len=4)
       01 00 00 12
nx_mw :DEBUG:<Rx (Len=18)
       66 0F 2A 9A    F0 3F 96 51    17 5B CC 24    06 46 21 B9
       91 AF
nx_mw :DEBUG:generate RndA (Len=16)
       8E F4 8F 05    2C 13 EA 78    DE 8B 51 1E    38 5B A2 8E
APDU  :DEBUG: [Enc RndA||RndBdash] (Len=32)
       B0 1D F0 E9    97 8D 69 5A    29 DB E8 5F    5D 17 70 0E
       D0 7E AC BA    B6 EA 93 71    EA DA 99 18    9E 71 5C EB
nx_mw :DEBUG:H> (Len=4)
       01 00 00 26
nx_mw :DEBUG:Tx> (Len=38)
       90 AF 00 00    20 B0 1D F0    E9 97 8D 69    5A 29 DB E8
       5F 5D 17 70    0E D0 7E AC    BA B6 EA 93    71 EA DA 99
       18 9E 71 5C    EB 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 22
nx_mw :DEBUG:<Rx (Len=34)
       54 5E D2 74    03 9D 7F 06    4F 50 45 A3    FD 17 A9 94
       5D BD 55 23    0B D7 35 82    24 D2 C3 29    07 6A 0C 1F
       91 00
nx_mw :DEBUG:FN: nx_DeCrypt
```

```
nx_mw :DEBUG: Input:rspBuf (Len=34)
      54 5E D2 74    03 9D 7F 06    4F 50 45 A3    FD 17 A9 94
      5D BD 55 23    0B D7 35 82    24 D2 C3 29    07 6A 0C 1F
      91 00
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :Clear GPIO1.

APDU  :DEBUG:GetConfiguration GPIOMgmt[GPIO Management] (3)
APDU  :DEBUG: [Option] = 0x11
nx_mw :DEBUG: Input:Native Command code (Len=1)
      65
nx_mw :DEBUG: Input:Native Command Header (Len=1)
      11
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x00
nx_mw :DEBUG:H> (Len=4)
      01 00 00 0F
nx_mw :DEBUG:Tx> (Len=15)
      90 65 00 00    09 11 CE F8    0A D3 35 EB    A6 CE 00
nx_mw :DEBUG:<H (Len=4)
      01 00 00 2A
nx_mw :DEBUG:<Rx (Len=42)
      CF 4A 8C D5    D4 76 A2 4A    90 3A EE 7D    81 F9 CC D2
      2A 8F 39 D9    79 BE 87 E0    2F CB 56 E3    9A BE DA 39
      97 8D 56 D9    4D 97 FC 7F    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=42)
      CF 4A 8C D5    D4 76 A2 4A    90 3A EE 7D    81 F9 CC D2
      2A 8F 39 D9    79 BE 87 E0    2F CB 56 E3    9A BE DA 39
      97 8D 56 D9    4D 97 FC 7F    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
      B4 97 E9 8D    3E 56 BA D9    1D 4D 5A 97    5C FC D2 7F
nx_mw :DEBUG:Decrypted the response (Len=30)
      02 00 08 00    00 00 01 00    10 00 00 00    00 00 30 30
      00 00 00 00    00 00 00 00    00 00 00 00    91 00

APDU  :DEBUG:ManageGPIO [Output] (4)
APDU  :DEBUG: [GPIONo] = 0x0
APDU  :DEBUG: [Operation] = 0x0
nx_mw :DEBUG: Input:Native Command code (Len=1)
      42
nx_mw :DEBUG: Input:Native Command Header (Len=2)
      00 00
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x01
nx_mw :DEBUG:H> (Len=4)
      01 00 00 10
nx_mw :DEBUG:Tx> (Len=16)
      90 42 00 00    0A 00 00 56    98 55 D9 74    5F 6D D2 00
nx_mw :DEBUG:<H (Len=4)
      01 00 00 0A
nx_mw :DEBUG:<Rx (Len=10)
      79 23 7E 69    2A F3 E6 AC    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=10)
      79 23 7E 69    2A F3 E6 AC    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
      8D 79 D5 23    DE 7E 41 69    EA 2A 9B F3    EF E6 11 AC
nx_mw :DEBUG:Decrypted the response (Len=2)
      91 00
```

```
nx_mw :DEBUG:H> (Len=4)
      04 00 00 02
nx_mw :DEBUG:Tx> (Len=2)
      01 00
nx_mw :DEBUG:<H (Len=4)
      04 00 00 00
nx_mw :DEBUG:<Rx (Len=0)
nx_mw :DEBUG:H> (Len=4)
      08 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
      01
nx_mw :DEBUG:<H (Len=4)
      08 00 00 01
nx_mw :DEBUG:<Rx (Len=1)
      00
nx_mw :INFO :Read HOST GPIO (PTB2): Low
nx_mw :DEBUG:resp=0
nx_mw :INFO :Set GPIO1.

APDU  :DEBUG:GetConfiguration GPIOMgmt[GPIO Management] (5)
APDU  :DEBUG: [Option] = 0x11
nx_mw :DEBUG: Input:Native Command code (Len=1)
      65
nx_mw :DEBUG: Input:Native Command Header (Len=1)
      11
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x02
nx_mw :DEBUG:H> (Len=4)
      01 00 00 0F
nx_mw :DEBUG:Tx> (Len=15)
      90 65 00 00    09 11 A7 42    43 28 A5 34    5F 78 00
nx_mw :DEBUG:<H (Len=4)
      01 00 00 2A
nx_mw :DEBUG:<Rx (Len=42)
      F4 50 3D 3F    00 C4 FA 70    4B 34 0F BF    49 17 B8 F6
      60 3A 99 94    6E 1D 8A 95    FB 62 7A 7F    B4 D5 9A 28
      D2 AA 0A 3C    5A 70 28 19    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=42)
      F4 50 3D 3F    00 C4 FA 70    4B 34 0F BF    49 17 B8 F6
      60 3A 99 94    6E 1D 8A 95    FB 62 7A 7F    B4 D5 9A 28
      D2 AA 0A 3C    5A 70 28 19    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
      CD D2 4A AA    2E 0A BE 3C    82 5A B6 70    61 28 55 19
nx_mw :DEBUG:Decrypted the response (Len=30)
      02 00 08 00    00 00 01 00    10 00 00 00    00 00 30 30
      00 00 00 00    00 00 00 00    00 00 00 00    91 00

APDU  :DEBUG:ManageGPIO [Output] (6)
APDU  :DEBUG: [GPIONo] = 0x0
APDU  :DEBUG: [Operation] = 0x1
nx_mw :DEBUG: Input:Native Command code (Len=1)
      42
nx_mw :DEBUG: Input:Native Command Header (Len=2)
      00 01
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x03
nx_mw :DEBUG:H> (Len=4)
      01 00 00 10
nx_mw :DEBUG:Tx> (Len=16)
```

```
       90 42 00 00    0A 00 01 DC    9A 3D D9 70    77 FC 15 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 0A
nx_mw :DEBUG:<Rx (Len=10)
       A6 54 F4 B2    39 7B F9 22    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=10)
       A6 54 F4 B2    39 7B F9 22    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
       B2 A6 65 54    EA F4 CB B2    FD 39 FA 7B    53 F9 80 22
nx_mw :DEBUG:Decrypted the response (Len=2)
       91 00
nx_mw :DEBUG:H> (Len=4)
       08 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
       01
nx_mw :DEBUG:<H (Len=4)
       08 00 00 01
nx_mw :DEBUG:<Rx (Len=1)
       01
nx_mw :INFO :Read HOST GPIO (PTB2): High
nx_mw :DEBUG:resp=1
nx_mw :INFO :Toggle GPIO1.

APDU  :DEBUG:GetConfiguration GPIOMgmt[GPIO Management]
APDU  :DEBUG: [Option] = 0x11
nx_mw :DEBUG: Input:Native Command code (Len=1)
       65
nx_mw :DEBUG: Input:Native Command Header (Len=1)
       11
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x04
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0F
nx_mw :DEBUG:Tx> (Len=15)
       90 65 00 00    09 11 EB 0D    2F 8F E7 A6    2B 8A 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 2A
nx_mw :DEBUG:<Rx (Len=42)
       5C B0 EA 8D    3C 41 B0 C2    F8 5C 7E 08    94 97 62 7E
       AC 75 12 8F    69 F9 01 61    E0 EB 5B 74    AE B9 F0 40
       FF 16 5E D6    5A EC A2 D3    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=42)
       5C B0 EA 8D    3C 41 B0 C2    F8 5C 7E 08    94 97 62 7E
       AC 75 12 8F    69 F9 01 61    E0 EB 5B 74    AE B9 F0 40
       FF 16 5E D6    5A EC A2 D3    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
       12 FF E7 16    04 5E 72 D6    F3 5A 3E EC    C8 A2 B6 D3
nx_mw :DEBUG:Decrypted the response (Len=30)
       02 00 08 00    00 00 01 00    10 00 00 00    00 00 30 30
       00 00 00 00    00 00 00 00    00 00 00 00    91 00

APDU  :DEBUG:ManageGPIO [Output] (7)
APDU  :DEBUG: [GPIONo] = 0x0
APDU  :DEBUG: [Operation] = 0x2
nx_mw :DEBUG: Input:Native Command code (Len=1)
       42
nx_mw :DEBUG: Input:Native Command Header (Len=2)
       00 02
```

```
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x05
nx_mw :DEBUG:H> (Len=4)
       01 00 00 10
nx_mw :DEBUG:Tx> (Len=16)
       90 42 00 00    0A 00 02 32    FF 7D E7 E5    9C BE A6 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 0A
nx_mw :DEBUG:<Rx (Len=10)
       71 96 E3 E3    A0 EF C3 A1    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=10)
       71 96 E3 E3    A0 EF C3 A1    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
       05 71 CB 96    00 E3 FE E3    A7 A0 F1 EF    CD C3 2A A1
nx_mw :DEBUG:Decrypted the response (Len=2)
       91 00
nx_mw :DEBUG:H> (Len=4)
       08 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
       01
nx_mw :DEBUG:<H (Len=4)
       08 00 00 01
nx_mw :DEBUG:<Rx (Len=1)
       00
nx_mw :INFO :Read HOST GPIO (PTB2): Low
nx_mw :DEBUG:resp=0
nx_mw :INFO :Set HOST GPIO (PTB3): Low
nx_mw :DEBUG:H> (Len=4)
       04 00 00 02
nx_mw :DEBUG:Tx> (Len=2)
       02 01
nx_mw :DEBUG:<H (Len=4)
       04 00 00 00
nx_mw :DEBUG:<Rx (Len=0)
nx_mw :DEBUG:H> (Len=4)
       06 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
       02
nx_mw :DEBUG:<H (Len=4)
       06 00 00 00
nx_mw :DEBUG:<Rx (Len=0)

APDU  :DEBUG:GetConfiguration GPIOMgmt[GPIO Management]
APDU  :DEBUG: [Option] = 0x11
nx_mw :DEBUG: Input:Native Command code (Len=1)
       65
nx_mw :DEBUG: Input:Native Command Header (Len=1)
       11
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x06
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0F
nx_mw :DEBUG:Tx> (Len=15)
       90 65 00 00    09 11 C3 80    CD EE A9 0F    C8 09 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 2A
nx_mw :DEBUG:<Rx (Len=42)
       C7 55 2B C8    3C B4 2D 67    BF A7 77 60    AE C6 E7 29
       5D 19 05 DB    68 1D 6F 9E    88 C9 65 D1    A4 3C 53 DF
```

```
       3C 52 57 D9    8B 80 6F 68    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=42)
       C7 55 2B C8    3C B4 2D 67    BF A7 77 60    AE C6 E7 29
       5D 19 05 DB    68 1D 6F 9E    88 C9 65 D1    A4 3C 53 DF
       3C 52 57 D9    8B 80 6F 68    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
       0B 3C D5 52    E8 57 1A D9    E3 8B 42 80    86 6F 68 68
nx_mw :DEBUG:Decrypted the response (Len=30)
       02 00 08 00    00 00 01 00    10 00 00 00    00 00 30 30
       00 00 00 00    00 00 00 00    00 00 00 00    91 00

APDU  :DEBUG:ReadGPIO [] (8)
nx_mw :DEBUG: Input:Native Command code (Len=1)
       43
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x07
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0E
nx_mw :DEBUG:Tx> (Len=14)
       90 43 00 00    08 90 AD 96    59 54 5A D4    19 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 1A
nx_mw :DEBUG:<Rx (Len=26)
       BF 13 CC F4    20 31 00 33    54 49 CC AF    0D 1E FF F7
       A8 DC 5A 3B    A1 57 FC 63    91 00
nx_mw :DEBUG:FN: nx_DeCrypt
nx_mw :DEBUG: Input:rspBuf (Len=26)
       BF 13 CC F4    20 31 00 33    54 49 CC AF    0D 1E FF F7
       A8 DC 5A 3B    A1 57 FC 63    91 00
nx_mw :DEBUG:Mac verified : (Len=16)
       EB A8 E5 DC    DA 5A 30 3B    2C A1 D9 57    F9 FC A1 63
nx_mw :DEBUG:Decrypted the response (Len=5)
       49 4C 4C 91    00
nx_mw :INFO :Read GPIO2 is low.
nx_mw :INFO :Set HOST GPIO (PTB3): High
nx_mw :DEBUG:H> (Len=4)
       05 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
       02
nx_mw :DEBUG:<H (Len=4)
       05 00 00 00
nx_mw :DEBUG:<Rx (Len=0)

APDU  :DEBUG:GetConfiguration GPIOMgmt[GPIO Management]
APDU  :DEBUG: [Option] = 0x11
nx_mw :DEBUG: Input:Native Command code (Len=1)
       65
nx_mw :DEBUG: Input:Native Command Header (Len=1)
       11
nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
nx_mw :DEBUG:Command counter = 0x08
nx_mw :DEBUG:H> (Len=4)
       01 00 00 0F
nx_mw :DEBUG:Tx> (Len=15)
       90 65 00 00    09 11 F7 C3    58 F9 D2 CB    99 08 00
nx_mw :DEBUG:<H (Len=4)
       01 00 00 2A
nx_mw :DEBUG:<Rx (Len=42)
       38 F4 C8 59    5F 0E 31 EE    32 49 49 4D    5C E0 74 1E
```

```
359         00 92 A3 29    27 09 19 DD    E9 8C B7 3C    B5 EB AC 1F
360         D9 90 9D 8F    C7 C2 C0 6B    91 00
361 nx_mw :DEBUG:FN: nx_DeCrypt
362 nx_mw :DEBUG: Input:rspBuf (Len=42)
363         38 F4 C8 59    5F 0E 31 EE    32 49 49 4D    5C E0 74 1E
364         00 92 A3 29    27 09 19 DD    E9 8C B7 3C    B5 EB AC 1F
365         D9 90 9D 8F    C7 C2 C0 6B    91 00
366 nx_mw :DEBUG:Mac verified : (Len=16)
367         85 D9 5E 90    C0 9D F5 8F    6D C7 5A C2    4E C0 3C 6B
368 nx_mw :DEBUG:Decrypted the response (Len=30)
369         02 00 08 00    00 00 01 00    10 00 00 00    00 00 30 30
370         00 00 00 00    00 00 00 00    00 00 00 00    91 00
371
372 APDU  :DEBUG:ReadGPIO [] (9)
373 nx_mw :DEBUG: Input:Native Command code (Len=1)
374         43
375 nx_mw :DEBUG:FN: nx_AES_EV2_MAC_CommandAPDU
376 nx_mw :DEBUG:Command counter = 0x09
377 nx_mw :DEBUG:H> (Len=4)
378         01 00 00 0E
379 nx_mw :DEBUG:Tx> (Len=14)
380         90 43 00 00    08 57 33 74    0B 78 2B ED    9B 00
381 nx_mw :DEBUG:<H (Len=4)
382         01 00 00 1A
383 nx_mw :DEBUG:<Rx (Len=26)
384         16 20 01 79    F0 D9 C0 9D    9E 31 D1 CE    07 A7 D8 E7
385         F6 E8 B8 25    18 A9 49 88    91 00
386 nx_mw :DEBUG:FN: nx_DeCrypt
387 nx_mw :DEBUG: Input:rspBuf (Len=26)
388         16 20 01 79    F0 D9 C0 9D    9E 31 D1 CE    07 A7 D8 E7
389         F6 E8 B8 25    18 A9 49 88    91 00
390 nx_mw :DEBUG:Mac verified : (Len=16)
391         7F F6 06 E8    7D B8 68 25    1A 18 7C A9    B9 49 FE 88
392 nx_mw :DEBUG:Decrypted the response (Len=5)
393         49 4C 48 91    00
394 nx_mw :INFO :Read GPIO2 is high.
395 nx_mw :INFO :ex_sss_gpio Example Success !!!...
396 nx_mw :INFO :ex_sss Finished
```

5.5.6 Logic trace

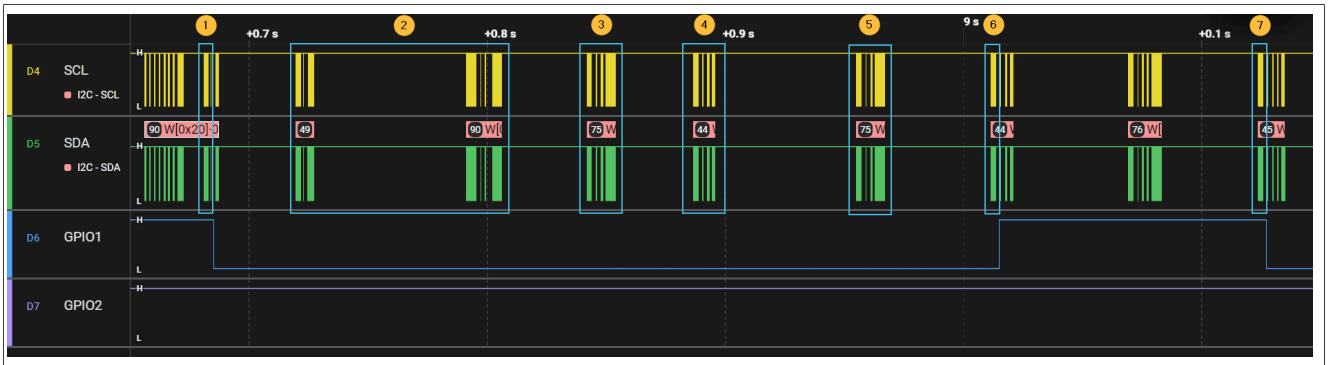

Figure 27. nx_gpio Logic trace 1 - graphics

- (1) Select NDEF Application - after this command, GPIOs are pushed/pulled regarding GPIOxConfig - Bit 0: Initial state after power-off cycle

- (2) Mutual Authentication: AES-128
- (3) (optional) Get NTAG's configuration - Option set to GPIO Management - NX MW always checks access rights and access conditions before triggering ManageGPIO command
- (4) ManageGPIO - clear GPIO1 - after "clear" state GPIOx is set to low. In our case default state of GPIO1 is low, therefore GPIO1 remains low after this command.
- (5) (optional) Get NTAG's configuration - Option set to GPIO Management - NX MW always checks access rights and access conditions before triggering ManageGPIO command
- (6) ManageGPIO - set GPIO1 - GPIO1 is pushed HIGH
- (7) ManageGPIO - toggle GPIO1 - GPIO1 is toggled

Shown APDUs are in T=1 over I²C format. More details in ref.[3].

Prologue Field (mandatory)			Information Field (optional)	Epilogue Field (mandatory)
NAD (1 byte)	PCB (1 byte)	LEN (2 byte)	INF (LEN bytes)	CRC (2 bytes)

Figure 28. T=1 over I²C block format

```
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
read to 0x20 nak
write to 0x20 ack data: 0x21 0xC0 0x00 0x00 0x65 0xAC
read to 0x20 ack data: 0x12 0xE0
read to 0x20 nak
read to 0x20 ack data: 0x00 0x00
read to 0x20 nak
read to 0x20 ack data: 0x0F 0xA8
write to 0x20 ack data: 0x21 0xC4 0x00 0x00 0x06 0xCD
read to 0x20 ack data: 0x12 0xE4
read to 0x20 nak
read to 0x20 ack data: 0x00 0x16
read to 0x20 nak
read to 0x20 ack data: 0x01 0x04 0x63 0x07 0x00 0x93 0x02 0x08 0x00 0x02 0x03
 0xE8 0x00 0x01 0x00 0x64 0x04 0x03 0xE8 0x00 0xFE 0x00 0xDC 0x8E
write to 0x20 ack data: 0x21 0x00 0x00 0x0D 0x00 0xA4 0x04 0x0C 0x07 0xD2 0x76
 0x00 0x00 0x85 0x01 0x01 0x00 0xF2 0x5D (1)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x02
read to 0x20 ack data: 0x90 0x00 0x11 0x8C
write to 0x20 ack data: 0x21 0x40 0x00 0x0E 0x90 0x71 0x00 0x00 0x08 0x00 0x06
 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x50 0xB3 (2)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x40
read to 0x20 ack data: 0x00 0x12
read to 0x20 ack data: 0x96 0x65 0xB9 0x10 0x0A 0x57 0x99 0x33 0x0B 0xF1 0xBD
 0xB0 0xB5 0x23 0x59 0x0C 0x91 0xAF 0xA7 0x43
write to 0x20 ack data: 0x21 0x00 0x00 0x26 0x90 0xAF 0x00 0x00 0x20 0xDD 0xB5
 0x2C 0xC3 0xF7 0x48 0xF2 0x7F 0x60 0xF9 0x1E 0x65 0x8F 0xCE 0xBE 0xC5 0xA2 0x4A
 0x98 0x5D 0x36 0xE0 0x20 0x13 0xC7 0xC4 0xA7 0x53 0x1D 0x6E 0x55 0x8A 0x00 0x3D
 0x06
```

```
read to 0x20 nak
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x22
read to 0x20 ack data: 0x57 0xE6 0xD7 0xB9 0xA5 0xC1 0x34 0xEE 0xBF 0x6E 0x06
 0x36 0x80 0xF4 0x66 0xB0 0x99 0x75 0x81 0x71 0x17 0xAC 0x35 0x40 0xAC 0xFF 0x5C
 0xCF 0x55 0x1E 0x2D 0x49 0x91 0x00 0x87 0x11
write to 0x20 ack data: 0x21 0x40 0x00 0x0F 0x90 0x65 0x00 0x00 0x09 0x11 0x70
 0x25 0x66 0x6B 0x89 0x68 0x1D 0xBE 0x00 0x37 0xA7 (3)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x40
read to 0x20 ack data: 0x00 0x2A
read to 0x20 nak
read to 0x20 ack data: 0x6B 0x63 0x19 0x7A 0x7F 0x5C 0xFC 0x76 0x26 0x2A 0xEF
 0xDF 0x50 0xB0 0x0A 0x2F 0x61 0x86 0xCD 0xA6 0x92 0x56 0x0F 0x4A 0x55 0xA2 0xCD
 0x92 0x3F 0xE6 0xA3 0x49 0x72 0x28 0x8F 0x72 0x31 0x6A 0x7A 0x42 0x91 0x00 0x9E
 0xE1
write to 0x20 ack data: 0x21 0x00 0x00 0x10 0x90 0x42 0x00 0x00 0x0A 0x00 0x00
 0x31 0x11 0x5E 0x35 0x3D 0xE4 0xE3 0x0D 0x00 0x00 0x30 (4)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x0A
read to 0x20 nak
read to 0x20 ack data: 0x51 0x4D 0x89 0x31 0xCA 0xA3 0x64 0x90 0x91 0x00 0x44
 0xA1
write to 0x20 ack data: 0x21 0x40 0x00 0x0F 0x90 0x65 0x00 0x00 0x09 0x11 0x70
 0x3E 0x56 0x57 0x6E 0x51 0xA3 0x07 0x00 0x32 0x59 (5)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x40
read to 0x20 ack data: 0x00 0x2A
read to 0x20 nak
read to 0x20 ack data: 0xB9 0xD9 0xC9 0xBE 0xBD 0xA4 0xF2 0xF6 0x8E 0xFA 0xAD
 0x22 0xB3 0x25 0xED 0x0A 0x75 0x40 0x8D 0xEC 0x2F 0xCB 0x80 0x49 0x38 0x98 0x92
 0xDF 0x4B 0xF6 0x1D 0xD6 0xAC 0x4F 0x5C 0x65 0x68 0x73 0xC9 0xF2 0x91 0x00 0x0B
 0x67
write to 0x20 ack data: 0x21 0x00 0x00 0x10 0x90 0x42 0x00 0x00 0x0A 0x00 0x01
 0xFE 0x29 0xCA 0x6A 0x0F 0xD7 0xFD 0x1B 0x00 0x86 0x5C (6)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x0A
read to 0x20 nak
read to 0x20 ack data: 0x30 0x60 0x59 0x07 0x22 0x2F 0xAF 0x3B 0x91 0x00 0x97
 0xDC
write to 0x20 ack data: 0x21 0x40 0x00 0x0F 0x90 0x65 0x00 0x00 0x09 0x11 0x78
 0x1E 0xDB 0xD8 0x7D 0x6B 0x21 0x8C 0x00 0x83 0xFA
read to 0x20 nak
read to 0x20 ack data: 0x12 0x40
read to 0x20 nak
read to 0x20 ack data: 0x00 0x2A
read to 0x20 nak
read to 0x20 ack data: 0x73 0x24 0x94 0xDF 0x68 0xEF 0x24 0xAE 0xD2 0x00 0x2F
 0x37 0xE9 0xE8 0xC2 0xA1 0x82 0x3A 0x88 0x0F 0xF3 0x9E 0xA0 0x3F 0x49 0xBE 0x98
 0x57 0x35 0xEA 0x57 0xDB 0x51 0xA0 0xD1 0x04 0x19 0xC7 0x17 0xB6 0x91 0x00 0x58
 0xF5
write to 0x20 ack data: 0x21 0x00 0x00 0x10 0x90 0x42 0x00 0x00 0x0A 0x00 0x02
 0xD9 0x51 0x3A 0x3B 0xBE 0x89 0x79 0xCB 0x00 0x84 0x12 (7)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 nak
read to 0x20 ack data: 0x00 0x0A
```

```
read to 0x20 nak
read to 0x20 ack data: 0x18 0xEA 0x75 0x36 0x5C 0x07 0xEC 0xC3 0x91 0x00 0x09
 0x34
write to 0x20 ack data: 0x21 0x40 0x00 0x0F 0x90 0x65 0x00 0x00 0x09 0x11 0xF4
 0x03 0x8D 0xCB 0xF0 0xD4 0x0E 0xB7 0x00 0xBC 0xDA
read to 0x20 nak
read to 0x20 ack data: 0x12 0x40
read to 0x20 nak
read to 0x20 ack data: 0x00 0x2A
read to 0x20 nak
read to 0x20 ack data: 0xE3 0x9E 0x86 0xDD 0xCA 0xBF 0x4C 0x2B 0x40 0x48 0xC8
 0x54 0x27 0x1A 0x81 0x4F 0x2E 0xFB 0xE1 0x63 0xC0 0xF9 0x28 0x15 0x39 0x3B 0xDB
 0x81 0xA3 0xA6 0xC4 0x86 0x61 0x17 0x9B 0x68 0xDD 0x66 0x83 0x4F 0x91 0x00 0x56
 0x02
```

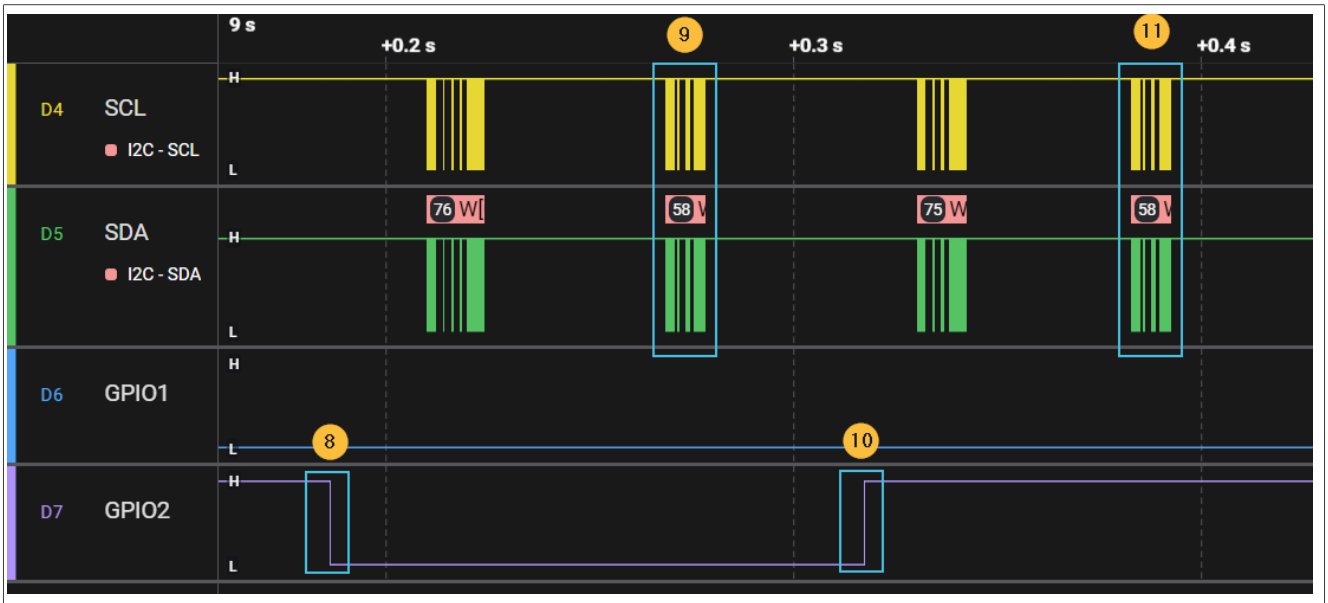

Figure 29. nx_gpio Logic trace 2 - graphics

- (8) MCU pulls GPIO2 low
- (9) ReadGPIO - GPIO2 - status, which is returned is "low" (decrypted response: 49h (Invalid) 4Ch (Low) 4Ch (Low))
- (10) MCU releases GPIO2 high
- (11) ReadGPIO - GPIO2 - status, which is returned is "high" (decrypted response: 49h (Invalid) 4Ch (Low) 48h (High))

```
write to 0x20 ack data: 0x21 0x00 0x00 0x0E 0x90 0x43 0x00 0x00 0x08 0xD3 0x06
 0x3B 0xD8 0xE3 0x59 0x9A 0xBD 0x00 0x32 0x23 (9)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x1A
read to 0x20 nak
read to 0x20 ack data: 0x2D 0x52 0xD4 0x24 0x6B 0xCE 0x10 0x17 0x32 0xF9 0x94
 0x5F 0xEE 0xA9 0x64 0x5A 0x8C 0x8A 0x58 0x43 0x5C 0x1D 0x6F 0x64 0x91 0x00 0x5D
 0xB6
write to 0x20 ack data: 0x21 0x40 0x00 0x0F 0x90 0x65 0x00 0x00 0x09 0x11 0xB5
 0x04 0x2A 0x2A 0x4C 0x3E 0x72 0x36 0x00 0xDE 0x0B
read to 0x20 nak
```

```
read to 0x20 ack data: 0x12 0x40
read to 0x20 ack data: 0x00 0x2A
read to 0x20 nak
read to 0x20 ack data: 0xE4 0x2F 0x6C 0xFA 0x22 0xEB 0x64 0x53 0x81 0x8B 0x63
 0x1C 0x36 0x01 0x3D 0x50 0xB5 0x80 0x56 0x27 0x13 0x76 0xB0 0x05 0x13 0xB7 0x26
 0xF8 0x5D 0xAE 0x39 0x0E 0x84 0x99 0x83 0x67 0x5E 0x07 0xF4 0xB4 0x91 0x00 0xB0
 0x5B
write to 0x20 ack data: 0x21 0x00 0x00 0x0E 0x90 0x43 0x00 0x00 0x08 0xDA 0xEA
 0xB1 0xF3 0x77 0xEA 0x1D 0xB4 0x00 0xE4 0x3F (11)
read to 0x20 nak
read to 0x20 ack data: 0x12 0x00
read to 0x20 ack data: 0x00 0x1A
read to 0x20 nak
read to 0x20 ack data: 0xEF 0x34 0x61 0x76 0xA7 0x09 0x6B 0xC8 0x2E 0x84 0x41
 0xDE 0xF0 0xB6 0x97 0xD5 0x8E 0xF2 0xD7 0xC6 0x8F 0x47 0xBA 0xC9 0x91 0x00 0xB8
 0xBF
```

5.6 GPIO - Notification (Authentication) example

The following example demonstrates a GPIO output, serving as successful authentication indicator. Setup is like:

- GPIO1 as NTAG X DNA's output connected to MCU's GPIO input and configured to toggle, when successful mutual authentication is done.

Example's description can be found under: https://github.com/NXP/nxmw/tree/main/demos/nx/gpio_notif.

5.6.1 Prerequisites - hardware

Same set-up as in Section 5.5.1.

5.6.2 Prerequisites - software and NTAG X DNA configuration

NTAG X DNA needs to be configured. For this, the **nx_tool_setconfig** tool can be used. See Section 5.4.

For the configuration:

- Certificates need to be provisioned.
- GPIO1 should be configured as an output.

This demo shows AES-128 (optionally SIGMA-I) Mutual authentication. After a successful authentication between host (NX MW can be build for contactless or contact) and NTAG X DNA, wired host is notified about the successful authentication through toggled GPIO1. In case of SIGMA-I, certificates need to be provisioned to the NTAG X DNA. Hint: Use nx_Personalization demo.

Cmake build options need to be enabled:

```
cmake -DNXMW_All_Auth_Code:string=Enabled .
cmake -DNXMW_Auth=SYMM_Auth -DNXMW_Secure_Tunneling=NTAG_AES128_EV2 .
```

After successful building of "nx_tool_setconfig" Section 5.4, go to Build (Debug or Release) directory and lunch following command from the command-line:

```
nx_tool_setconfig.exe -gpio1mode output -gpio1Notif auth -gpioMgmtCM full -
gpioReadCM full -gpioMgmtAC 0x0 -gpioReadAC 0x0 "COM30"
```

What NXMW_All_Auth_Code means:

- Supports **both Sigma-I and Symmetric authentication** at **runtime**.
- When built with `ALL_AUTH_ENABLE` CMake option:
  - No need to recompile for switching between Sigma-I or Symmetric.
  - The tool dynamically supports both modes.
    ALL_AUTH_DISABLED
    Only one mode (Sigma-I or Symmetric) is compiled in.
  - Switching requires changing CMake options and rebuilding.

Options description:

- **gpio1mode**: Set GPIO1 to disabled, input, output, input tag tamper or energy harvesting power output
- **gpio1Notif**: Set GPIO1 notification (toggling on external/NFC event) to disabled, enable authentication notification or enable Cmd.ISOSelectFile of NDEF Application notification
- **gpioMgmtCM**: Set ManageGPIO communication mode full (CmdMode.FULL - ManageGPIO command is executed in the secure channel - encrypted and mac-ed)
- **gpioReadCM**: Set ReadGPIO communication mode full (CmdMode.FULL - ReadGPIO command is executed in the secure channel - encrypted and mac-ed)
- **gpioMgmtAC**: Set ManageGPIO access condition 0x0 (Key 0x0 Mutually authenticated host only has access to ManageGPIO command)
- **gpioReadAC**: Set ReadGPIO access condition 0x0 (Key 0x0 Mutually authenticated host only has access to ManageGPIO command)

5.6.3 Building the example - ex_gpio_notif

To build a single example (ex_gpio_notif) run:

```
cmake --build . --target ex_gpio_notif
```

5.6.4 Run/Debug the example

To run a single example (ex_gpio_notif) run below command in command-line from `<nx_mw_root_folder>/nxmw_build/se_x86/bin/Debug`. Replace COM number with yours:

```
ex_gpio_notif COM30
```

5.6.5 Console output

```
 1 nx_mw :INFO :NX_PKG_v02.05.01_20250825
 2 nx_mw :INFO :Running ex_gpio_notif
 3 nx_mw :INFO :Using PortName='COM30' (CLI)
 4 nx_mw :DEBUG:Using File: C:\nxp\configuration\plain_appkey.txt
 5 nx_mw :INFO :Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/
sss/inc/fsl_sss_nx_auth_keys.h. (You can use appkeys from file by setting
 ENV=EX_SSS_BOOT_APPKEY_PATH to its path)
 6 nx_mw :DEBUG:Using appkey (EX_SYMM_AUTH_AES*_KEY) from file - lib/sss/inc/
fsl_sss_nx_auth_keys.h.
 7 Opening COM Port '\\.\COM30'
 8 nx_mw :DEBUG:Get CIP (Len=4)
 9       00 00 00 00
10 nx_mw :DEBUG:pCip (Len=26)
11       00 00 00 16    01 04 63 07    00 93 02 08    00 02 03 E8
```

```
12          00 01 00 64    04 03 E8 00    FE 00
13    nx_mw :DEBUG:H> (Len=4)
14          01 00 00 0D
15    nx_mw :DEBUG:Tx> (Len=13)
16          00 A4 04 0C    07 D2 76 00    00 85 01 01    00 (1)
17    nx_mw :DEBUG:<H (Len=4)
18          01 00 00 02
19    nx_mw :DEBUG:<Rx (Len=2)
20          90 00
21    nx_mw :INFO :cip (Len=22)
22          01 04 63 07    00 93 02 08    00 02 03 E8    00 01 00 64
23          04 03 E8 00    FE 00
24    APDU  :DEBUG: [KeyNo] = 0x0
25    APDU  :DEBUG: [pcdCap2Len] = 0x0
26    nx_mw :DEBUG:H> (Len=4)
27          01 00 00 08
28    nx_mw :DEBUG:Tx> (Len=8)
29          90 71 00 00    02 00 00 00 (2)
30    nx_mw :DEBUG:<H (Len=4)
31          01 00 00 12
32    nx_mw :DEBUG:<Rx (Len=18)
33          67 89 2B 0B    30 C9 A1 52    C7 74 22 F1    D1 4D CC 8E
34          91 AF
35    nx_mw :DEBUG:generate RndA (Len=16)
36          DA 53 7B D8    B0 D9 F5 0A    CC 55 1B 61    D1 22 C2 4F
37    APDU  :DEBUG: [Enc RndA||RndBdash] (Len=32)
38          4C 6E D8 50    10 6D A4 46    1D 06 15 75    CC C3 F1 08
39          DD E6 B9 9E    FA 0D 26 AF    EE 4F 17 C8    3F 3A D3 95
40    nx_mw :DEBUG:H> (Len=4)
41          01 00 00 26
42    nx_mw :DEBUG:Tx> (Len=38)
43          90 AF 00 00    20 4C 6E D8    50 10 6D A4    46 1D 06 15 (3)
44          75 CC C3 F1    08 DD E6 B9    9E FA 0D 26    AF EE 4F 17
45          C8 3F 3A D3    95 00
46    nx_mw :DEBUG:<H (Len=4)
47          01 00 00 22
48    nx_mw :DEBUG:<Rx (Len=34)
49          76 B8 B3 43    BA 38 2D C1    F3 33 28 5E    4A 4E F2 B8
50          E4 DD F1 D2    69 52 94 D4    70 AD E9 8F    88 D9 B4 B5
51          91 00
52    nx_mw :DEBUG:FN: nx_DeCrypt
53    nx_mw :DEBUG: Input:rspBuf (Len=34)
54          76 B8 B3 43    BA 38 2D C1    F3 33 28 5E    4A 4E F2 B8
55          E4 DD F1 D2    69 52 94 D4    70 AD E9 8F    88 D9 B4 B5
56          91 00
57    nx_mw :INFO :Session Open Succeed
58    nx_mw :INFO :Successfully opened AES (EV2) session
59    nx_mw :DEBUG:H> (Len=4)
60          04 00 00 02
61    nx_mw :DEBUG:Tx> (Len=2)
62          01 00
63    nx_mw :DEBUG:<H (Len=4)
64          04 00 00 00
65    nx_mw :DEBUG:<Rx (Len=0)
66    nx_mw :DEBUG:H> (Len=4)
67          08 00 00 01
68    nx_mw :DEBUG:Tx> (Len=1)
69          01
70    nx_mw :DEBUG:<H (Len=4)
71          08 00 00 01
```

```
nx_mw :DEBUG:<Rx (Len=1)
      01
nx_mw :INFO :Read GPIONotif: High
Opening COM Port '\\.\COM30'

 Already  COM Port Open
 nx_mw :DEBUG:Get CIP (Len=4) (4)
      00 00 00 00
nx_mw :DEBUG:pCip (Len=26)
      00 00 00 16    01 04 63 07    00 93 02 08    00 02 03 E8
      00 01 00 64    04 03 E8 00    FE 00
nx_mw :DEBUG:H> (Len=4)
      01 00 00 0D
nx_mw :DEBUG:Tx> (Len=13)
      00 A4 04 0C    07 D2 76 00    00 85 01 01    00 (5)
nx_mw :DEBUG:<H (Len=4)
      01 00 00 02
nx_mw :DEBUG:<Rx (Len=2)
      90 00
nx_mw :INFO :cip (Len=22)
      01 04 63 07    00 93 02 08    00 02 03 E8    00 01 00 64
      04 03 E8 00    FE 00
nx_mw :WARN :Communication channel is Plain.
nx_mw :WARN :!!!Security and privacy must be assessed.!!!
nx_mw :INFO :Session Open Succeed
nx_mw :INFO :Successfully opened Plain-session
nx_mw :DEBUG:H> (Len=4)
      08 00 00 01
nx_mw :DEBUG:Tx> (Len=1)
      01
nx_mw :DEBUG:<H (Len=4)
      08 00 00 01
nx_mw :DEBUG:<Rx (Len=1)
      00
nx_mw :INFO :Read GPIONotif: Low
nx_mw :INFO :ex_sss_gpio_notif Example Success !!!...
nx_mw :INFO :ex_sss Finished
```

5.6.6 Logic trace

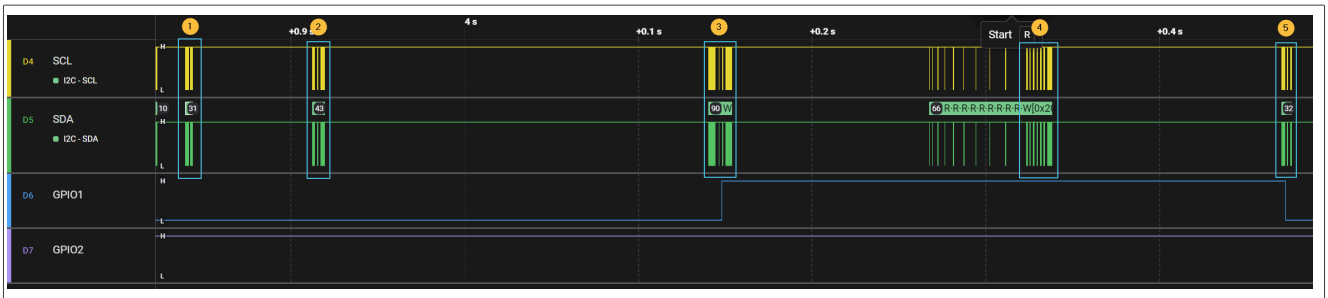

Figure 30. nx_gpio_notif Logic trace - graphics

1. (1) Select NDEF Application (Cmd.ISOSelectFile) - after this command, GPIOs are pushed/pulled regarding GPIOxConfig - Bit 0: Initial state after power-off cycle
2. (2) Mutual Authentication: AES-128 request (Cmd.AuthenticateEV2First - part1)

3. (3) Mutual Authentication: AES-128 request (Cmd.AuthenticateEV2First - part2) - when NTAG X DNA computes session keys and knows that oposite party is authentic, GPIO1 is toggled automatically
4. (4) Resync and CIP requests (S-blocks)
5. (5) Select NDEF Application (Cmd.ISOSelectFile) - after this command, GPIO1 goes to initial state

5.7 Dual interface - NFCPause example

The following example demonstrates a Dual Interface (NFC and I²C) without energy harvesting:

- GPIO2 as NTAG X DNA's output, is connected to MCU's GPIO input and configured to toggle:
  - on ISOReadBinary command - read of any byte of NDEF File (FileNo.: 0x02) between NFCPauseOffset ↔ (NFCPauseOffset + NFCPauseLength)
  - on ManageGPIO command

The description of the example can be found under: https://github.com/NXP/nxmw/tree/main/demos/nx/dual_interfaces.

5.7.1 Prerequisites - hardware

5.7.1.1 NFC Reader

To have full control over NFC interface and to go step by step for better understanding, the Pegoda (see https://www.nxp.com/design/design-center/development-boards-and-designs/CLRD730) or any PCSC compatible reader can be used.

Figure 31. NXP Pegoda 3

5.7.1.2 MCXN dev. board

NTAG-X-DNA-EVAL should be connected to FRDM-MCXN947 board or similar. For more options take a look into ref.[7].

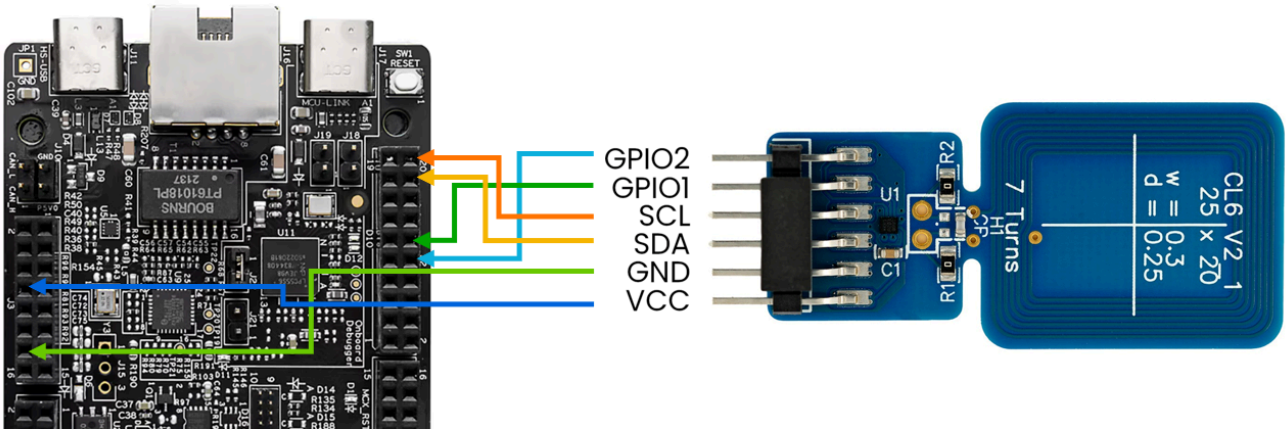

Figure 32. NTAG-X-DNA-EVAL connection to FRDM-MCXN947 development board

Table 4. Connections to FRDM-MCXN947 development board

NTAG-X-DNA-EVAL pin	FRDM-MCXA153 development board pin
GPIO2	J2 → 10 - P0_26
GPIO1	J2 → 12 - P0_25
SCL	J2 → 20 - P4_1
SDA	J2 → 18 - P4_0
GND	J2 → 14 (GND) or J3 → 10, 12 (GND)
VCC	J3 → 6 (+3V3) or 8 (+5V)

5.7.2 Prerequisites - software and NTAG X DNA configuration

Install latest RFIDDiscover and MCUXpresso IDE.

NTAG X DNA needs to be configured.

1. Use either I²C interface tools:
   a. `nx_tool_setconfig` (built for VCOM)
   b. demo example "`ex_set_config`" (requires editing)
2. Use the NFC interface tools:
   a. RFIDDiscover
   b. `nx_tool_setconfig` (built for PCSC)

NTAG X DNA Configuration for this demo shall look like:

- GPIO2 mode is configured as an "output with NFCPause"
- GPIO2 initial state is set to HIGH
- ManageGPIO CommMode is PLAIN, AccessRights are 0x0E

5.7.2.1 Configuration with nx_tool_setconfig

See nx_tool_setconfig https://github.com/NXP/nxmw/blob/main/demos/nx/nx_tool_setconfig/readme.md tool or Section 5.4

5.7.2.2 Configuration with RFIDDiscover

Following instructions show how to configure NTAG X DNA with RFIDDiscover ref.[8]. NXP Pegoda 3 HF reader is used and NTAG-X-DNA-EVAL is put on it.

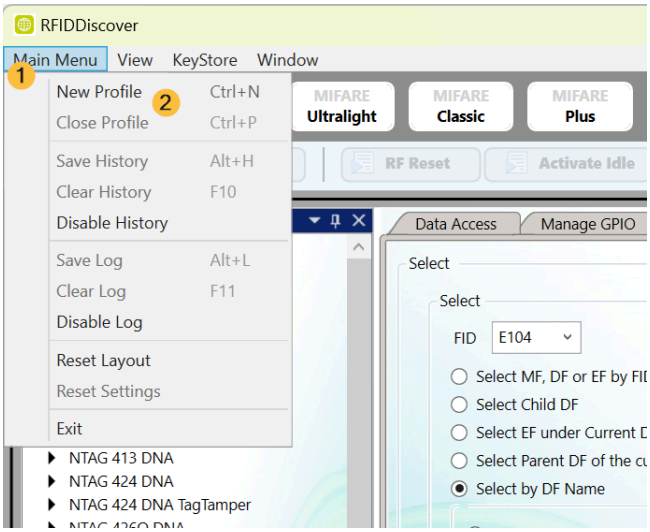

Figure 33. Config. for Dual Interface Demo with RFIDDiscover 1

1. Go to "Main Menu"
2. Select "New Profile"

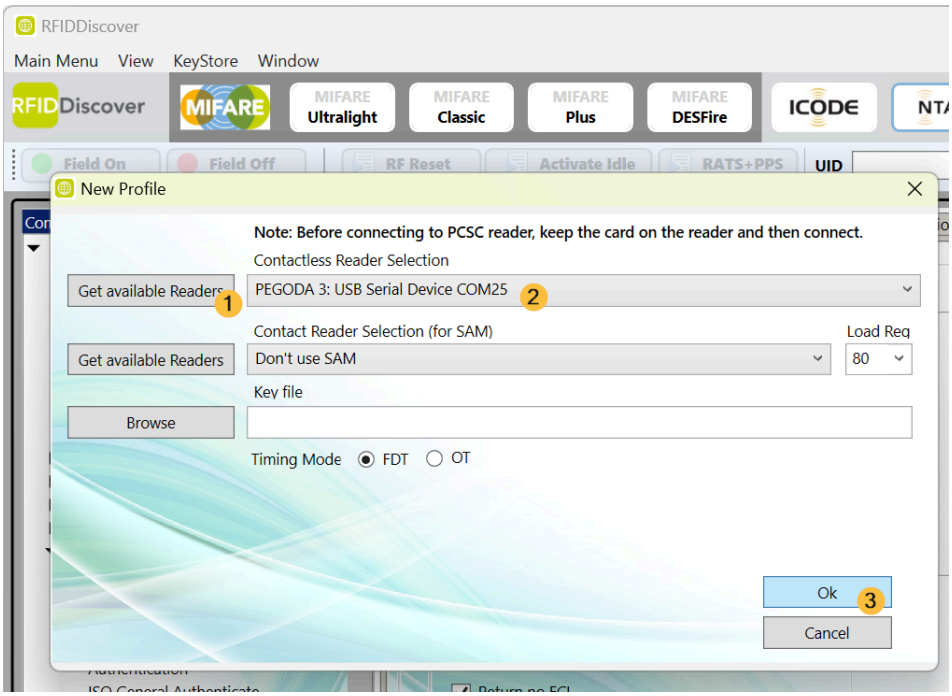

Figure 34. Config. for Dual Interface Demo with RFIDDiscover 2

1. Press "Get available Readers"
2. Choose Pegoda 3 (or PC/SC reader) from dropdown

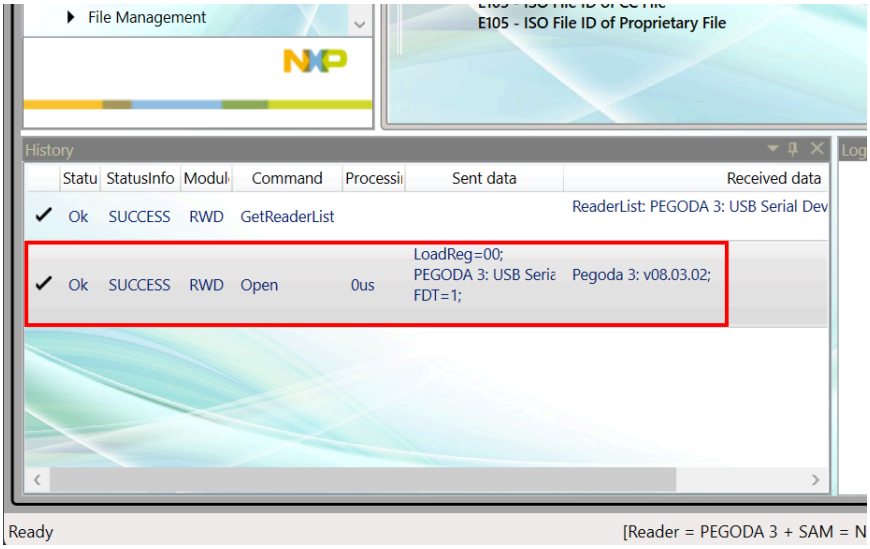

Figure 35. Config. for Dual Interface Demo with RFIDDiscover 3

1. Check if the connection to the reader was successfully done.

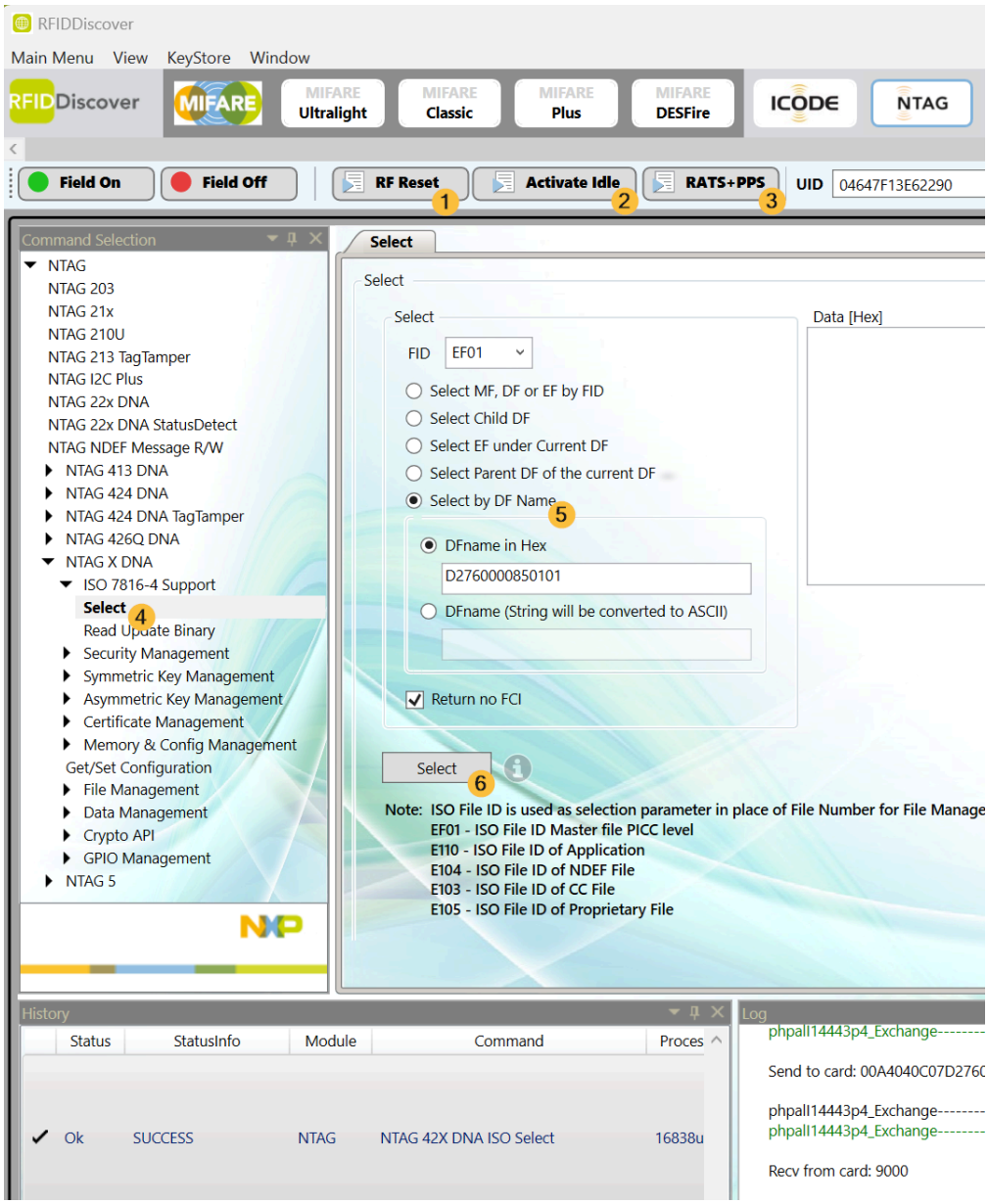

Figure 36. Config. for Dual Interface Demo with RFIDDiscover 4

1. Press "RF Reset", to turn RF field OFF and ON again
2. Press "Activate Idle" to perform ISO1444-3 activation
3. Press "RATS + PPS" to perform ISO1444-4 activation
4. Choose "Select" sub-menu under NTAG → NTAG X DNA → ISO 7816-4 Support
5. Choose "Select by DF name" (enter NDEF Application's DF name, if not predefined yet)
6. Press "Select". At this point, NDEF Application is selected.

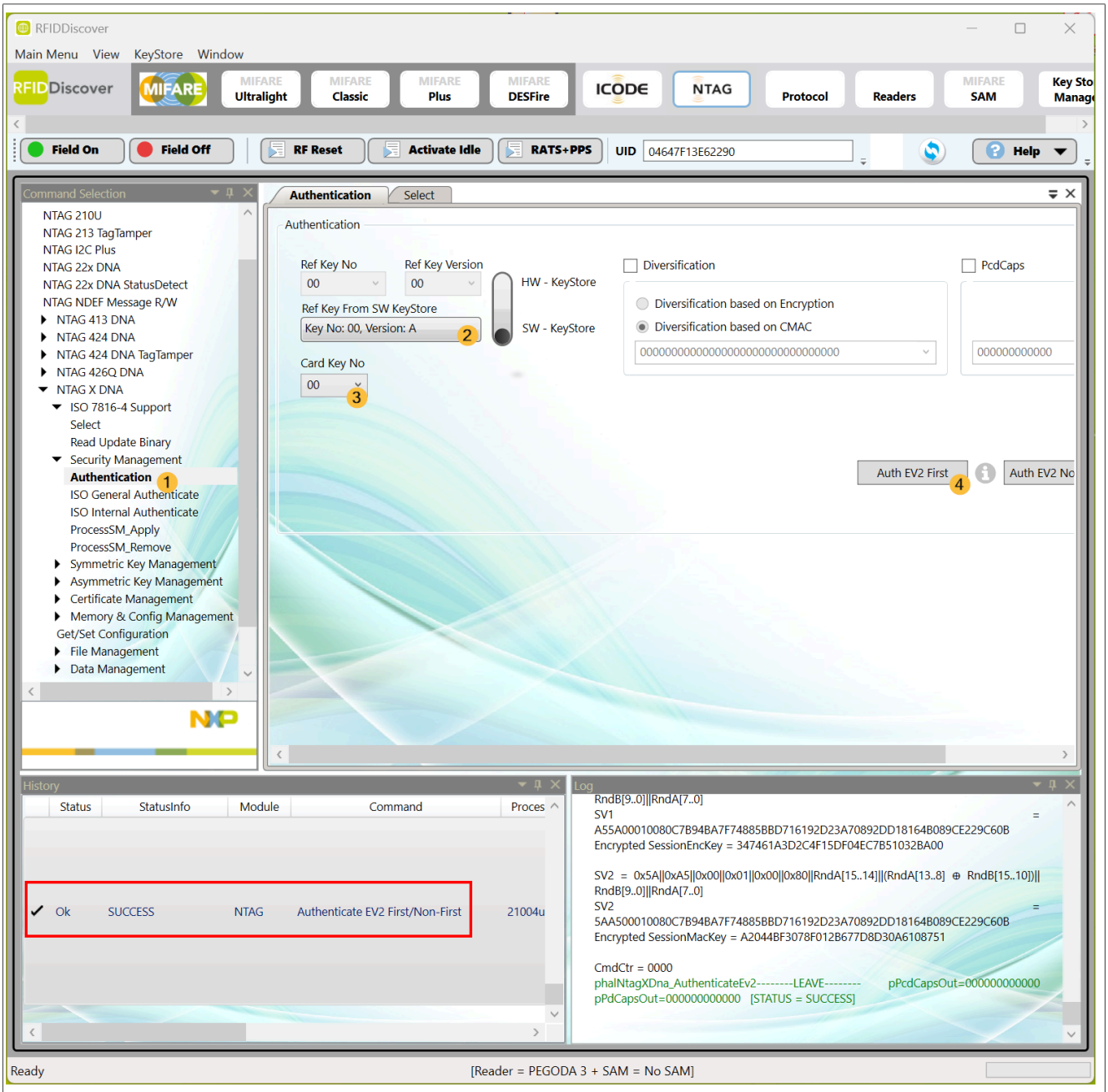

Figure 37. Config. for Dual Interface Demo with RFIDDiscover 5

1. Choose "Authentication", as Cmd.SetConfig, which will follow, requires active Mutual Authentication and CommMode.Full

2. Choose a reference key from a SW Key Store within RFIDDiscover. Note: Create a key entry in the SW Key Store if not done so yet. Use the targeted key-type in Key Store entry (e.g. AES-128), default keys values are all 00-oes.
3. Choose, which key is targeted on the NTAG X DNA
4. Press "Authenticate EV2 First"

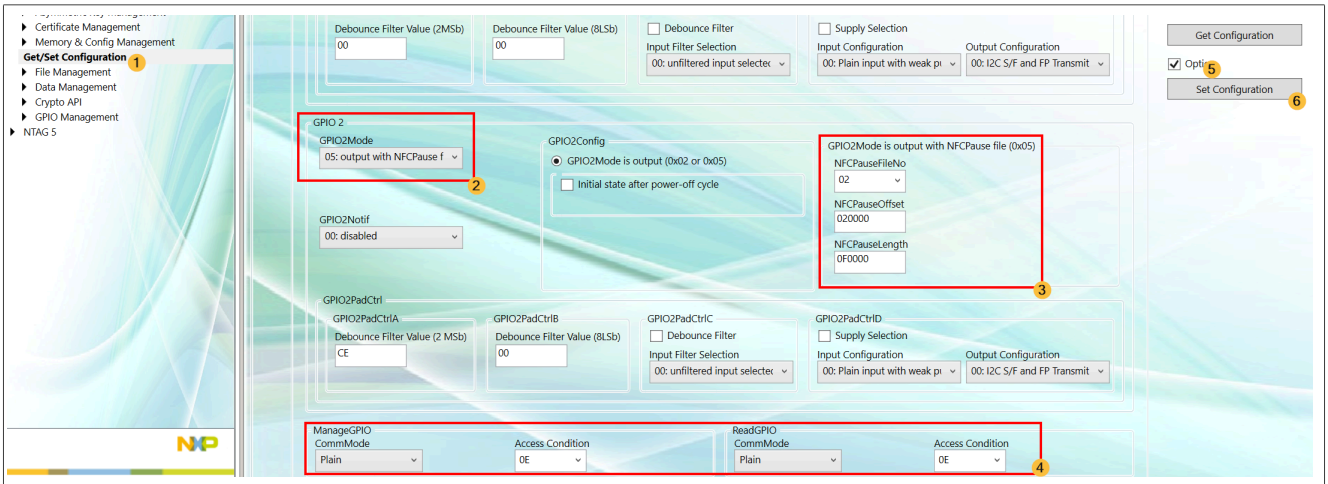

Figure 38. Config. for Dual Interface Demo with RFIDDiscover 6

1. Choose "Get/Set Configuration"
2. Select "05: output with NFCPause feature
3. Choose NFCPauseFileNo (targeted file number) 0x02, NFCPauseOffset of: 020000 (LSB first), NFCPauseLength (0F0000). Setting means, that every read (Cmd.ISOReadBinary or Cmd.ReadData) targeting NFCPauseFileNo file and reading between NFCPauseOffset and NFCPauseOffset + NFCPauseLength, will automatically toggle GPIO2
4. Set CommModes and Access Conditions to be in Plain and Free (for easier demo)

5.7.3 Building the example - dual_interfaces

Follow instructions from below link, to import `ex_dual_interfaces` example into MCUXpresso IDE:

https://github.com/NXP/nxmw/tree/main/mcux_project

Output from MCU application is useful for debugging purpose. There are two ways to get Serial Output of the application:

1. Built-in (MCUXPresso) Terminal
   a. Select the Project
   b. Press "Debug"
   c. After building process, automatic breakpoint is set at entry into main() function. Program is paused there.
   d. Press Terminal icon
   e. Check a COM port number in Windows Device Manager. Search for MCU_link VCOM Port
   f. Choose correct Serial port number
   g. Press OK to connect

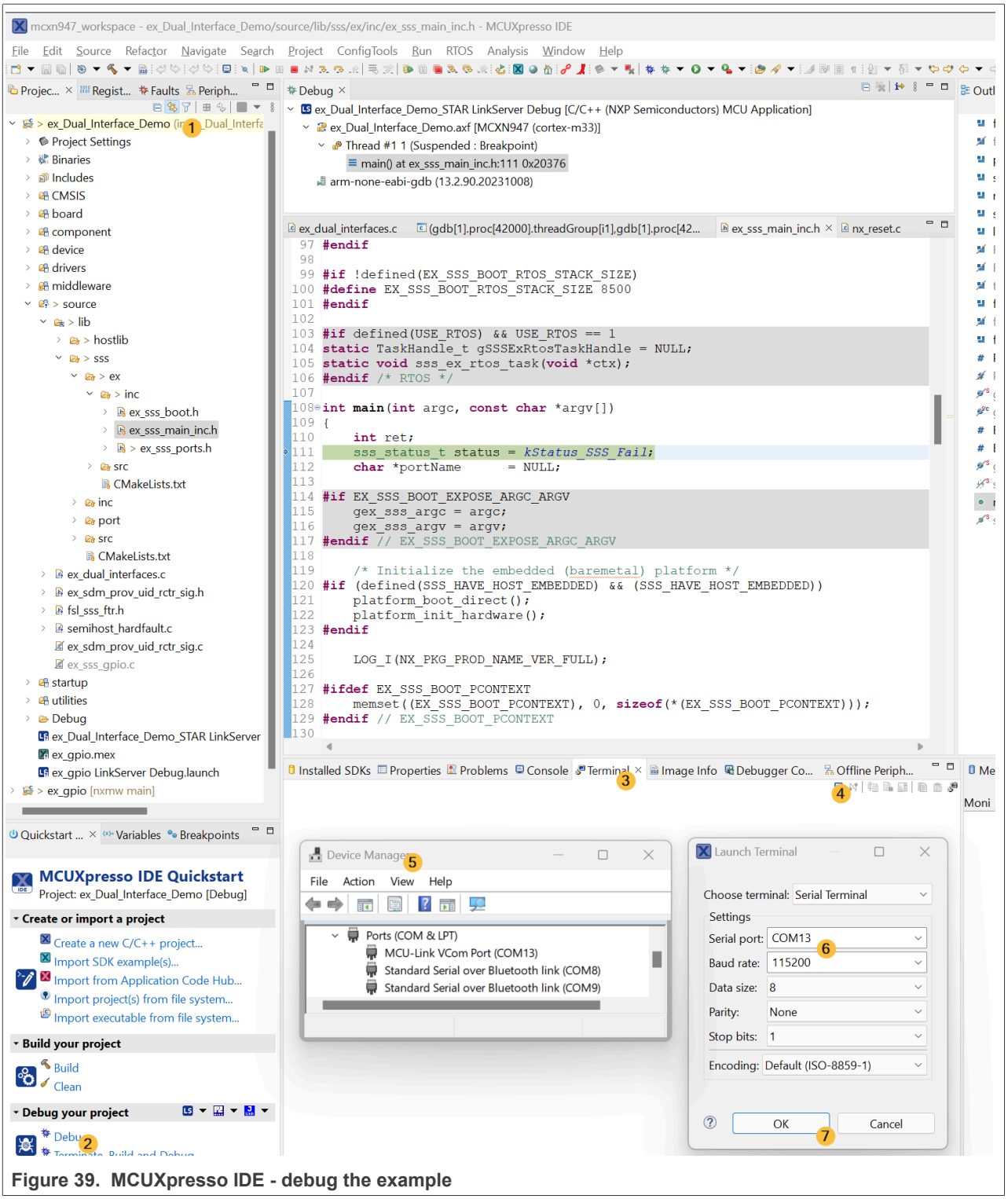

Figure 39. MCUXpresso IDE - debug the example

2. Any stand-alone Serial Terminal software (MobaTerm used in following pictures)

5.7.4 Run/Debug the example

5.7.4.1 MCUXpresso IDE

Run the example in MCUXpresso IDE:

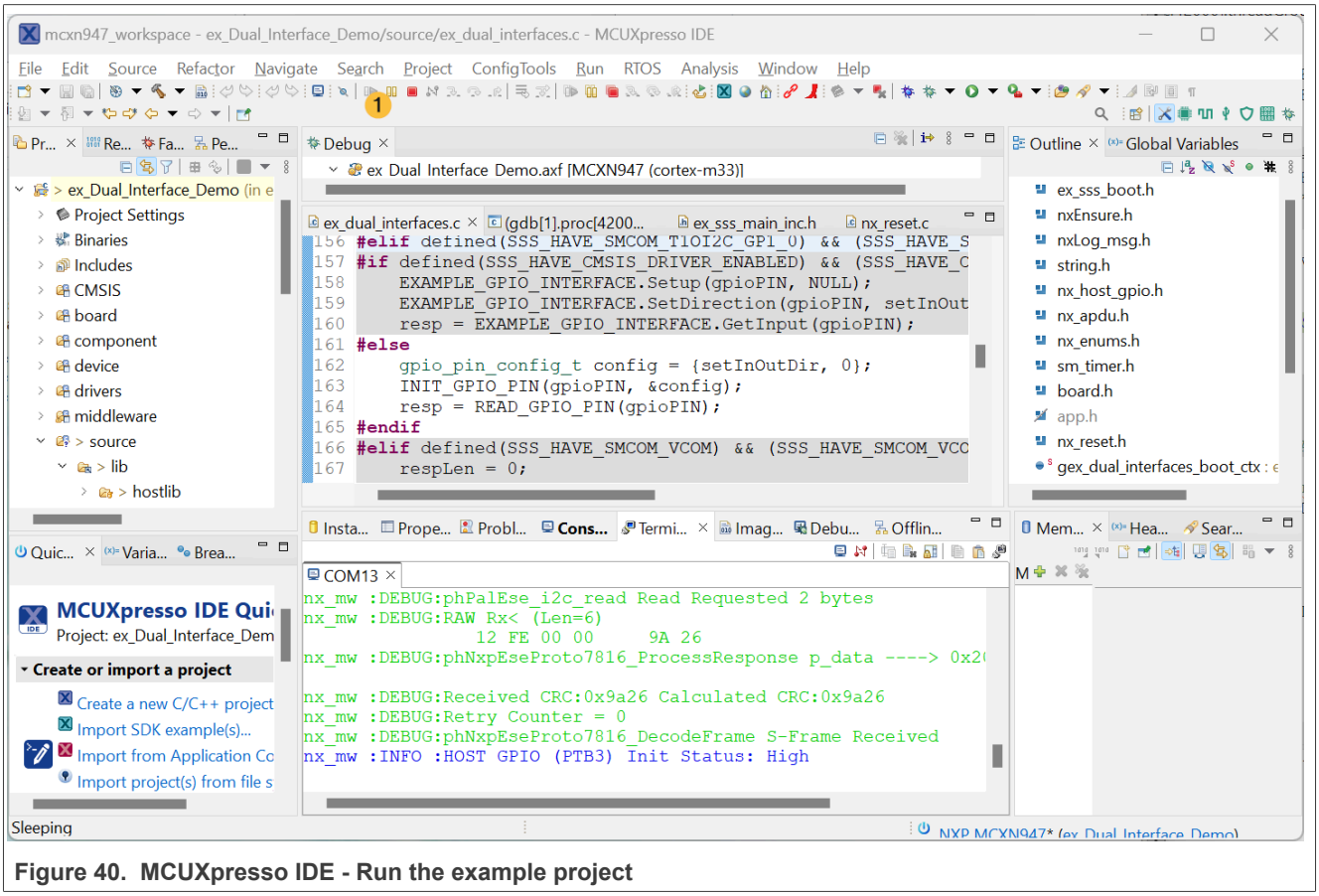

Figure 40. MCUXpresso IDE - Run the example project

NTAG X DNA and µC are waiting for read command (Cmd.ISOReadBinary or Cmd.ReadData) from NFC - that GPIO2 is set to Low.

5.7.4.2 Using RFIDDiscover

Open RFIDDiscover. Place NTAG-X-DNA-EVAL (while still connected to FRDM-MCXN947) on the Pegoda 3 reader. Connect to the reader if not done yet (Figure 33 and Figure 34).

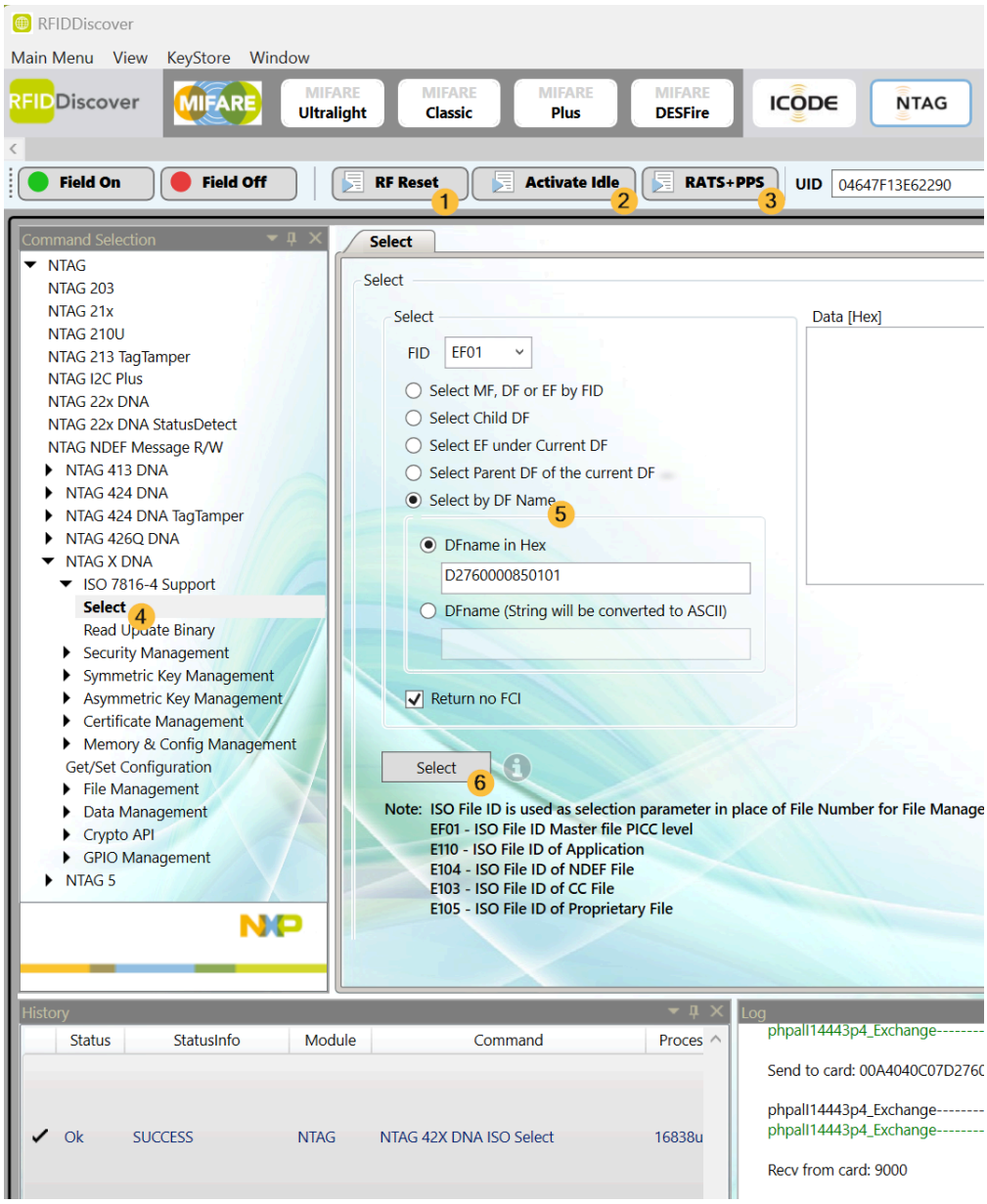

Figure 41. Perform steps group 1 with RFIDDiscover

1. Press "RF Reset", to turn RF field OFF and ON again
2. Press "Activate Idle" to perform ISO1444-3 activation
3. Press "RATS + PPS" to perform ISO1444-4 activation
4. Choose "Select" sub-menu under NTAG → NTAG X DNA → ISO 7816-4 Support
5. Choose "Select by DF name" (enter NDEF Application's DF name, if not predefined yet)
6. Press "Select". At this point, NDEF Application is selected.

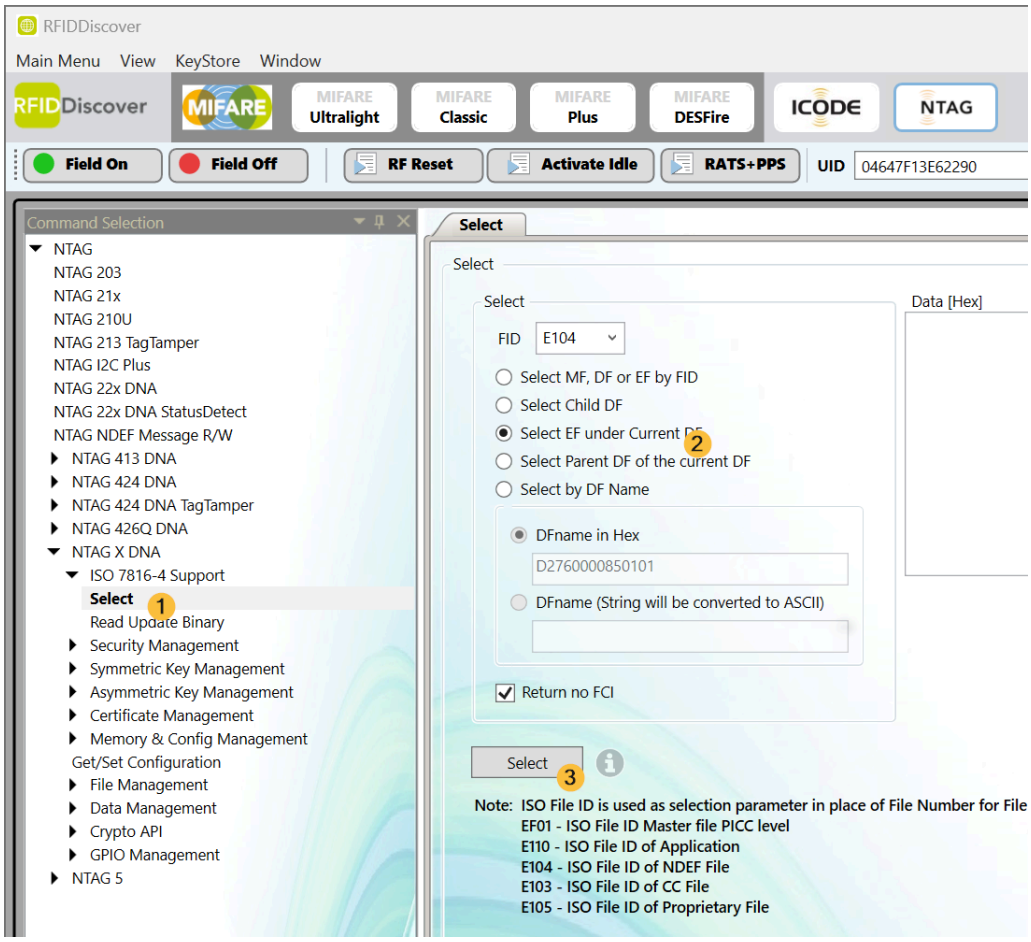

Figure 42. Perform steps group 2 with RFIDDiscover

These steps are not needed if NDEF file will be read by Cmd.ReadData

1. Choose "Select" sub-menu under NTAG → NTAG X DNA → ISO 7816-4 Support
2. Choose "Select EF under Current DF"
3. Press "Select". At this point, NDEF File (0xE104) is selected.

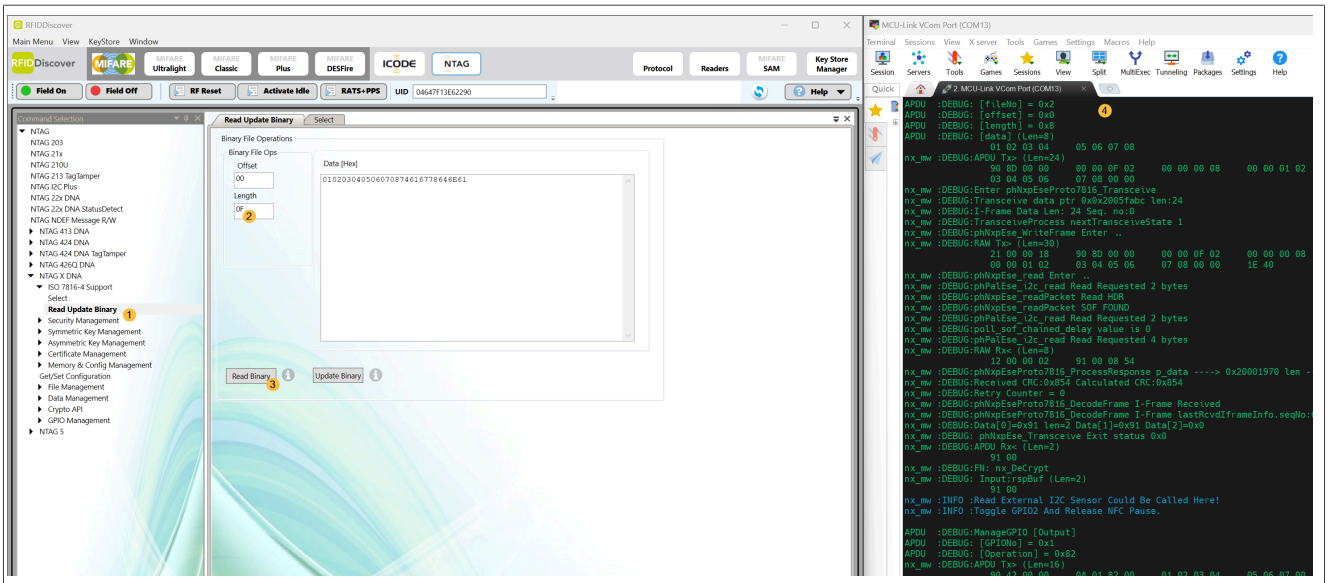

Figure 43. Perform steps group 3 with RFIDDiscover

1. Choose "Read Update Binary" sub-menu under NTAG → NTAG X DNA → ISO 7816-4 Support
2. Enter desired length to be read
3. Press "Read Binary" (if GPIO2Mode would be "0x05: output with NFCPause file" at this point, Cmd.ISOReadBinary would trigger GPIO2 to go LOW automatically). MCU will detect that, will write data to NDEF File and will send Cmd.ManageGPIO to toggle GPIO2 and make NTAG X DNA stop sending S(WTX) (waiting time extensions).

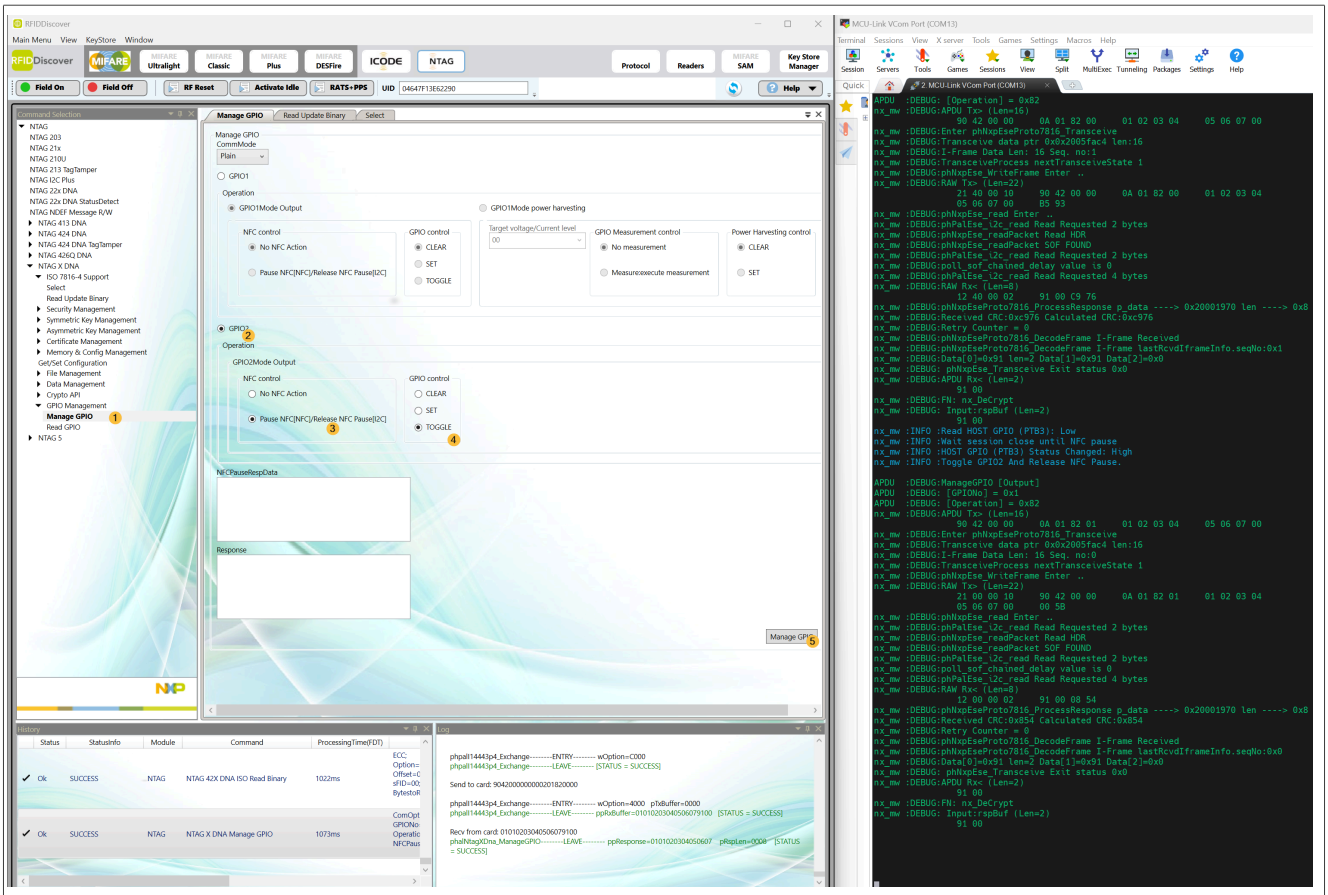

Figure 44. Perform steps group 4 with RFIDDiscover

This procedure shows, how NFCPause feature can be triggered by command - Cmd.ManageGPIO:

1. Choose "Manage GPIO" sub-menu under NTAG → NTAG X DNA → ISO 7816-4 Support
2. Select "GPIO2" radio button
3. Select "No NFC Action"
4. Choose "Toggle"
5. Press "Manage GPIO"

At this point, GPIO2 will be toggled. MCU will detect that and just toggle GPIO2 back and release NFC Pause - stop sending S(WTX) (waiting time extensions).

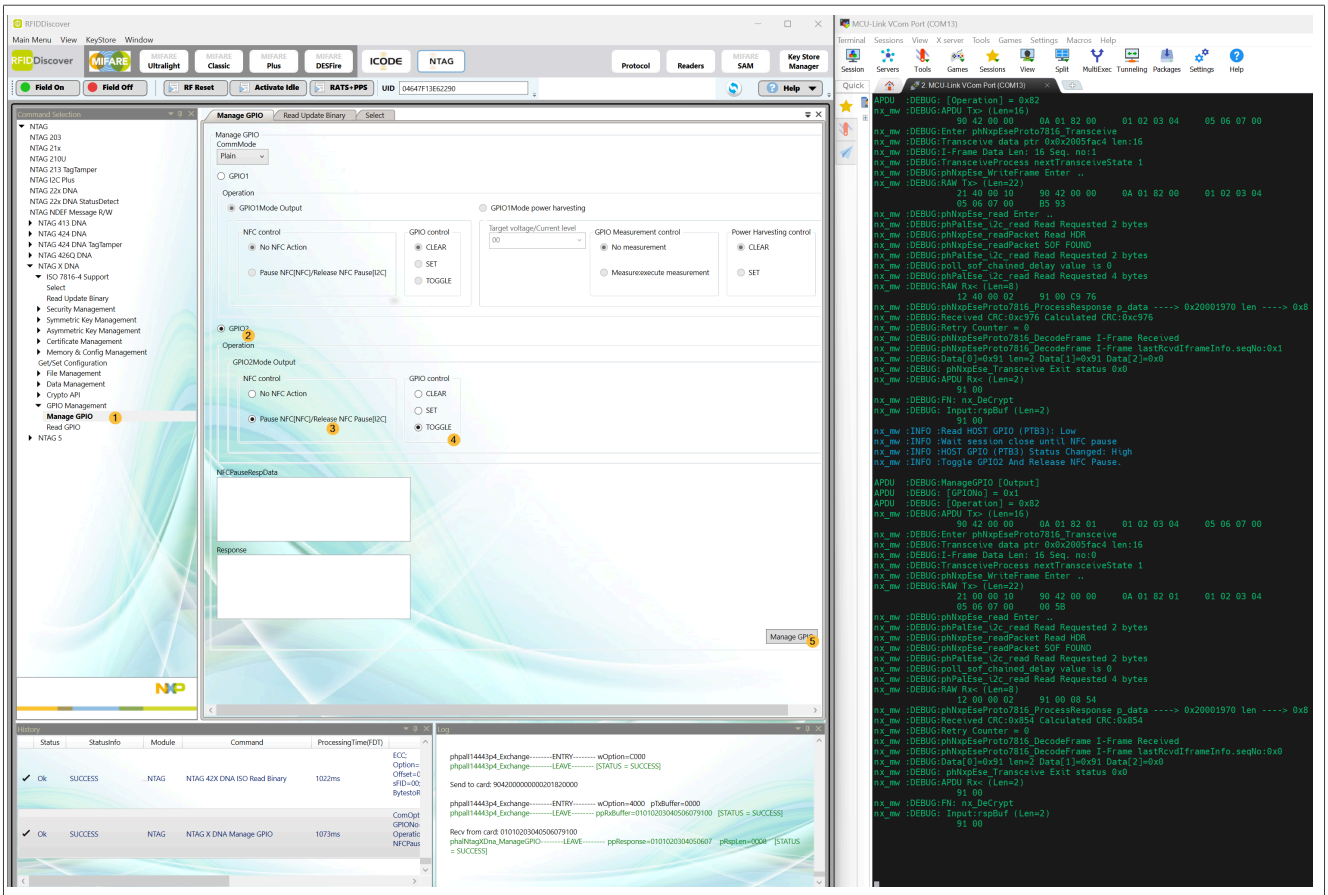

Figure 45. Perform steps group 5 with RFIDDiscover

5.7.4.3 Using Card Test Framework

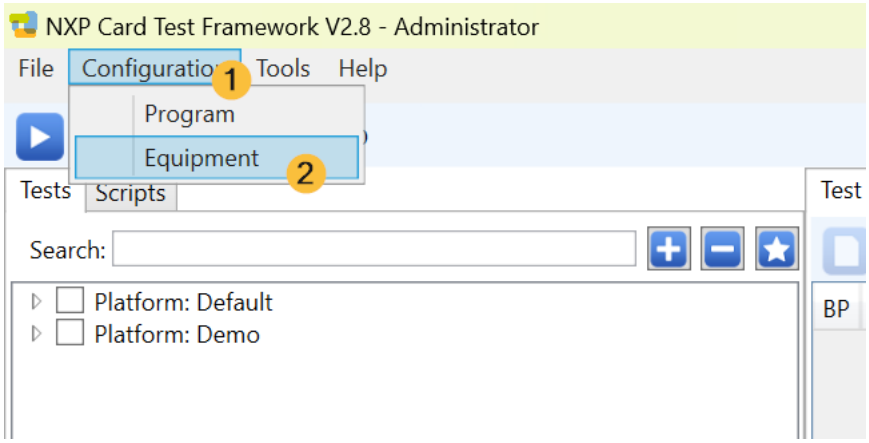

Figure 46. CTF - Perform steps group 1

Choose a connected and supported HF reader.

1. Click “Configuration”

2. Click “Equipment”

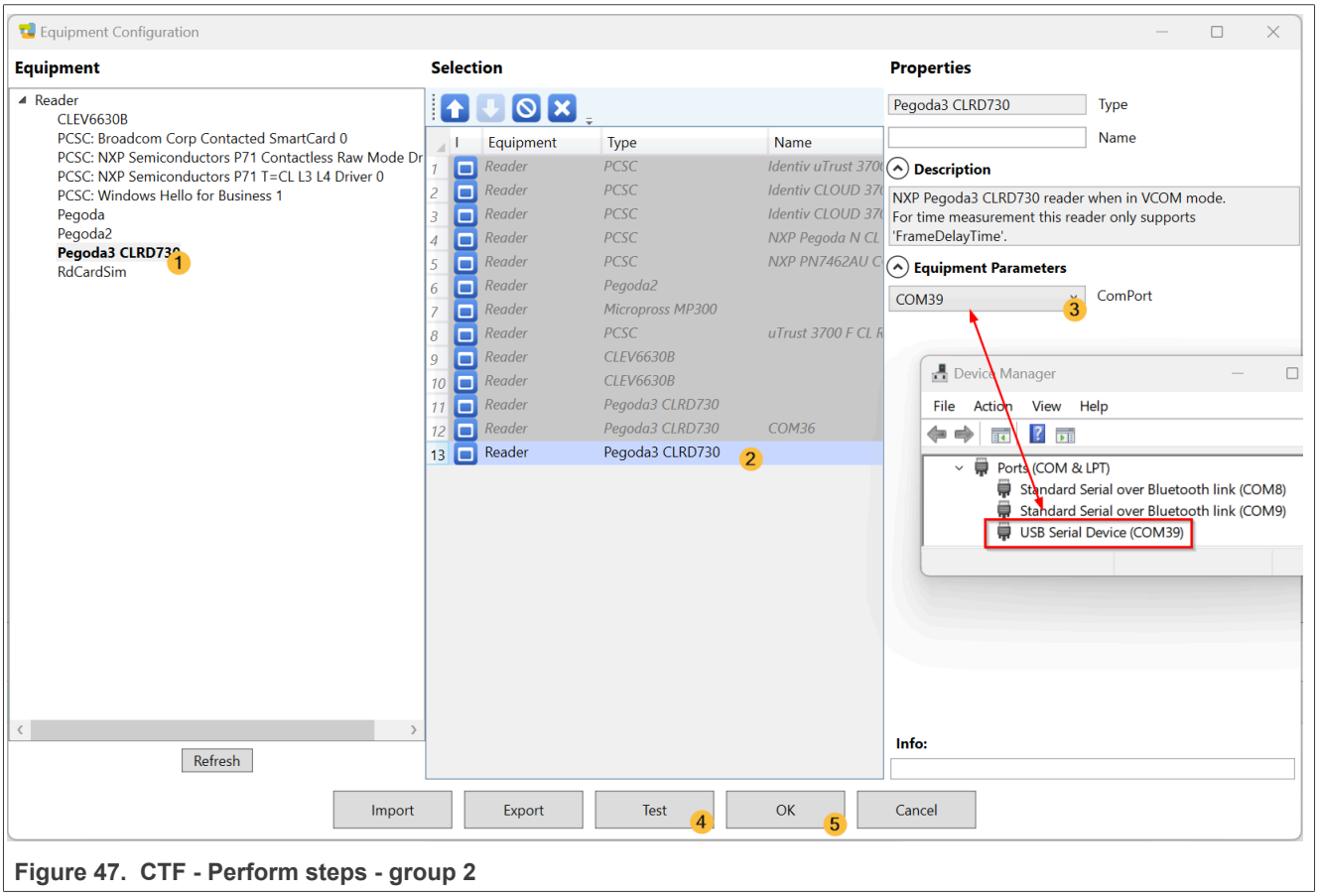

Figure 47. CTF - Perform steps - group 2

1. Add the correct reader from the left menu
2. Mark added reader
3. Check Device Manager list and adjust the reader's COM port (if the available COM port from Device Manager is not visible in Card Test Framework, restart the software)

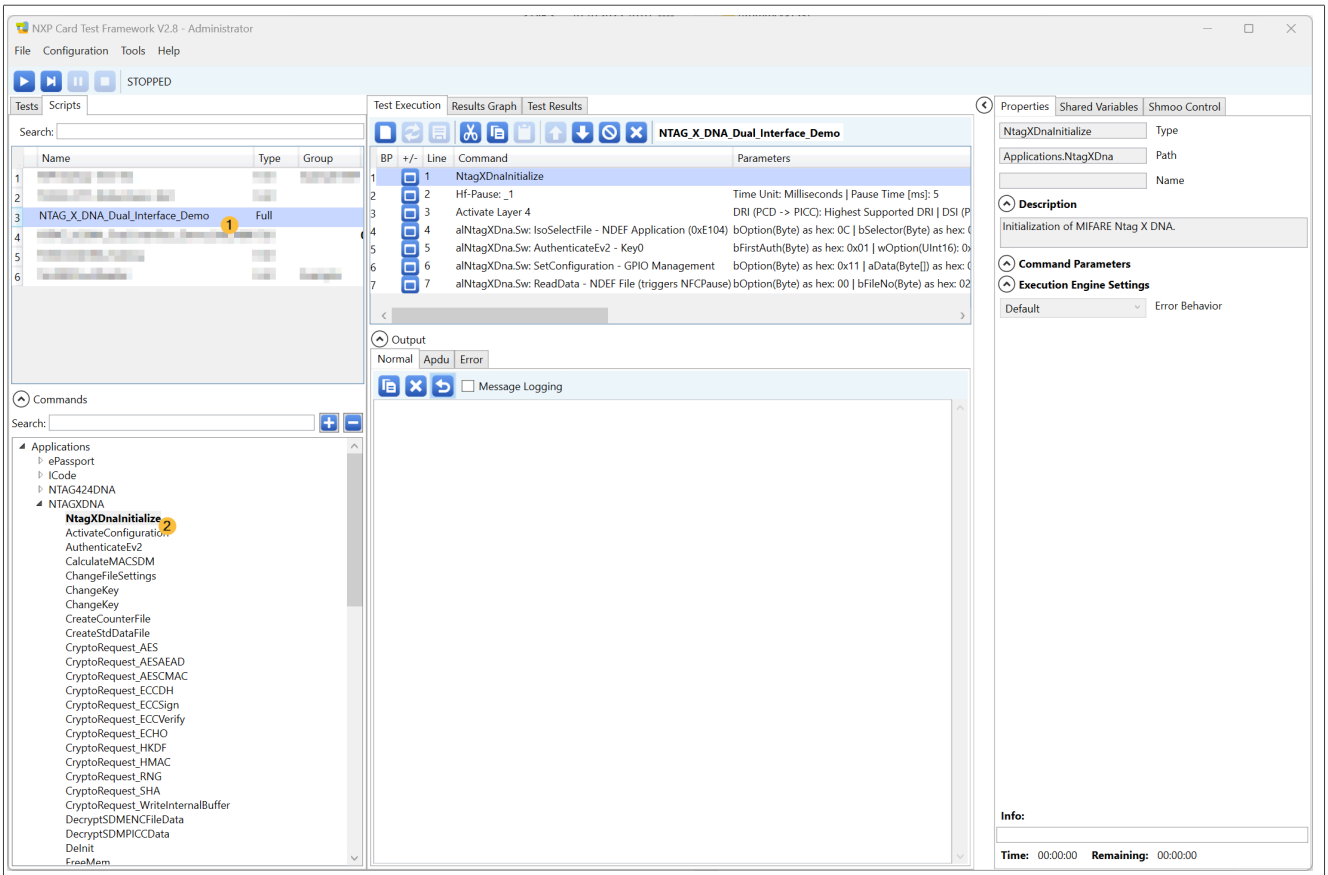

Figure 48. CTF - Perform steps - group 3

Before first run, NTAG X DNA needs to be configured for Dual Interface Demo as described in Section 5.7.2.2.

1. Search for "NTAG_X_DNA_Dual_Interface_Demo" script under the “Scripts” tab
2. Explore all of the steps and run the script.

5.7.5 Logic trace

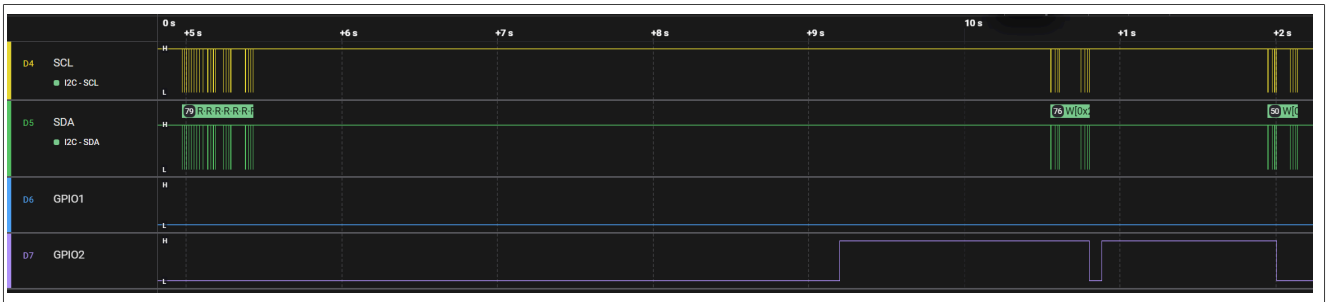

Figure 49. nx_dual_interfaces Logic trace - graphics

6 State diagrams

6.1 NTAG X DNA dual interface state diagram

Following image shows state diagram of NTAG X DNA interface arbitration.

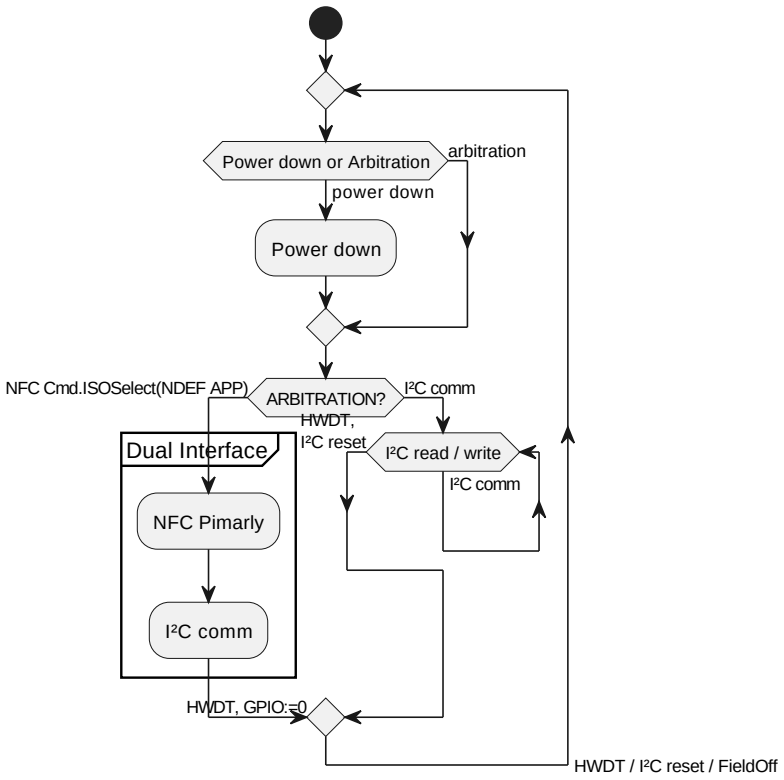

Figure 50. NTAG X DNA dual interface state diagram

6.2 Example of NFC reader application state diagram

The following figure shows an example of how the NFC reader SW can be designed for Section 4.5. Note that "WTX" ("NFC Pause") is standardized in ISO 14443-4 ref.[2].

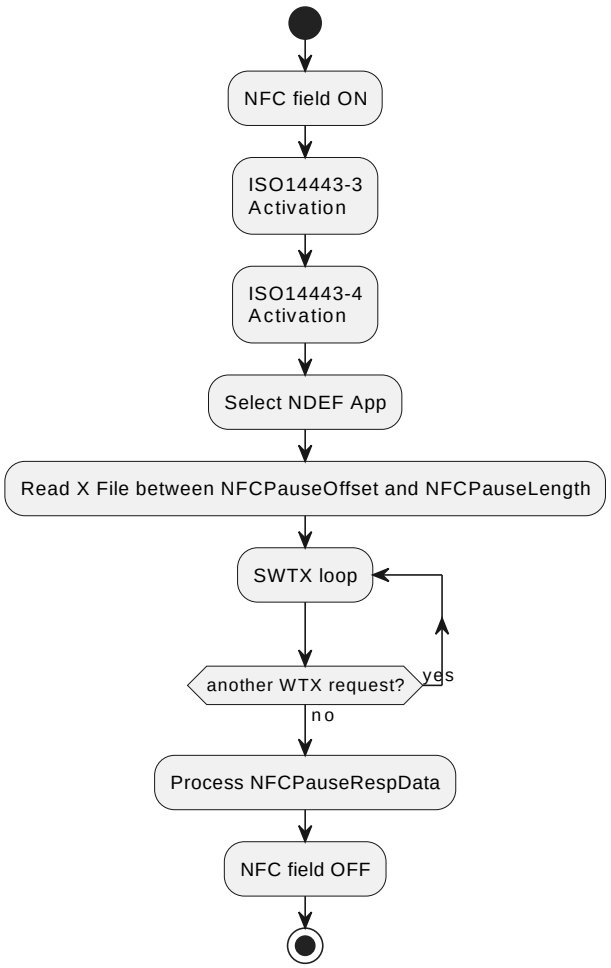

Figure 51. Example of NFC reader application state diagram

6.3 Example of MCU state diagram

The following figure shows an example of how the MCU SW can be designed for Section 4.5. The idea is to use interrupt on one of the GPIOs of the MCU.

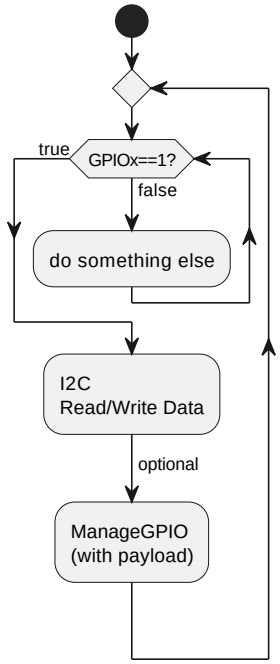

Figure 52. Example of MCU state diagram

7 Abbreviations

Table 5. Abbreviations

Acronym	Description
EH	Energy harvesting (NFC Field)
AID	Application IDentifier
APDU	Application Protocol Data Unit
DF-Name	ISO7816 Dedicated filename
C-APDU	Command APDU
CMAC	MAC according to NIST Special Publication 800-38B
CRC	Cyclic Redundancy Check
ECC	Eliptic Curve Cryptography
IC	Integrated Circuit
KDF	Key derivation function
LSB	Lowest Significant Byte
LSb	Lowest Significant bit
MAC	Message Authentication Code
NDEF	NFC Data Exchange Format
NDPP	NFC Digital Product Passport
NFC	Near Field Communication
NVM	Non-volatile memory
PCD	Proximity Coupling Device
PICC	Proximity-Integrated Circuit Card
PRF	Pseudo Random Function
R-APDU	Response APDU (received from PICC)
SSM	Standard Secure Messaging
SUN	Secure Unique NFC Messaging
UID	Unique Identifier
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

8 References

[1] Specification - ISO/IEC 14443-3:2018 - Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 3: Initialization and anti-collision

[2] Specification - ISO/IEC 14443-4:2018 - Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol

[3] Document - Globalplatform technology - apdu transport over spi / i2c - version 1.0. Version 1.0, January 2020

[4] Data sheet - NTAG X DNA - Secure NFC Forum T4T compliant IC with PKI (Public Interface Structure) (link)

[5] Application note - AN14137 - NTAG X DNA - Features and hints (link)

[6] Application note - AN14362 - NTAG X DNA - Energy harvesting (link)

[7] User guide - UG10083 - NTAG X DNA - Quick start guide with support package (link)

[8] Software - RFIDDiscover - PC application for NXP HF products evaluation. Available on https://www.nxp.com.

9 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10 Revision history

Table 6. Revision history

Document ID	Release date	Description
AN14513 v.2.0	10 November 2025	Editorial changes (typos, layout, figures, etc.) • Section 5 "NX Middleware examples": added. • Section 5.7 "Dual interface - NFCPause example": added. • Section 5.6 "GPIO - Notification (Authentication) example": added. • Section 5.5 "GPIO - Output/Input example": added. • Section 9 "Note about the source code in the document": added.
AN14513 v.1.0	27 May 2025	• Initial version

Legal information

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.

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.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://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.

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.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

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.

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.

Trademarks

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

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

MIFARE — is a trademark of NXP B.V.

Tables

Figures

Contents