

RX Family

PTX NFC Control Module Using Firmware Integration Technology

Introduction

This application note describes the usage of the PTX NFC control module, which conforms to the Firmware Integration Technology (FIT) standard.

In the following pages, the PTX NFC control module software is referred to collectively as “the PTX FIT module” or “the FIT module.”

The FIT module supports the following NFC PMOD module:

- PTX105R (PTX105RQC)

In the following pages, the PTX105R is referred to as “the NFC module”.

Target Device

- RX Family
- RX200 Series
- RX261 Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family

For details of the confirmed operation contents of each compiler, refer to “5.1 Confirmed Operation Environment”.

Related Documents

- [1] RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- [2] RX Smart Configurator User's Guide: e² studio (R20AN0451)
- [3] RX Family SCI Module Using Firmware Integration Technology (R01AN1815)
- [4] RX Family BYTEQ Module Using Firmware Integration Technology (R01AN1683)
- [5] RX Family CMT Module Using Firmware Integration Technology (R01AN1856)
- [6] RX Family GPIO Module Using Firmware Integration Technology (R01AN1721)
- [7] RX Family IRQ Module Using Firmware Integration Technology (R01AN1668)
- [8] EK-RX261 v1 – User's Manual (R20UT5351)

Contents

1. Overview	4
1.1 PTX NFC FIT Module	4
1.2 Overview of the PTX NFC FIT Module	4
1.2.1 Connection with the PTX NFC Module	4
1.2.2 Hardware Configuration	5
1.2.3 Software Configuration	6
1.3 API Overview	7
1.4 Status Transitions	8
2. API Information	9
2.1. Hardware Requirements	9
2.2. Software Requirements	9
2.3. Supported Toolchain	9
2.4. Interrupt Vector	9
2.5. Header Files	9
2.6. Integer Types	9
2.7. Compile Settings	10
2.8. Code Size	13
2.9. Parameters	14
2.10. Return Values	19
2.11. Adding the FIT Module to Your Project	20
2.12. “for”, “while” and “do while” statements	21
3. API Functions	22
3.1. R_NFC_PTX_Open()	22
3.2. R_NFC_PTX_StartDiscovery()	23
3.3. R_NFC_PTX_GetStatus()	24
3.4. R_NFC_PTX_GetCardRegistry()	25
3.5. R_NFC_PTX_ActivateCard()	26
3.6. R_NFC_PTX_DataExchange()	27
3.7. R_NFC_PTX_ReaderDeactivation()	29
3.8. R_NFC_PTX_SWReset()	30
3.9. R_NFC_PTX_PowerModeSet()	31
3.10. R_NFC_PTX_NDEF_Init()	32
3.11. R_NFC_PTX_NDEF_DataExchange()	34
3.12. R_NFC_PTX_NDEF_Lock()	36
3.13. R_NFC_PTX_Native_Tag_Init()	37
3.14. R_NFC_PTX_Native_Tag_Read()	39
3.15. R_NFC_PTX_Native_Tag_Write()	41
3.16. R_NFC_PTX_Native_Tag_Select()	42

3.17. R_NFC_PTX_Native_Tag_Lock()	44
3.18. R_NFC_PTX_Native_Tag_Sleep()	45
3.19. R_NFC_PTX_Close()	46
4. Demo Projects	47
4.1. NFC PTX105RQC Get Card Information Demo Project	47
4.1.1 Prerequisites	47
4.1.2 Import the Demo Project	47
4.1.3 Hardware Setup	48
4.1.4 How to Run the Demo	49
4.1.5 Understanding NFC Data Fields in the Example	55
4.2. Adding a Demo to a Workspace	57
4.3. Downloading Demo Projects	57
5. Appendices	58
5.1 Confirmed Operation Environment	58
5.2 Troubleshooting	59
6. Reference Documents	60
Revision History	61

1. Overview

1.1 PTX NFC FIT Module

The PTX NFC FIT module can be used by being implemented in a project as an API. See section “2.11 Adding the FIT Module to Your Project” for details on methods to implement this FIT module into a project.

1.2 Overview of the PTX NFC FIT Module

The PTX NFC module supplies an implementation for the NFC IoT Reader on the PTX module. The NFC functionalities are provided via a modular NFC Soft Controller (NCS) architecture, and this FIT module will include the NSC (known as the NFC SDK). Functionality is provided in a split-stack architecture, where time-critical operations are running on the on-chip MCU, and the rest of the NFC logic is handled by the host controller to carry out applications such as the IoT Reader. The driver may also be adapted to further support next-generation PTX devices in future (which have not yet been released).

- The PTX NFC FIT Module supplies these features:
- Initializes the IOTRD (IoT Reader) API and NSC stack
- Initializes the PTX chip
- Discovers cards according to NFC Forum
- Selects a specific card in case multiple cards/protocols were discovered
- Retrieves card details like technical/activation parameters
- Exchanges RF data and bitstreams
- Stops RF communication
- Controls various GPIO pins of the PTX chip
- Provides NDEF (NFC Data Exchange Format) message handling for standardized NFC data exchange operations
- Native-Tag add-on API for implementing the Application Protocol Data Units (APDU) over ISO-DEP

1.2.1 Connection with the PTX NFC Module

An example connection to the PTX NFC module is shown below.

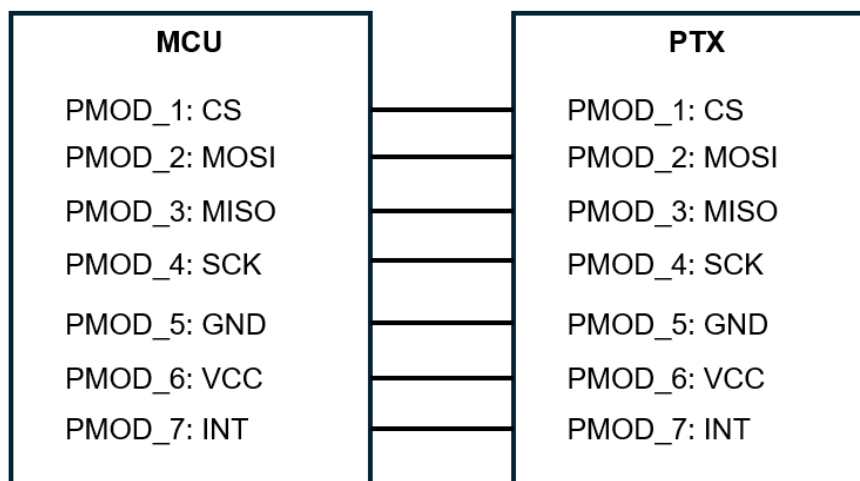


Figure 1.1 Example Connection to the PTX NFC Module

1.2.2 Hardware Configuration

The hardware configuration for MCU host and the PTX105R NFC PMOD module is shown below.

Table 1.1 MCU Host Hardware Configuration

Item	Content	Description
EK-RX261 Evaluation Kit *1	Target board for EK-RX261 Part number: RTK5EK2610S00001BE	Please see detail at: https://www.renesas.com/rx/ek-rx261

Note 1: The EK-RX261 Evaluation Kit only supports PMOD1 for the NFC PMOD module

Table 1.2 PMOD Module Hardware Configuration

Item	Content	Description
PTX105R NFC PMOD module	NFC IoT Reader connection	This PMOD is used with MCU host for NFC connection. Please see detail at: PTX105RQC - PTX105R PMOD™ Board for IoT Renesas

1.2.3 Software Configuration

Figure 1.2 shows the software configuration.

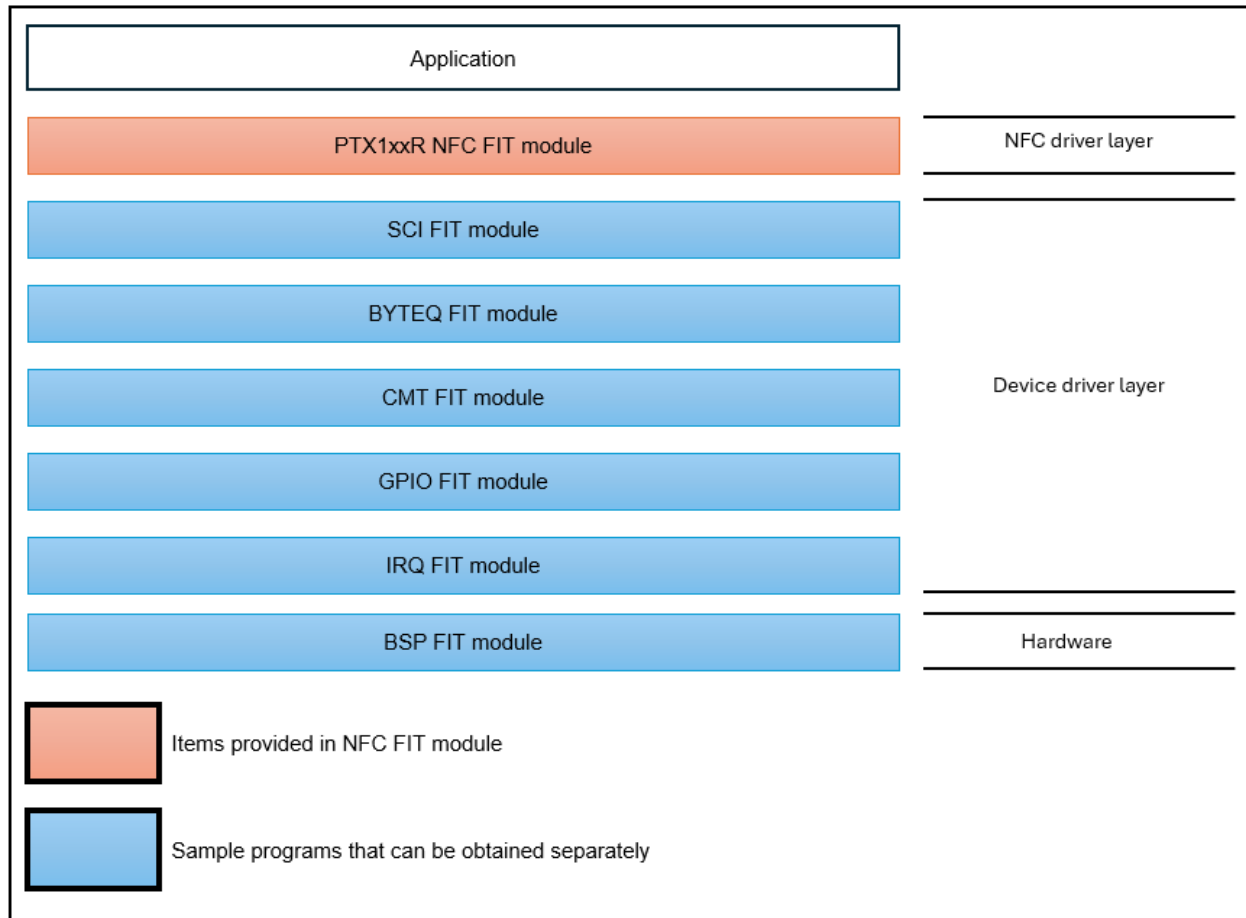


Figure 1.2 Software Configuration Diagram

1. PTX NFC FIT module
The FIT module. This software is used to control the NFC module.
2. SCI FIT module
Implements communication between the NFC module and the MCU. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.
3. BYTEQ FIT module
Implements circular buffers used by the SCI FIT module. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.
4. CMT FIT module
Implements compare match timer used by the PTX NFC FIT module. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.
5. GPIO FIT module
Implements pin direction set function used by the PTX NFC FIT module. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.
6. IRQ FIT module
Implements external IRQ source used by the PTX NFC FIT module. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.
7. BSP FIT module
The Board Support Package module. A sample program is available. Refer to "Related Documents" on page 1 and obtain the software.

1.3 API Overview

Table 1.3 lists the API functions included in the FIT module. The required memory sizes are lists in “2.8 Code Size”

Table 1.3 API Functions

Function	Function Description
R_NFC_PTX_Open()	Initialize the NFC module
R_NFC_PTX_StartDiscovery()	Initiates discovery of NFC tags
R_NFC_PTX_GetStatus()	Gets the status information on the provided status identifier (system, discovery, etc.)
R_NFC_PTX_GetCardRegistry()	Gets the internal card registry
R_NFC_PTX_ActivateCard()	Gets the details of the discovered card and connects to it
R_NFC_PTX_DataExchange()	Sends and receives data to/from the activated card
R_NFC_PTX_ReaderDeactivation()	Deactivates the activated card to return to the desired state (idle, discovery, etc.).
R_NFC_PTX_SWReset()	Performs a soft-reset of the PTX chip.
R_NFC_PTX_PowerModeSet()	Puts the device into either Active (all features) or Stand-by mode (most features off), to cut down energy consumption.
R_NFC_PTX_NDEF_Init()	Initializes the optional NDEF API add-on.
R_NFC_PTX_NDEF_DataExchange()	Exchanges NDEF data with an NFC tag by reading from or writing to it.
R_NFC_PTX_NDEF_Lock()	Locks an NFC tag to prevent writing to it, making it read-only.
R_NFC_PTX_Native_Tag_Init()	Initializes the optional Native-Tag API add-on. Must be done after NFC stack is initialized.
R_NFC_PTX_Native_Tag_Read()	Performs a read operation using the Native-Tag API add-on.
R_NFC_PTX_Native_Tag_Write()	Performs a write operation using the Native-Tag API add-on.
R_NFC_PTX_Native_Tag_Select()	Performs a Select operation using the Native-Tag API add-on, which is different for each tag type.
R_NFC_PTX_Native_Tag_Lock()	Performs a Lock operation using the Native-Tag API add-on. - This function is exclusive to Type 5 tags.
R_NFC_PTX_Native_Tag_Sleep()	Performs a Sleep operation using the Native-Tag API add-on. - This function is exclusive to Type 5 tags.
R_NFC_PTX_Close()	Closes the NFC module and resets all variables.

1.4 Status Transitions

Figure 1.3 shows the status transitions of the FIT module.

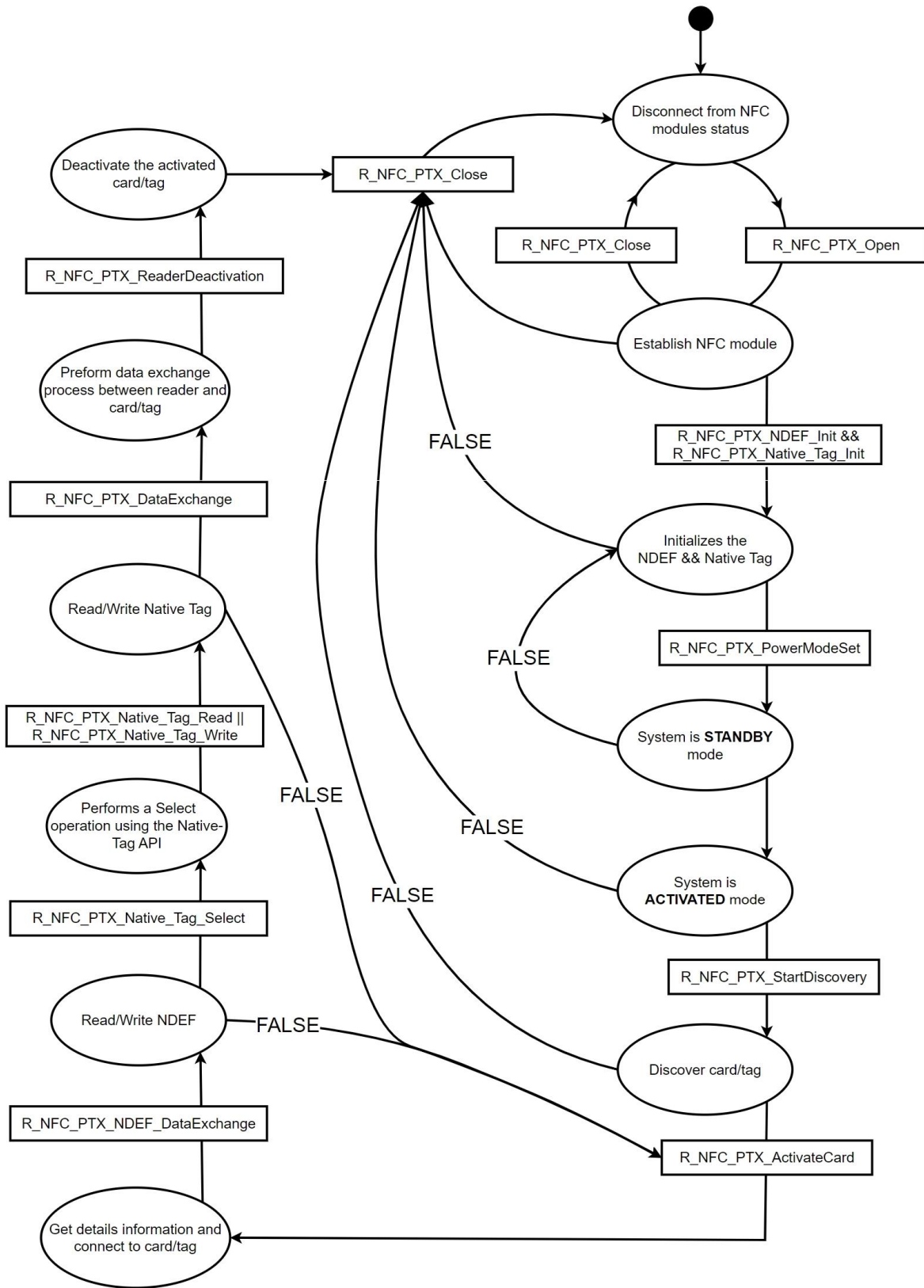


Figure 1.3 Status Transitions

(Note: Can check detail state machine in PTX1xxR NFC IoT-Reader API Non-OS Stack Integration User Manual).

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1. Hardware Requirements

The MCU used must support the following functions:

- Serial communication
- I/O ports

2.2. Software Requirements

The driver is dependent upon the following FIT modules:

- r_bsp
- r_sci_rx
- r_byteq_rx
- r_cmt_rx
- r_gpio_rx
- r_irq_rx

2.3. Supported Toolchain

The FIT module has been confirmed to work with the toolchain listed in “5.1 Confirmed Operation Environment”.

2.4. Interrupt Vector

None

2.5. Header Files

All API calls and their supporting interface definitions are located in r_nfc_ptx_if.h.

2.6. Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.7. Compile Settings

The configuration option settings of the FIT module are contained in `r_nfc_ptx_rx_config.h`. The names of the options and their setting values are listed in the table below.

Table 2.1 Configuration Options (`r_nfc_ptx_rx_config.h`)

Configuration Options in <code>r_nfc_ptx_rx_config.h</code>	
NFC_CFG_EXT_IRQ_PORT Default: "D"	Configure the general port that controls IRQ port triggered by PTX to indicate to host that data is available. Set this option to match the port to be controlled.
NFC_CFG_EXT_IRQ_PIN Default: "5"	Configure the general pin that controls IRQ pin triggered by PTX to indicate to host that data is available. Set this option to match the pin to be controlled
NFC_CFG_IRQ_NUM Default: "5"	IRQ number that connected to IRQ pin of PTX Set this option to match the IRQ number to be controlled.
NFC_CFG_SCI_CHANNEL Default: "5"	SCI channel for serial communication with PTX. Set this option to match the SCI port to be controlled.
NFC_CFG_SCI_SSPI_SS_PORT Default: "C"	Configure the general port that controls SS port (for SPI interface only). Set this option to match the port to be controlled.
NFC_CFG_SCI_SSPI_SS_PIN Default: "0"	Configure the general pin that controls SS pin (for SPI interface only). Set this option to match the pin to be controlled.
NFC_CFG_SCI_MODE Default: "0"	Configure SCI mode for communication 0 = Simple SPI
NFC_CFG_SCI_SSPI_SPEED Default: "1,000,000"	Configure SCI SSPI bitrate Set this value in range from 1,000,000 to 10,000,000
NFC_SCI_SPI_MSB_FIRST Default: "1"	Configure SCI data transfer mode 0: LSB first transfer, 1: MSB first transfer
NFC_CFG_SLEEP_TIMER_CHANNEL Default: "0"	Configure sleep timer channel Set this option to match the timer to be controlled
NFC_CFG_TDC_TIMER_CHANNEL Default: "1"	Configure Transparent Data Channel timer channel Set this option to match the timer to be controlled
NFC_CFG_NDEF_SUPPORT Default: "0"	Configure to include the APIs provided by the NDEF Add-on. 0: Disable, 1: Enable
NFC_CFG_NATIVE_TAG_SUPPORT Default: "0"	Controls native NFC tag functionality for reading/writing custom data to blank NFC tags. 0: Not use Native Tag, 1: Use Native Tag.
NFC_CFG_NDEF_WORK_BUF_SIZE Default: "64"	Determines the maximum size of NDEF messages that can be handled This buffer is used for NDEF read/write operations
NFC_CFG_DISCOVER_MODE Default: "0"	Configure discovery polling mode: 0: Regular Polling 1: Low-power Card Detection (LPCD) 2 - 255: LPCD with every n-th cycle as Regular Polling
NFC_CFG_POLL_TYPE_A Default: "1"	Select to discover NFC Type-A tags 0: Disable, 1: Enable
NFC_CFG_POLL_TYPE_B Default: "1"	Select to discover NFC Type-B tags 0: Disable, 1: Enable

NFC_CFG_POLL_TYPE_F Default: "1"	Select to discover NFC Type-F tags 0: Disable, 1: Enable
NFC_CFG_POLL_TYPE_V Default: "1"	Select to discover NFC Type-V tags 0: Disable, 1: Enable
NFC_CFG_TEMP_SENSOR_SHUTDOWN	Set the expected thermal shutdown threshold value (Celsius). Set this value in range from 1 to 222
NFC_CFG_TEMP_SENSOR_AMBIENT	Set the ambient temperature value at which calibration takes place (Celsius). Set this value in range from 1 to 84
NFC_CFG_DEVICE_LIMIT	Select the max number of tags to discover Set this value in range from 1 to 49
NFC_CFG_IDLE_TIME_MS	Select idle time between polling cycles (ms) Set this value in range from 1 to 65534

Table 2.2 Configuration Options (r_sci_rx_config.h)

Configuration Options in r_sci_rx_config.h	
<pre>#define SCI_CFG_SSPI_INCLUDED</pre> <p>Note: The default is 0.</p>	<p>Specifies whether to include code for SSPI mode.</p> <p>This option needs to be set to 1 in this v1.00 (only support SPI).</p>
<pre>#define SCI_CFG_CHx_INCLUDED</pre> <p>Notes: 1. CHx = CH0 to CH12 2. The default values are as follows: CH0 CH2 to CH12: 0, CH1: 1</p>	<p>Each channel has resources such as transmit and receive buffers, counters, interrupts, other programs, and RAM. Setting this option to 1 assigns related resources to the specified channel.</p>
<pre>#define SCI_CFG_TEI_INCLUDED</pre> <p>Note: The default is 0.</p>	<p>Enables the transmit end interrupt for serial transmissions. This option should be set to 1.</p>

2.8. Code Size

Typical code sizes associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.7 *Compile Settings*. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.3 *Supported Toolchain*. The compile option default values are optimization level: 2, optimization type: for size, and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

The values in the table below are confirmed under the following conditions.

Module Revision: r_nfc_ptx_rx rev1.20.

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.07.00

(The option of "-lang=c99" is added to the default settings of the integrated development environment.)

Configuration Options: Default settings.

Table 2.3 Memory Sizes for RX261

Device	Memory Type	Category	Memory usage
			Renesas Compiler (CCRX)
RX261	With platform modules	ROM	152300 bytes
		RAM	60928 bytes
	Without platform modules	ROM	131153 bytes
		RAM	8265 bytes

(**Note:** platform modules are low-level modules: SCI, GPIO, IRQ, CMT).

2.9. Parameters

This section describes the parameter structure used by the API functions in this module. The structure is located in `r_nfc_ptx_if.h` as are the prototype declarations of API functions.

Table 2.4 Member in Structure for Data Exchange Packet (`nfc_reader_ptx_data_info_t`)

Type	Name	Description
uint8_t *	p_tx_buf	Pointer to buffer holding data to send
uint32_t	tx_length	Length of data to send
uint8_t *	p_rx_buf	Pointer to buffer for data to receive
uint32_t	rx_length	Length of data to receive

Table 2.5 Member in Structure for NFC Configuration Block (`nfc_reader_ptx_cfg_t`)

Type	Name	Description
bool	poll_type_a	Flag to indicate enabling discovery for Type-A tags
bool	poll_type_b	Flag to indicate enabling discovery for Type-B tags
bool	poll_type_f	Flag to indicate enabling discovery for Type-F tags
bool	poll_type_v	Flag to indicate enabling discovery for Type-V tags
uint8_t	device_limit	Defines the maximum number of devices that can be connected at once
uint32_t	idle_time_ms	Idle time between polling cycles in milliseconds
uint8_t	temp_sensor_calibrate	Flag to enable/disable temperature sensor calibration
uint8_t	temp_sensor_shutdown	Expected thermal shutdown threshold value
uint8_t	temp_sensor_ambient	Ambient temperature at which temperature sensor calibration takes place
ptxIoTRd_t *	iot_reader_context	Instance of the NFC SDK IoT Reader main structure
ptxNDEF_t *	p_ndef_context	Pointer to the optional NDEF API context

Table 2.6 Member in Structure for NFC State Machine Status (`nfc_reader_ptx_state_t`)

Value	Name	Description
1	NFC_PTX_IDLE	Idle state before starting discovery
2	NFC_PTX_POLLING	Polling state for discovering tags
3	NFC_PTX_DISCOVERED	Discovered state for tags that were found
4	NFC_PTX_ACTIVATED	Activated state for activating a discovered tag
5	NFC_PTX_ERROR_TEMP	Error state for temperature sensor
6	NFC_PTX_ERROR_OVERCURRENT	Error state for overcurrent
7	NFC_PTX_ERROR_RF	Error state for RF error

Table 2.7 Member in Structure for NFC Control Block (nfc_reader_ptx_ctrl_t)

Type	Name	Description
uint8_t	ndef_work_buf	Buffer for NDEF internal data exchange
uint32_t	open	Flag to indicate if NFC has been opened
uint32_t	ndef_open	Flag to indicate if NDEF has been opened
uint32_t	native_tag_open	Flag to indicate if NFC Native-Tag has been initialized
nfc_reader_ptx_state_t	state_flag	Flag to track Idle/discovered/activated
ptxNativeTag_T2T_t	t2t_comp	Type 2 tag component for Native-Tag add-on API
ptxNativeTag_T2T_InitParams_t	t2t_params	Type 2 tag parameters for Native-Tag add-on API component
ptxNativeTag_T3T_t	t3t_comp	Type 3 tag component for Native-Tag add-on API
ptxNativeTag_T3T_InitParams_t	t3t_params	Type 3 tag parameters for Native-Tag add-on API component
ptxNativeTag_T4T_t	t4t_comp	Type 4 tag component for Native-Tag add-on API
ptxNativeTag_T4T_InitParams_t	t4t_params	Type 4 tag parameters for Native-Tag add-on API component
ptxNativeTag_T5T_t	t5t_comp	Type 5 tag component for Native-Tag add-on API
ptxNativeTag_T5T_InitParams_t	t5t_params	Type 5 tag parameters for Native-Tag add-on API component
uint8_t	enabled_tag_mask	Bit mask of which tag types were initialized
nfc_reader_ptx_cfg_t const *	p_cfg	Pointer to configuration block for NFC

Table 2.8 Member in Structure for NDEF data exchange flags (nfc_reader_ptx_ndef_read_write_t)

Value	Name	Description
1	NFC_PTX_NDEF_WRITE	Write flag for NDEF data exchange
2	NFC_PTX_NDEF_READ	Read flag for NDEF data exchange

Table 2.9 Member in Structure for NFC power mode flags (nfc_reader_ptx_power_mode_t)

Value	Name	Description
0	NFC_PTX_ACTIVE_MODE	Active mode, all features enabled
1	NFC_PTX_STANDBY_MODE	Stand-by mode, most features disabled for low power

Table 2.10 Member in Structure NFC Native-tag Optional Parameters for Read Operation (nfc_reader_ptx_native_tag_read_params_t)

Type	Name	Description
uint8_t	t2t_block_number	Type 2 tag block number for Native-Tag Read/Write functions
ptxNativeTag_T3T_Services_t	t3t_service_info	Type 3 tag service code information for Native-Tag Check/Update functions
ptxNativeTag_T3T_Blocks_t	t3t_block_info	Type 3 tag block information for Native-Tag Check/Update functions
uint8_t*	t3t_nfcid2	Type 3 tag NFCID2 identifier for Native-Tag Check/Update functions
uint8_t	t3t_nfcid2_len	Type 3 tag NFCID2 identifier length for Native-Tag Check/Update functions
uint32_t	t4t_file_offset	Type 4 tag file offset for Native-Tag Read/Update (and ODO) functions
uint8_t	t4t_data_num_bytes	Type 4 tag number of expected response bytes for Native-Tag Read (and ODO) functions
uint8_t	t5t_option_flag	Type 5 tag command option flag for Native-Tag Read/Write/Select/Lock/Sleep functions
uint16_t	t5t_block_number	Type 5 tag starting block number for Native-Tag Read function
uint8_t	t5t_num_blocks	Type 5 tag number of blocks to read for Native-Tag Read function

Table 2.11 Member in Structure NFC Native-tag Optional Parameters for Write Operation (nfc_reader_ptx_native_tag_write_params_t)

Type	Name	Description
uint8_t	t2t_block_number	Type 2 tag block number for Native-Tag Read/Write functions
ptxNativeTag_T3T_Services_t	t3t_service_info	Type 3 tag service code information for Native-Tag Check/Update functions
ptxNativeTag_T3T_Blocks_t	t3t_block_info	Type 3 tag block information for Native-Tag Check/Update functions
uint8_t*	t3t_nfcid2	Type 3 tag NFCID2 identifier for Native-Tag Check/Update functions
uint8_t	t3t_nfcid2_len	Type 3 tag NFCID2 identifier length for Native-Tag Check/Update functions
uint32_t	t4t_file_offset	Type 4 tag file offset for Native-Tag Read/Update (and ODO) functions
uint16_t	t5t_block_number	Type 5 tag starting block number for Native-Tag Write function
uint8_t	t5t_option_flag	Type 5 tag command option flag for Native-Tag Read/Write/Select/Lock/Sleep functions

Table 2.12 Member in Structure NFC Native-tag Optional Parameters for Select Operation (nfc_reader_ptx_native_tag_select_params_t)

Type	Name	Description
uint8_t	t2t_sector_number	Type 2 tag sector number for Native-Tag Select function
uint32_t	t4t_file_offset	Type 4 tag file offset for Native-Tag Read/Update (and ODO) functions
uint8_t	t4t_apdu_byte_1	Type 4 tag APDU (command header) first byte for Native-Tag Select function
uint8_t	t4t_apdu_byte_2	Type 4 tag APDU (command header) second byte for Native-Tag Select function
uint8_t	t4t_expected_len	Type 4 tag expected length of response for Native-Tag Select function
uint8_t	t5t_option_flag	Type 5 tag command option flag for Native-Tag Read/Write/Select/Lock/Sleep functions
uint8_t*	t5t_uid	Type 5 tag UID identifier for Native-Tag Select/Sleep functions
uint8_t	t5t_uid_len	Type 5 tag UID identifier length for Native-Tag Select/Sleep functions

Table 2.13 Member in Structure NFC Native-tag Optional Parameters for Lock Operation (nfc_reader_ptx_native_tag_lock_params_t)

Type	Name	Description
uint8_t	t5t_option_flag	Type 5 tag command option flag for Native-Tag Read/Write/Select/Lock/Sleep functions
uint16_t	t5t_block_number	Type 5 tag starting block number for Native-Tag Lock function

Table 2.14 Member in Structure NFC Native-tag Optional Parameters for Sleep Operation (nfc_reader_ptx_native_tag_sleep_params_t)

Type	Name	Description
uint8_t	t5t_option_flag	Type 5 tag command option flag for Native-Tag Read/Write/Select/Lock/Sleep functions
uint8_t*	t5t_uid	Type 5 tag UID identifier for Native-Tag Select/Sleep functions
uint8_t	t5t_uid_len	Type 5 tag UID identifier length for Native-Tag Select/Sleep functions

Table 2.15 Member in Structure NFC Native-tag Type Flag (nfc_reader_ptx_native_tag_type_t)

Value	Name	Description
0	NFC_READER_PTX_NATIVE_TAG_TYPE_NONE	No tag was selected for Native-Tag add-on APIs
(1U << 0)	NFC_READER_PTX_NATIVE_TAG_TYPE_T2T	Type 2 tag bit mask for Native-Tag add-on APIs
(1U << 1)	NFC_READER_PTX_NATIVE_TAG_TYPE_T3T	Type 3 tag bit mask for Native-Tag add-on APIs
(1U << 2)	NFC_READER_PTX_NATIVE_TAG_TYPE_T4T	Type 4 tag bit mask for Native-Tag add-on APIs
(1U << 3)	NFC_READER_PTX_NATIVE_TAG_TYPE_T5T	Type 5 tag bit mask for Native-Tag add-on APIs

2.10. Return Values

This section describes return values of API functions. This enumeration is located in `r_nfc_ptx_if.h` as are the prototype declarations of API functions.

Table 2.16 API Error Codes (nfc_err_t)

Value	Error code	Description
0	NFC_SUCCESS	OK, no error
1	NFC_ERR_ASSERTION	Parameters assertion
2	NFC_ERR_INVALID_DATA	Input data input is invalid or incomplete
3	NFC_ERR_NOT_OPEN	NFC module has not been opened
4	NFC_ERR_ALREADY_OPEN	NFC module had been initialized
5	NFC_ERR_INVALID_STATE	Invalid status state
6	NFC_ERR_INVALID_ARGUMENT	Invalid input argument
7	NFC_ERR_INVALID_MODE	One of the available tag types was not selected
8	NFC_ERR_UNSUPPORTED	Current feature is not supported

2.11. Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) or (5) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

- (1) Adding the FIT module to your project using the Smart Configurator in e² studio
By using the Smart Configurator in e² studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: e² studio (R20AN0451)” for details.
- (2) Adding the FIT module to your project using the FIT Configurator in e² studio
By using the FIT Configurator in e² studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.
- (3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: CS+ (R20AN0470)” for details.
- (4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.
- (5) Adding the FIT module to your project using the Smart Configurator in IAREW
By using the Smart Configurator Standalone version, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: IAREW (R20AN0535)” for details.

2.12. “for”, “while” and “do while” statements

In FIT module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

This FIT module does not have any WAIT_LOOP. But others might have. Please take care for this WAIT_LOOP.

3. API Functions

3.1. R_NFC_PTX_Open()

This function initializes the NFC IoT Reader.

Format

```
nfc_err_t R_NFC_PTX_Open(  
    nfc_reader_ptx_ctrl_t * const p_ctrl,  
    nfc_reader_ptx_cfg_t const * const p_cfg  
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_cfg	Pointer to NFC IoT Reader configuration structure

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assert error occurred
NFC_ERR_ALREADY_OPEN	Module is already opened.
NFC_ERR_INVALID_DATA	NFC SDK initialization data was incomplete

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Initialize NFC IoT Reader.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
const nfc_reader_ptx_cfg_t p_cfg;  
err = R_NFC_PTX_Open(&p_ctrl, &p_cfg);
```

Special Notes:

None

3.2. R_NFC_PTX_StartDiscovery()

This function initiates discovery of NFC tags.

Format

```
nfc_err_t R_NFC_PTX_StartDiscovery(  
    nfc_reader_ptx_ctrl_t * const p_ctrl  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the idle state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Initiates discovery of NFC tags.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t        p_ctrl;  
nfc_err_t                    err;  
err = R_NFC_PTX_StartDiscovery(&p_ctrl);
```

Special Notes:

None

3.3. R_NFC_PTX_GetStatus()

This function gets the status information on the provided status identifier (system, discovery, etc.). Used during polling to confirm a card was discovered

Format

```
nfc_err_t R_NFC_PTX_GetStatus(
    nfc_reader_ptx_ctrl_t  const p_ctrl,
    ptxIoTRd_StatusType_t  status_type,
    uint8_t * const      p_status
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
status_type	Status identifier for target info
p_status	Pointer to location to store NFC chip status flag

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the polling state
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Gets the status information on the provided status identifier.

Uses during polling to confirm a card was discovered.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t  p_ctrl;
ptxIoTRd_StatusType_t  status_type = StatusType_Discover;
uint8_t                p_status = PTX_SYSTEM_STATUS_OK;
nfc_err_t              err;
err = R_NFC_PTX_GetStatus(&p_ctrl, status_type, &p_status);
```

Special Notes:

None

3.4. R_NFC_PTX_GetCardRegistry()

This function gets the internal card registry.

Format

```
nfc_err_t R_NFC_PTX_GetCardRegistry(  
    nfc_reader_ptx_ctrl_t * const p_ctrl,  
    ptxIoTRd_CardRegistry_t      pp_card_registry  
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
pp_card_registry	Pointer to card registry for discovered cards

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Gets the status information on the provided status identifier.

Uses during polling to confirm a card was discovered.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t      p_ctrl;  
ptxIoTRd_CardRegistry_t    *card_registry = NULL;  
nfc_err_t                  err;  
err = R_NFC_PTX_GetCardRegistry(&p_ctrl, &card_registry);
```

Special Notes:

None

3.5. R_NFC_PTX_ActivateCard()

This function gets the details of the discovered card and connects to it.

Format

```
nfc_err_t R_NFC_PTX_ActivateCard(  
    nfc_reader_ptx_ctrl_t * const p_ctrl,  
    ptxIoTRd_CardParams_t * const p_card_params,  
    ptxIoTRd_CardProtocol_t    protocol  
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_card_params	Parameters of the card to be activated
protocol	The NFC protocol of the card to be activated

Return value

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Connects and gets the details of the discovered card

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
ptxIoTRd_CardRegistry_t *card_registry = NULL;  
ptxIoTRd_CardProtocol_t prot = Prot_T2T;  
nfc_err_t                err;  
err = R_NFC_PTX_ActivateCard(&p_ctrl, &card_registry->Cards[0], prot);
```

Special Notes:

None

3.6. R_NFC_PTX_DataExchange()

This function sends and receives data to/from the activated card

Format

```
nfc_err_t R_NFC_PTX_DataExchange(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl,  
    nfc_reader_ptx_data_info_t * const p_data_info  
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Sends and receives data to/from the activated card.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t      p_ctrl;
uint8_t q_nfc_tx_buf[100] = {0};
uint8_t q_nfc_rx_buf[100] = {0};
nfc_reader_ptx_data_info_t nfc_buf =
{
    .p_tx_buf  = nfc_tx_buf,
    .tx_length = sizeof(nfc_tx_buf),
    .p_rx_buf  = nfc_rx_buf,
    .rx_length = sizeof(nfc_rx_buf),
};
err = R_NFC_PTX_DataExchange(&p_ctrl, &nfc_buf);
```

Special Notes:

None

3.7. R_NFC_PTX_ReaderDeactivation()

This function deactivates the activated card to return to the desired state (idle, discovery, etc.).

Format

```
nfc_err_t R_NFC_PTX_ReaderDeactivation(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl,  
    nfc_reader_ptx_return_state_t    return_state  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure
return_state Expectation for end state of NFC state machine

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Deactivates the activated card to return to the desired state.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
nfc_reader_ptx_return_state_t    return_state = NFC_PTX_RETURN_IDLE;  
err = R_NFC_PTX_ReaderDeactivation(&p_ctrl, return_state);
```

Special Notes:

None

3.8. R_NFC_PTX_SWReset()

This function resets system state, card registry and IoT Reader instance.

Format

```
nfc_err_t R_NFC_PTX_SWReset(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the idle/polling state
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a soft-reset of the PTX chip.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
err = R_NFC_PTX_SWReset(&p_ctrl);
```

Special Notes:

None

3.9. R_NFC_PTX_PowerModeSet()

This function puts the device into either Active (all features) or Stand-by mode (most features off), to cut down energy consumption.

Format

```
nfc_err_t R_NFC_PTX_PowerModeSet(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl,  
    nfc_reader_ptx_power_mode_t    power_mode  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure
power_mode Flag to indicate if setting to Active or Stand-by mode

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the idle/polling state.
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to NFC Status function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Puts the device into either Active (all features) or Stand-by mode (most features off), to cut down energy consumption. RF communication is not possible from Stand-by mode, and Wake-up is possible only via communication interface. This function is only available exclusively before starting discovery.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
nfc_reader_ptx_power_mode_t power_mode = NFC_PTX_ACTIVE_MODE;  
err = R_NFC_PTX_PowerModeSet (&p_ctrl, power_mode);
```

Special Notes:

None

3.10. R_NFC_PTX_NDEF_Init()

This function initializes the optional NDEF API add-on

Format

```
nfc_err_t R_NFC_PTX_NDEF_Init(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl,  
    nfc_reader_ptx_data_info_t * const p_data_info  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure

p_data_info Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_ALREADY_OPEN	NDEF API add-on was already initialized
NFC_ERR_UNSUPPORTED	NDEF API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Initializes the optional NDEF API add-on. Must be done after NFC stack is initialized.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t      p_ctrl;
uint8_t q_nfc_tx_buf[100] = {0};
uint8_t q_nfc_rx_buf[100] = {0};
nfc_reader_ptx_data_info_t nfc_buf =
{
    .p_tx_buf  = nfc_tx_buf,
    .tx_length = sizeof(nfc_tx_buf),
    .p_rx_buf  = nfc_rx_buf,
    .rx_length = sizeof(nfc_rx_buf),
};
err = R_NFC_PTX_NDEF_Init(&p_ctrl, &nfc_buf);
```

Special Notes:

None

3.11. R_NFC_PTX_NDEF_DataExchange()

This function exchanges NDEF data with an NFC tag by reading from or writing to it

Format

```
nfc_err_t R_NFC_PTX_NDEF_DataExchange(
    nfc_reader_ptx_ctrl_t *          const p_ctrl,
    uint8_t *                       p_ndef_message,
    uint32_t                         message_length,
    nfc_reader_ptx_ndef_read_write_t ndef_read_write_flag
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_ndef_message	Pointer to the buffer to transmit or receive a message
message_length	Length of the NDEF message buffer
ndef_read_write_flag	Flag to control if the data exchange is a read or a write

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_NOT_INITIALIZED	NDEF add-on is not initialized yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_UNSUPPORTED	NDEF API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Exchanges NDEF data with an NFC tag by reading from or writing to it.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;
uint8_t ndef_msg_buffer[NFC_CFG_NDEF_WORK_BUF_SIZE] = {0};
err = R_NFC_PTX_NDEF_DataExchange(&p_ctrl, ndef_msg_buffer,
sizeof(ndef_msg_buffer), NFC_PTX_NDEF_READ);
```

Special Notes:

None

3.12. R_NFC_PTX_NDEF_Lock()

This function locks an NFC tag

Format

```
nfc_err_t R_NFC_PTX_NDEF_Lock(  
nfc_reader_ptx_ctrl_t * const      p_ctrl  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_NOT_INITIALIZED	NDEF add-on is not initialized yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_UNSUPPORTED	NDEF API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Exchanges NDEF data with an NFC tag by reading from or writing to it.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t      p_ctrl;  
err = R_NFC_PTX_NDEF_Lock(&p_ctrl);
```

Special Notes:

None

3.13. R_NFC_PTX_Native_Tag_Init()

Initializes the optional Native-Tag API add-on. Must be done after NFC stack is initialized.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Init(  
nfc_reader_ptx_ctrl_t * const      p_ctrl,  
nfc_reader_ptx_data_info_t * const p_data_info,  
uint8_t                           tag_select  
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_data_info	Pointer to the NFC TX/RX data.
tag_select	User-provided bit mask of tags to be initialized.

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_ALREADY_OPEN	Native-Tag API add-on was already initialized
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Initializes the optional Native-Tag API add-on. Must be done after NFC stack is initialized. For each tag type to be initialized, the tag type bit values must be set in the tag select parameter.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t nfc_reader_ptx0_ctrl;
nfc_reader_ptx_data_info_t nfc_send_buf =
{
    .p_tx_buf = nfc_tx_buf,
    .tx_length = sizeof(nfc_tx_buf),
    .p_rx_buf = nfc_rx_buf,
    .rx_length = sizeof(nfc_rx_buf),
};
uint8_t select_tag_mask = NFC_READER_PTX_NATIVE_TAG_TYPE_T2T |
                          NFC_READER_PTX_NATIVE_TAG_TYPE_T3T |
                          NFC_READER_PTX_NATIVE_TAG_TYPE_T4T |
                          NFC_READER_PTX_NATIVE_TAG_TYPE_T5T;

err = R_NFC_PTX_Native_Tag_Init(&nfc_reader_ptx0_ctrl, &nfc_send_buf,
select_tag_mask);
```

Special Notes:

None

3.14. R_NFC_PTX_Native_Tag_Read()

Read operation using the Native-Tag API add-on.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Read (
nfc_reader_ptx_ctrl_t * const           p_ctrl,
nfc_reader_ptx_native_tag_read_params_t * const   p_read_params,
nfc_reader_ptx_native_tag_type_t       tag_type,
nfc_reader_ptx_data_info_t * const     p_data_info
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_read_params	Pointer to optional parameter structure for read operation
tag_type	The target tag type for the operation
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_INVALID_MODE	One of the available tag types was not selected
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a read operation using the Native-Tag API add-on. Optional configurations must be set for the required tag type based on the command to be used.

Reentrant

No

Example

None

Special Notes:

None

3.15. R_NFC_PTX_Native_Tag_Write()

Write operation using the Native-Tag API add-on.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Write (
nfc_reader_ptx_ctrl_t * const           p_ctrl,
nfc_reader_ptx_native_tag_write_params_t * const  p_write_params,
nfc_reader_ptx_native_tag_type_t       tag_type,
nfc_reader_ptx_data_info_t * const      p_data_info
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_write_params	Pointer to optional parameter structure for read operation
tag_type	The target tag type for the operation
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_INVALID_MODE	One of the available tag types was not selected
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a write operation using the Native-Tag API add-on.

Reentrant

No

Example

None

Special Notes:

None

3.16. R_NFC_PTX_Native_Tag_Select()

Select operation using the Native-Tag API add-on, which is different for each tag type.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Select (
nfc_reader_ptx_ctrl_t * const                p_ctrl,
nfc_reader_ptx_native_tag_select_params_t * const  p_select_params,
nfc_reader_ptx_native_tag_type_t            tag_type,
nfc_reader_ptx_data_info_t * const          p_data_info
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_select_params	Pointer to optional parameter structure for read operation
tag_type	The target tag type for the operation
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_INVALID_MODE	One of the available tag types was not selected
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a Select operation using the Native-Tag API add-on, which is different for each tag type.

- For Type 2 tags, this selects a different sector.
- For Type 4 tags, it selects a different File and/or Application on the tag.
- For Type 5 tags, this sets a tag as selected, allowing a reader to communicate with a specific tag without its UID.

Reentrant

No

Example

None

Special Notes:

None

3.17. R_NFC_PTX_Native_Tag_Lock()

Lock operation using the Native-Tag API add-on.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Lock (
nfc_reader_ptx_ctrl_t * const                p_ctrl,
nfc_reader_ptx_native_tag_lock_params_t *   const p_lock_params,
nfc_reader_ptx_native_tag_type_t           tag_type,
nfc_reader_ptx_data_info_t * const         p_data_info
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_lock_params	Pointer to optional parameter structure for read operation
tag_type	The target tag type for the operation
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_DATA	Problem with input parameters or data transmission
NFC_ERR_INVALID_MODE	One of the available tag types was not selected
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a Lock operation using the Native-Tag API add-on. This function is exclusive to Type 5 tags.

Reentrant

No

Example

None

Special Notes:

None

3.18. R_NFC_PTX_Native_Tag_Sleep()

Sleep operation using the Native-Tag API add-on.

Format

```
nfc_err_t R_NFC_PTX_Native_Tag_Sleep (
nfc_reader_ptx_ctrl_t * const          p_ctrl,
nfc_reader_ptx_native_tag_sleep_params_t * const  p_sleep_params,
nfc_reader_ptx_native_tag_type_t      tag_type,
nfc_reader_ptx_data_info_t * const     p_data_info
)
```

Parameters

p_ctrl	Pointer to NFC IoT Reader control structure
p_sleep_params	Pointer to optional parameter structure for read operation
tag_type	The target tag type for the operation
p_data_info	Pointer to the NFC TX/RX data

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_ASSERTION	Assertion error occurred
NFC_ERR_NOT_OPEN	Module is not opened yet
NFC_ERR_INVALID_STATE	NFC module is not in the activated state
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to the function
NFC_ERR_UNSUPPORTED	Native-Tag API add-on was not included in project configuration

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Performs a Sleep operation using the Native-Tag API add-on. This function is exclusive to Type 5 tags.

Reentrant

No

Example

None

Special Notes:

None

3.19. R_NFC_PTX_Close()

This function closes the NFC module and resets all variables.

Format

```
nfc_err_t R_NFC_PTX_Close(  
    nfc_reader_ptx_ctrl_t * const    p_ctrl  
)
```

Parameters

p_ctrl Pointer to NFC IoT Reader control structure

Return values

NFC_SUCCESS	Function completed successfully
NFC_ERR_INVALID_ARGUMENT	Invalid input parameters to NFC Status function

Properties

Prototype declarations are contained in r_nfc_ptx_if.h.

Description

Closes the FSP NFC module and resets all variables.

Reentrant

No

Example

```
nfc_reader_ptx_ctrl_t    p_ctrl;  
err = R_NFC_PTX_Close(&p_ctrl);
```

Special Notes:

None

4. Demo Projects

Demo projects include function main() that utilizes the FIT module and its dependent modules (e.g. r_bsp). This FIT module includes the following demo projects.

4.1. NFC PTX105RQC Get Card Information Demo Project

4.1.1 Prerequisites

- Hardware requirements:
 - EK-RX261: Renesas Evaluation Kit for RX261 v1 (Product no.: RTK5EK2610S00001BE)
 - PTX105R: PTX105R PMOD™ Board for IoT as NFC module
 - PC running Windows 11.
 - Micro-USB cables for Debugging and the serial log output (included as part of the kit. See EK-RX261 v1 – User's Manual at "Related Documents" on page 1).
- Software requirements for Windows 11 PC
 - IDE: e2 studio 2025-10 or later.
 - Compiler: Renesas Electronics C/C++ Compiler for RX Family V3.07.00.
 - Tera Term v4.106 or later.

4.1.2 Import the Demo Project

User can import the demo project by adding the demo to their e² studio workspace (see section 4.2 Adding a Demo to a Workspace or by downloading the demo project (see section 4.3 Downloading Demo Projects).

4.1.3 Hardware Setup

- Connect the NFC PTX105RQC PMOD module to the EK-RX261 **PMOD1** connector.
- Connect a micro-USB cable from the PC to the EK-RX261 micro-USB connector (**J26**) for power supply.
- Connect a micro-USB cable from the PC to the EK-RX261 micro-USB connector (**J16**) for serial log output.
- Open **J24** to enable E2 on-board debugger circuit.
- Short **pins 1-2** of **J31** to enable SCK pin of PMOD SPI.
- Short **pins 1-2** of **J32** to enable MISO pin of PMOD SPI.

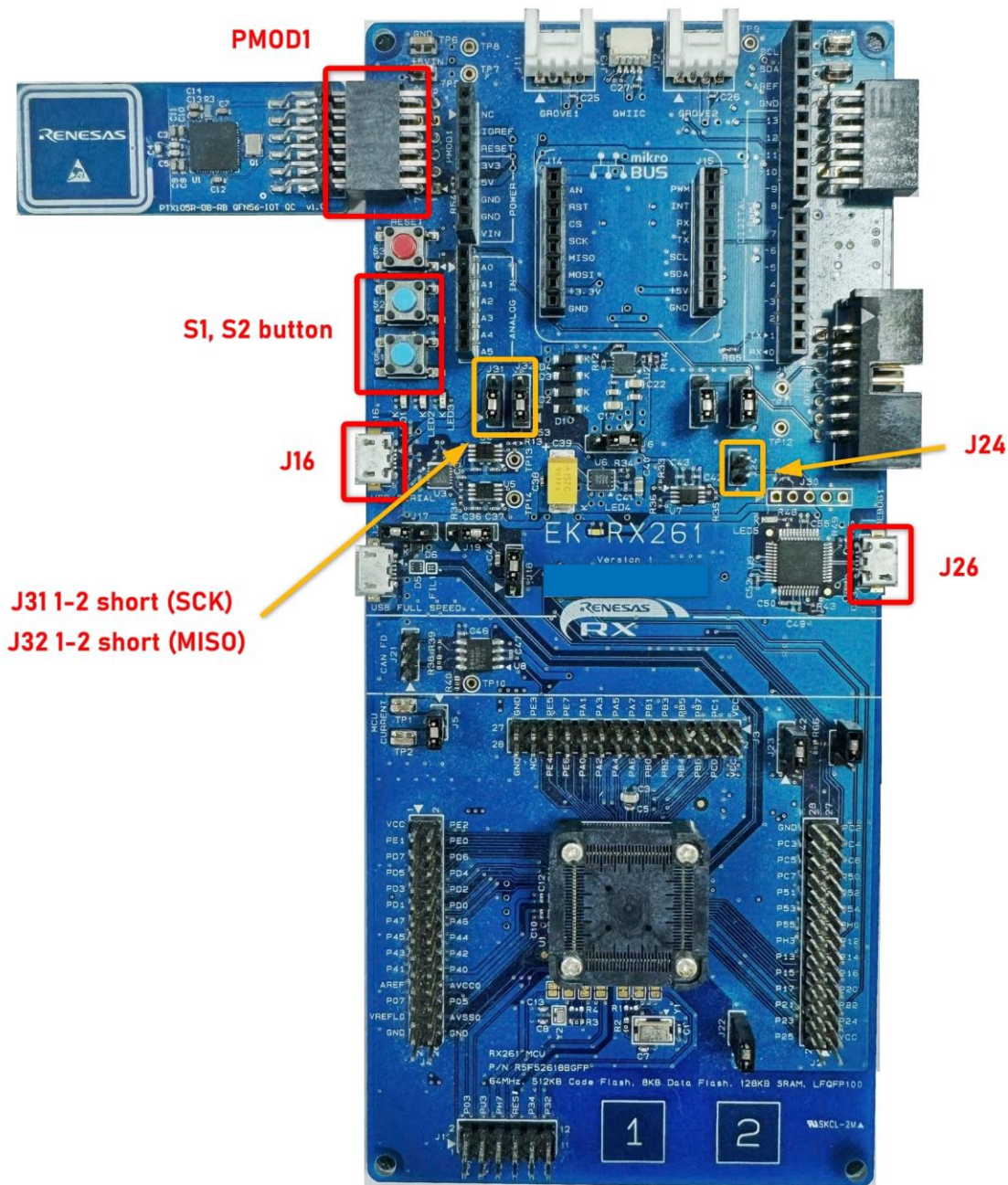


Figure 4.1 Hardware Setup

4.1.4 How to Run the Demo

- a) Configure Basic Settings required to discover, activate, and exchange data with an NFC tag. Open the Smart Configurator as shown in the image below and change essential parameters.

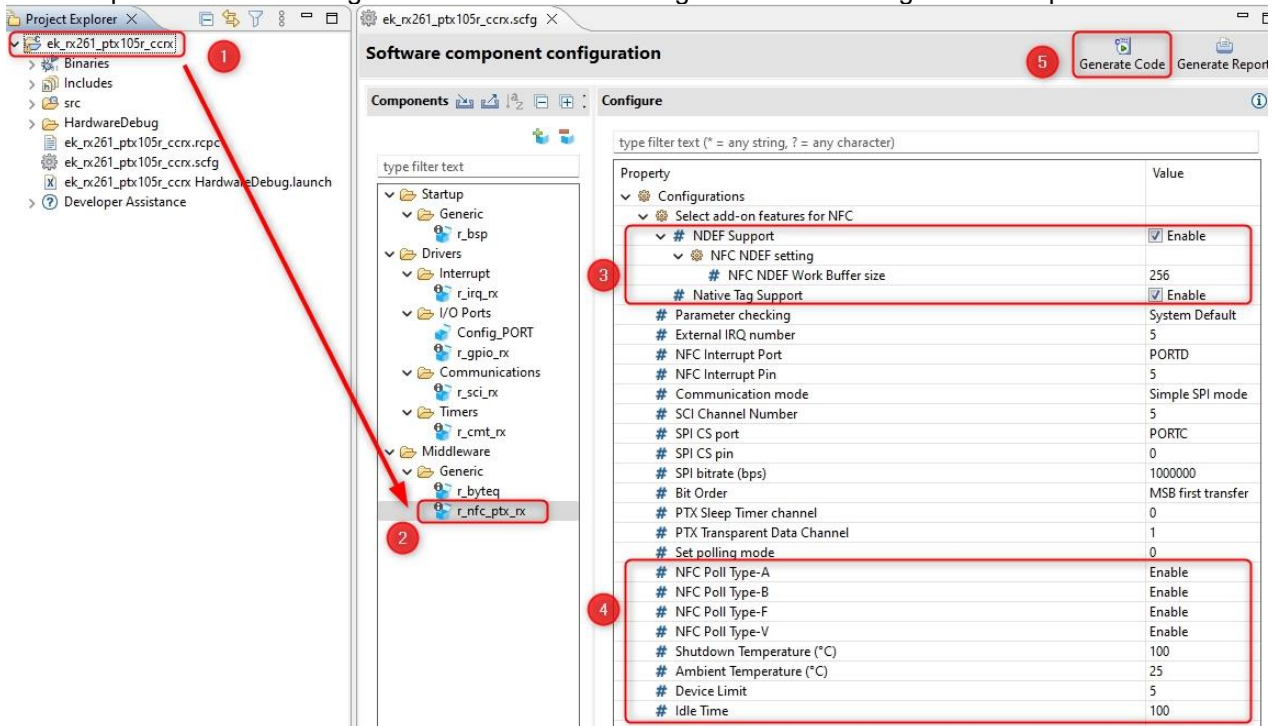


Figure 4.1 NFC Settings

- Enable “NFC_CFG_NDEF_SUPPORT” and set “NFC_CFG_NDEF_WORK_BUF_SIZE” to 256
- Enable “NFC_CFG_NATIVE_TAG_SUPPORT”
- “NFC_CFG_POLL_TYPE_A”: Select to discover NFC Type-A tags
- “NFC_CFG_POLL_TYPE_B”: Select to discover NFC Type-B tags
- “NFC_CFG_POLL_TYPE_F”: Select to discover NFC Type-F tags
- “NFC_CFG_POLL_TYPE_V”: Select to discover NFC Type-V tags
(**Note:** Please select at least one polling type for proper card)
- “NFC_CFG_TEMP_SENSOR_SHUTDOWN”: Expected thermal shutdown threshold value (unit: Celsius).
- “NFC_CFG_TEMP_SENSOR_AMBIENT”: Set the ambient temperature value at which calibration takes place (unit: Celsius).
- “NFC_CFG_DEVICE_LIMIT”: Select the max number of tags to discover (maximum number is 49).
- “NFC_CFG_IDLE_TIME_MS”: Select idle time between polling cycles for RF-Discovery loop (unit: ms)

- b) Building the Demo Project

Build the project and confirm no build errors occur.

```

Renesas Optimizing Linker Completed
Loading input file ek_rx261_ptx105r_ccrx.abs
Parsing the ELF input file....
25 segments required LMA fixes
Converting the DWARF information....
Constructing the output ELF image....
Saving the ELF output file ek_rx261_ptx105r_ccrx.x
Invoking Converter: ek_rx261_ptx105r_ccrx.mot

```

```

Renesas Optimizing Linker Completed
Build complete.

```

```

18:24:22 Build Finished. 0 errors, 0 warnings. (took 47s.331ms)

```

Figure 4.3 Confirm the Demo Project Build

- c) Go to **Run → Debug Configurations...**, double-click on **Renesas GDB Hardware Debugging**, and set up the debugger.
- d) Check whether exact **.x** file is selected (**.x** file is generated in HardwareDebug folder after build succeeded) same as figure below.

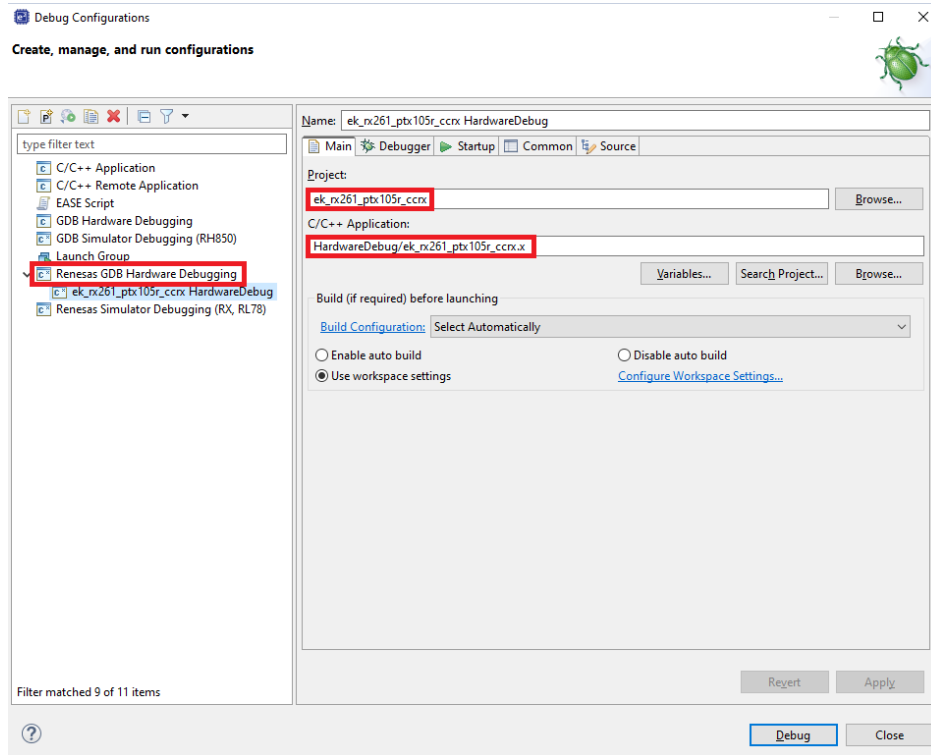


Figure 4.4 Set Up the Debug File

- e) Check “Debug hardware” and “Target Device”, ensure them are same as below figure by going to Renesas GDB Hardware Debugging->Debugger

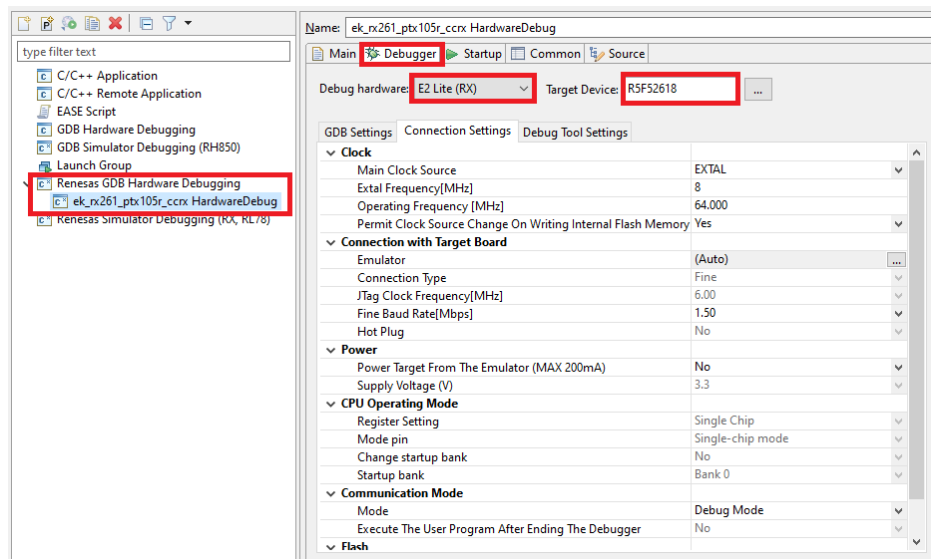


Figure 4.5 Set Up the Debugger

- f) Click **Debug** to flash the application on the board.
- g) When completed, click **Resume** twice to start the application.

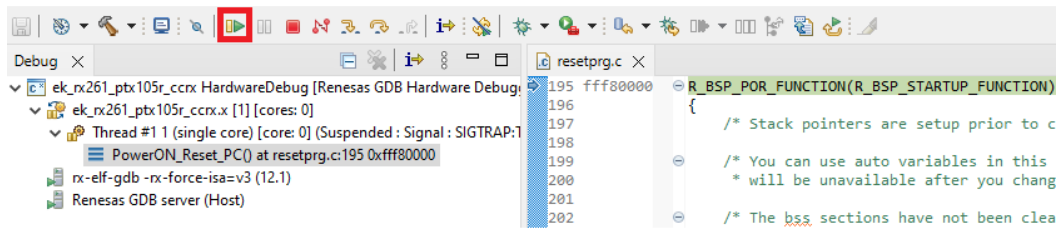


Figure 4.6 Start the Application

To view the application output, open the Tera Term set up below settings

- h) Click **Setup > Terminal...**, select **New line: Receive** as **AUTO** and **Transmit** as **CR+LF** then tick **Local echo**.

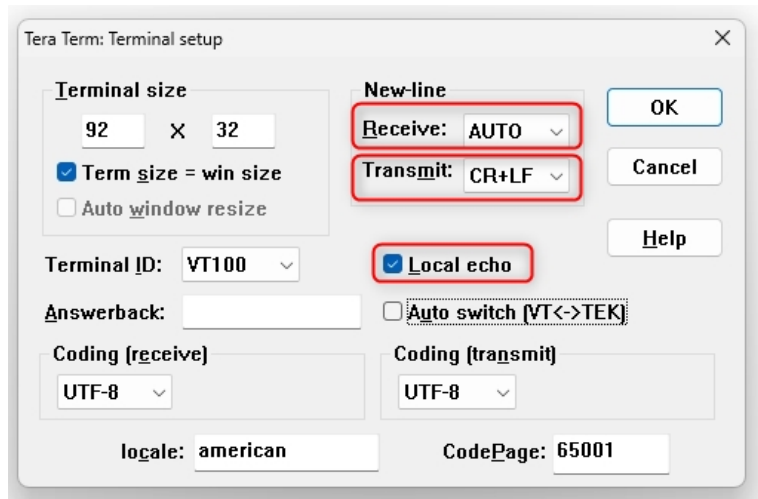


Figure 4.7 Terminal Setup

- i) Click **Setup > Serial port...** and ensure that the speed is set to **115200**.

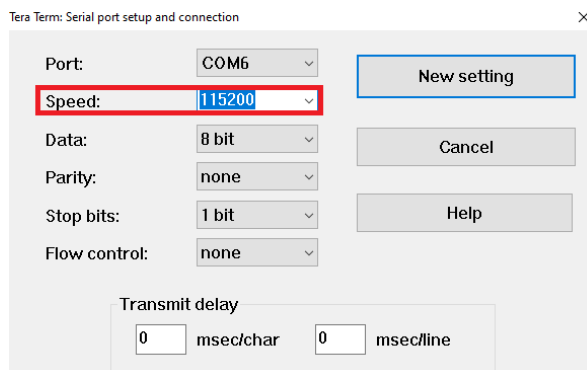


Figure 4.8 Terminal Setup for USB Serial Port

- j) After successful initialization, the system defaults to STANDBY mode where most features are disabled for power conservation. Press button **S2** on board to transition to *ACTIVATED* mode.

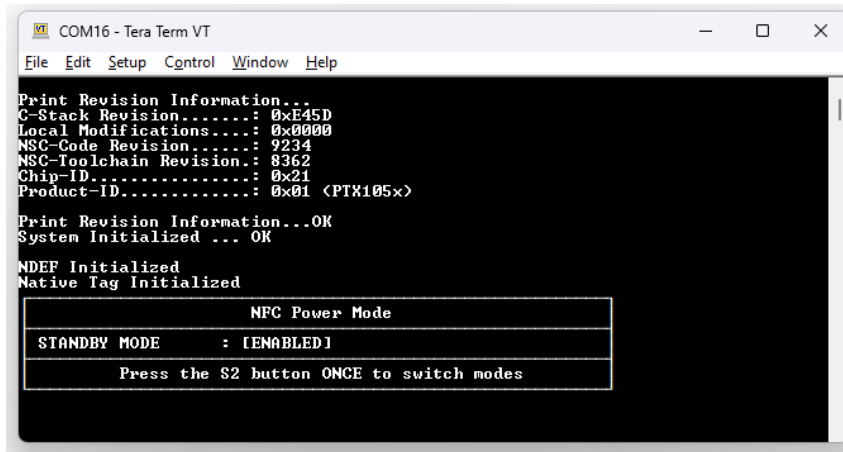


Figure 4.9 Power mode set

- k) Once **S2** is pressed, *ACTIVATED* mode is enabled. Now it automatically runs in Data Exchange & Read NDEF/Native Tag mode, the serial terminal shows “*Waiting for discovered Cards or external Fields ...*”, the user can place a card in the field and view the data displayed on the terminal.

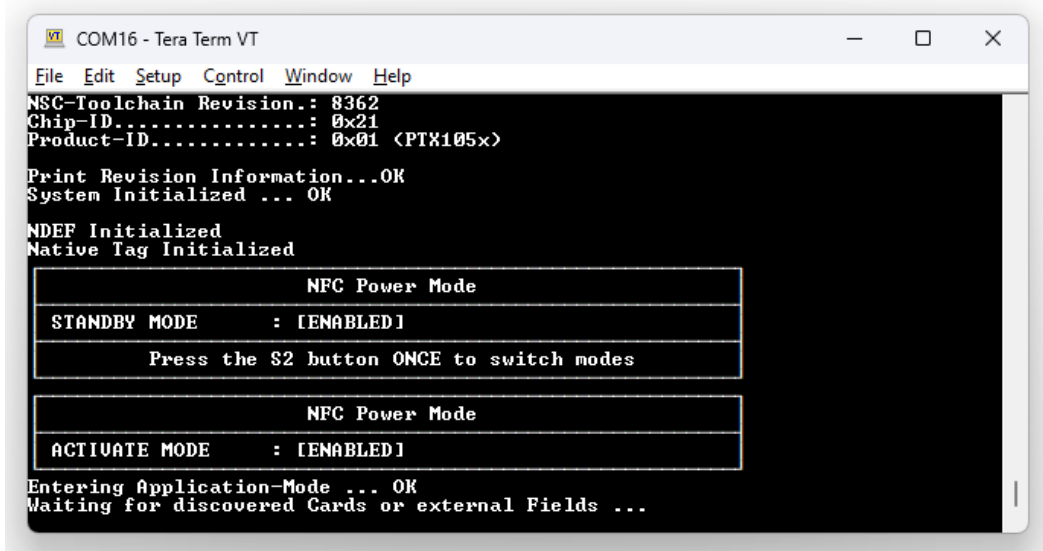


Figure 4.10 Data Exchange & Read NDEF/Native Tag mode

- 1) *Write NDEF mode*: After the *Note Instruction* board appears as shown below, **long press** the S2 button until LED2 turns on to enter *WRITE NDEF mode*.

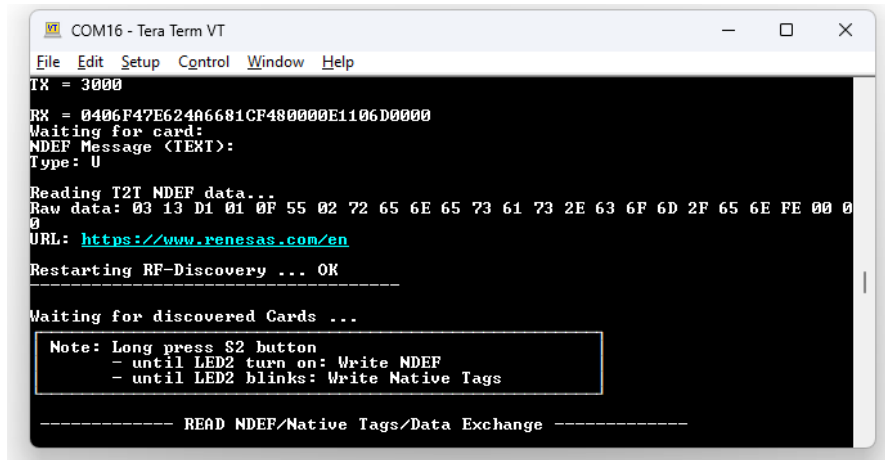


Figure 4.11 Write NDEF mode

Enter the message the user want to store on the NFC card. Position the card in proximity to the NFC chip to ensure proper data writing smoothly. Once NDEF has been successfully written to the tag, it will automatically switch to Data Exchange & Read NDEF/Native Tag mode.

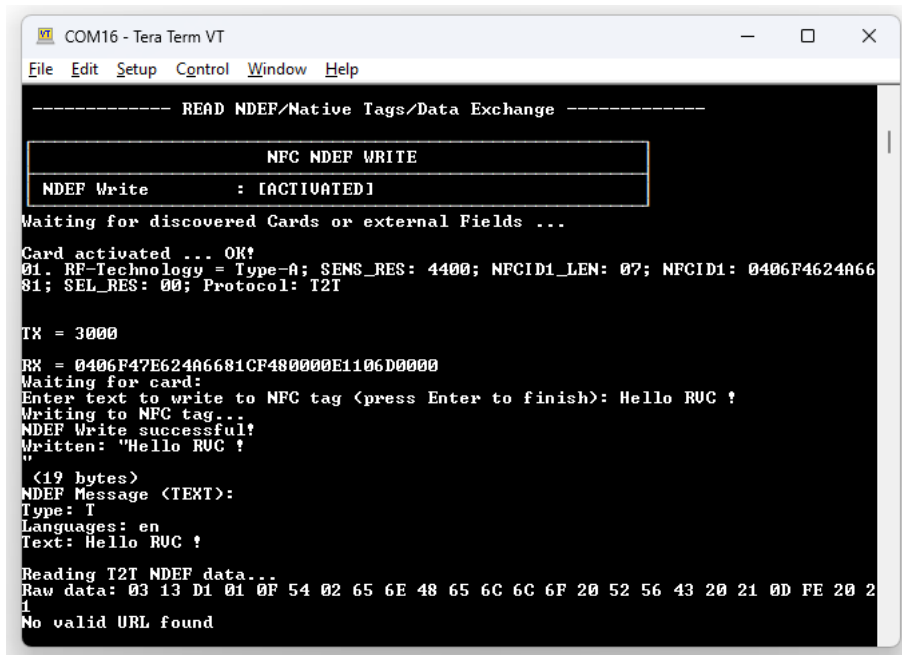


Figure 4.12 Write & Read NDEF mode

- m) *Read/Write Native Tag add-on*: With Native Tag features, this demo only shows the URL write and URL read functionality through the Native Tag Read/Write API. When the *Note Instruction* board appears as shown in the figure below, **long press** the S2 button until LED2 on the board blinks.

```
Waiting for discovered Cards ...
Note: Long press S2 button
- until LED2 turn on: Write NDEF
- until LED2 blinks: Write Native Tags
```

Figure 4.13 Write Native Tag mode

Next, bring the card close to the reading area to proceed with writing URL to the card.

```
COM16 - Tera Term VT
File Edit Setup Control Window Help
NFC Native Tag WRITE
Native Tag Write : [ACTIVATED]
Waiting for discovered Cards or external Fields ...
Card activated ... OK!
01. RF-Technology = Type-A; SENS_RES: 4400; NFCID1_LEN: 07; NFCID1: 0406F4624A6681; SEL_RES: 00; Protocol: T2T
TX = 3000
RX = 0406F47E624A6681CF480000E1106D0000
Waiting for card:
NDEF Message <TEXT>:
Type: U
Reading T2T NDEF data...
Raw data: 03 13 D1 01 0F 55 02 72 65 6E 65 73 61 73 2E 63 6F 6D 2F 65 6E FE 00 00
URL: https://www.renesas.com/en
Restarting RF-Discovery ... OK
```

Figure 4.14 Write & Read Native Tag mode

Note:

1. This Demo was tested with 3 types of cards: T2T, T4T, and T5T.
2. The demo also supports selection from multiple cards by placing multiple cards over the PTX. Program will print out generic information for each card and only perform data exchange for the card that was first detected.

```
Waiting for discovered Cards ...
Multiple Card(s) detected - resolving ... ?
Multiple Card(s) detected - resolving ... ?
Multiple Card(s) detected - resolved ... OK!
01. RF-Technology = Type-A; SENS_RES: 0400; NFCID1_LEN: 04; NFCID1: C1049552; SEL_RES: 20
02. RF-Technology = Type-A; SENS_RES: 0400; NFCID1_LEN: 04; NFCID1: 04E3D58C; SEL_RES: 08
Selecting first detected card/protocol (RF-Protocol == ISO-DEP)... ... OK!
01. RF-Technology = Type-A; SENS_RES: 0400; NFCID1_LEN: 04; NFCID1: C1049552; SEL_RES: 20; Protocol: ...: ISO-DEP; PPS1: 00; A1S: 0B78071024B4F4E412320
TX = 00A404000E325041592E5359532E444446303100
RX = 6F30840E325041592E5359532E4444463031A51EBF0C1B61194F07A000000727101050084E415041532044656269748701019000
```

Figure 4.15 Multiple Card Selection Handling

4.1.5 Understanding NFC Data Fields in the Example

In this section, the data exchange example shown in Figure 4.9 will be explained.

Sample NFC data

```
01. RF-Technology = Type-A; SENS_RES: 0400; NFCID1_LEN: 04; NFCID1: C1049552; SEL_RES: 20; Protocol....: ISO-DEP; PPS1: 00; ATS:
0B788071024B4F4E412320
TX = 00A404000E325041592E5359532E444446303100
RX = 6F30840E325041592E5359532E4444463031A51EBF0C1B61194F07A000007271010500B4E415041532044656269748701019000
```

The above data exchange example is generated when a bank card using RF Technology Type A with the ISO-DEP protocol is placed in the field.

Polling/activation fields

Field	Value (Hex)	Description
RF Technology	Type-A	Indicates this is an ISO/IEC 14443 Type A card
SENS_RES (ATQA)	0x0400	ATQA (Answer to Request) to REQA (Request command of reader). Indicates card's UID (unique identifier) type is Single-length UID (4-byte UID)
NFCID1_LEN	0x04	Length of NFCID1 (UID) 0x04 means the UID is 4 bytes long.
NFCID1	0x C1049552	4-byte unique identifier (UID) of card
SEL_RES (SAK)	0x20	Select Response (also known as SAK – Select Acknowledge). This 1-byte response comes after the reader selects the card's UID. Confirms the card is now in an active state. A value of "0x20" means: ISO/IEC 14443-4 compliant card (supports ISO-DEP).
Protocol	ISO-DEP	Indicates the high-level protocol used is ISO-DEP (ISO Data Exchange Protocol).
PPS1	0x00	Protocol and Parameter Selection (PPS) byte. After the ATS (see below), the reader can send a PPS to adjust communication parameters (such as bitrate). 0x00 means no parameter changes were requested.
ATS	0x0B788071024B4F4E412320	The Answer to Select (ATS) from the card. The ATS provides the card's communication capabilities for the ISO-DEP layer (such as: ATS bytes length, maximum frame size, bitrate, timeout, manufacturer info, etc).

Table 4.11 General Card Parameters

APDU Command/Response (TX and RX) Explained**TX (Transmit)****00A404000E325041592E5359532E444446303100**

This is an APDU command used to select an application (AID - Application Identifier) on a smart card.

Byte(s)	Meaning
00	CLA (Class Byte) – Standard ISO 7816 command
A4	INS (Instruction) – A4 means SELECT an application or file on the card
04	P1 (Parameter 1) – Select by name (AID)
00	P2 (Parameter 2) – First occurrence
0E	Lc (Length of AID) – 14 bytes
325041592E5359532E4444463031	Data (AID) – "2PAY.SYS.DDF01" (directory name of payment cards, often called PSE – Payment System Environment)
00	Indicates not specifying an expected length for the RX response

Table 4.12 Select Proximity Payment System Environment (PPSE)**RX (Response)****6F30840E325041592E5359532E4444463031A51EBF0C1B61194F07A0000007271010500B4E415041532044656269748701019000**

Decoding the response message:

Byte(s)	Meaning
6F 30	FCI Template (TLV structure, 48 bytes)
84 0E 325041592E5359532E4444463031	Dedicated File (DF) Name – "2PAY.SYS.DDF01"
A5 1E	Proprietary Information
BF0C 1B	FCI Issuer Discretionary Data (27 bytes)
61 19	Application Template (25 bytes)
4F 07 A0000007271010	Application Identifier (AID) – A0000007271010
50 0B 4E41504153204465626974	Application Label – "NAPAS Debit"
87 01 01	Application Priority Indicator (Priority = 1)
9000	Status Word – Success

Table 4.13 Select Proximity Payment System Environment (PPSE)

4.2. Adding a Demo to a Workspace

Demo projects are found in the FITDemos subdirectory of the distribution file for this application note. To add a demo project to a workspace, select *File >> Import >> General >> Existing Projects into Workspace*, then click "Next". From the Import Projects dialog, choose the "Select archive file" radio button. "Browse" to the FITDemos subdirectory, select the desired demo zip file, then click "Finish".

4.3. Downloading Demo Projects

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on this application note and select "Sample Code (download)" from the context menu in the *Smart Brower >> Application Notes* tab.

5. Appendices

5.1 Confirmed Operation Environment

This section describes confirmed operation environment for the PTX NFC FIT module.

Table 5.5.1 Confirmed Operation Environment (Ver. 1.00)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio 2025.04
C compiler	Renesas Electronics C/C++ Compiler for RX Family V3.07.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Big endian / little endian
Revision of the module	Rev.1.00
Board used	Renesas EK-RX261 Evaluation Kit (Product no.: RTK5EK2610S00001BE)

Table 5.5.2 Confirmed Operation Environment (Ver. 1.10)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio 2025.07
C compiler	Renesas Electronics C/C++ Compiler for RX Family V3.07.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Big endian / little endian
Revision of the module	Rev.1.10
Board used	Renesas EK-RX261 Evaluation Kit (Product no.: RTK5EK2610S00001BE)

Table 5.5.3 Confirmed Operation Environment (Ver. 1.20)

Item	Contents
Integrated development environment	Renesas Electronics e2 studio 2025.10
C compiler	Renesas Electronics C/C++ Compiler for RX Family V3.07.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian order	Big endian / little endian
Revision of the module	Rev.1.20
Board used	Renesas EK-RX261 Evaluation Kit (Product no.: RTK5EK2610S00001BE)

5.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e² studio:

Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"
When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

A: The setting in the file "r_nfc_ptx_rx_config.h" may be wrong. Check the file "r_nfc_ptx_rx_config.h". If there is a wrong setting, set the correct value for that. Refer to 2.7 Configuration Overview for details.

6. Reference Documents

User's Manual: Hardware

The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

RX Family CC-RX Compiler User's Manual (R20UT3248)

The latest versions can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/contact/>

Revision History

Rev.	Date	Revision History	
		Page	Summary
1.00	Jun. 27, 2025	-	First edition issued
1.10	Nov 17, 2025	-	Change format layout
		7	Updated table 1.3 API Function for new APIs and description
		8	Updated Figure 1.3 with new features
		10 - 11	Updated Table 2.1 Configuration Options with new features
		13	Updated Table 2.3 Memory Sizes for RX261
		14 - 15	Update Section 2.9 Parameters for new features Add Table 2.8 Member in Structure for NDEF data exchange flags Add Table 2.9 Member in Structure for NFC power mode flags
		16	Update Table 2.10 API Error Codes with NFC_ERR_UNSUPPORTED
		18	Added new section 2.13 Usage Notes
		28	Added 3.10 R_NFC_PTX_PowerModeSet()
		29	Added 3.11. R_NFC_PTX_NDEF_Init()
		30 - 31	Added 3.12. R_NFC_PTX_NDEF_DataExchange()
		32	Added 3.13. R_NFC_PTX_NDEF_Lock()
		34	Change image Section 4.1.3 Hardware Setup
		37 - 38	Section 4.1.15 Update how to run the demo
43	Added Table 5.5.2 Confirmed Operation Environment (Ver. 1.10)		
1.20	Jan 22, 2026	-	Edit format layout
		-	Included spdx file in the package
		-	Removed Section 2.13 Usage Notes (Limitations)
		4	1.2 Overview of the PTX NFC FIT Module - Updated list features support
		7	1.3 API Overview - Updated table API (Added new function Native Tag)
		8	Update figure 1.3 – Status Transitions
		10	Table 0.1 Configuration Options (r_nfc_ptx_rx_config.h) - Added new macro NFC_CFG_NATIVE_TAG_SUPPORT
		13	Section 2.8 – Updated code size
		15 - 18	Section 2.9 Parameters - Table 0.2 Updated Member in Structure - Created new table 2.10, 2.11, 2.12, 2.13, 2.14, 2.15
		19	Section 2.10 Return Values - Added new macro error code: NFC_ERR_INVALID_MODE
		37 - 45	Section 3. API Functions - Created new fuctions which is listed below: <ul style="list-style-type: none"> • R_NFC_PTX_Native_Tag_Init() • R_NFC_PTX_Native_Tag_Read() • R_NFC_PTX_Native_Tag_Write() • R_NFC_PTX_Native_Tag_Select() • R_NFC_PTX_Native_Tag_Lock() • R_NFC_PTX_Native_Tag_Sleep()
		49, 52 – 54	Section 4. Demo Projects - 4.1.4 How to run demo – Guide user enable NFC_CFG_NATIVE_TAG_SUPPORT - Update Read/Write with Native Tag Add-on
		58	Section 5.1 Confirmed Operation Environment - Added new table 5.5.3 – Confirmed Operation Environment (Ver. 1.20)

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
 - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
 - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
- Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
 11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.