Application note

ST25R embedded NFC library RFAL examples

Introduction

The RFAL library is part of the ST25 NFC embedded library package (STSW-ST25R-LIB).

The STSW-ST25R-LIB embedded software provides middleware, and related examples, which can be reused when developing an application with ST25R products.

For more information, visit the STSW-ST25R-LIB web page available on www.st.com.

For details about the board function, refer to the documentation of X-NUCLEO-NFC03A1, X-NUCLEO-NFC05A1, or X-NUCLEO-NFC06A1, and the documentation of X-CUBE-NFC3, X-CUBE-NFC5, or X-CUBE-NFC6.

The demonstrations contained in the ST25 embedded NFC library provide a showcase for RFAL. They are build on the top of RFAL. The RFAL is also available, without demonstration code, in the STSW-ST25RFAL001, STSW-ST25RFAL002, and STSW-ST25RFAL003 embedded library package.

1 Demonstrations overview

These demonstrations are available for the ST25R reader chips. These demos typically run on a NUCLEO-L476RG, which has plugged in one of the X-NUCLEO-NFC03A1, X-NUCLEO-NFC05A1 or X-NUCLEO-NFC06A1 shield.

The demos currently provided are:

1. Proprietary active peer to peer (it is not available for ST25R95/X-NUCLEO-NFC03A1)
2. Simple polling loop
3. FreeRTOS™ (for the ST25R3916/X-NUCLEO-NFC06A1 only)
4. NDEF read/write demonstration
5. Embedded device test application eDTA for CCC digital key reader device profile

Table 1. **List of acronyms**

Acronym	List of acronyms
BSP	Board support package
CMSIS	Arm® Cortex® microcontroller software interface standard
HAL	Hardware abstraction layer
LED	Light emitting diode
MCU	Microcontroller unit
NFC	Near field communication
RFAL	RF abstract layer
SDK	Software development kit
SPI	Serial peripheral interface

2 Setup

This section describes the various setup requirements and procedures.

2.1 Hardware setup

The following hardware components are needed:

- STM32 Nucleo development platform for STM32L476: NUCLEO-L476RG
- Depending on interest a fitting Nucleo shield (for instance: X-NUCLEO-NFC03A1, X-NUCLEO-NFC05A1, X-NUCLEO-NFC06A1)
- One USB Type-A to USB Mini-B cable to connect the STM32 Nucleo to the PC.

2.2 Software setup

This section lists the minimum requirements for the developer to setup the SDK.

- Development tool-chains and compilers:
  - In this document, the demos are built for STM32CubeIDE.
- A serial terminal program (TeraTerm, HyperTerminal, Hterm, etc.) set to 115200 8n1.
- The drivers for the STLINK virtual COM port:
  - The Microsoft Windows® 10 system includes them.
  - For the other systems, check the STSW-LINK009.

2.3 Running the demos

Locate the appropriate demo and open/import the project into your STM32CubeIDE.

After compiling and loading the application, the user can observe the demo operation in the serial terminal. The demos are detailed the following sections.

2.3.1 Polling

This demo implements a simple polling loop for various technologies (NFC-A, NFC-B, NFC-F, NFC-V), and shows how some simple frames can be exchanged.

2.3.2 AP2P_prop

This proprietary active peer-to-peer demo is not available for ST25R95/X-NUCLEO-NFC03A1.

It requires to start two setups, which can be based on the same chip or on different chips. If the user brings these two setups together, they start to exchange a simple text message concatenated with a counter.

The used low-level communication mode is the ISO18092 or the NFC Forum active communication mode.

This demo uses standard P2P initialization and uses normal NFCDEP activation. However, after activation it directly exchanges messages on top of the NFCDEP layer.

This demo is not able to exchange messages with Android™ phones running LLCP/SNEP. It uses a normal ATR_REQ command to initiate the communication. Some phones treat the ATR_REQ as a successful establishment of the LLCP/SNEP link (this feature is not performed in this demo).

2.3.3 FreeRTOS

This demonstration shows how to integrate the RFAL for the ST25R3916 into the FreeRTOS, making the best out of MCU utilization.

This demonstration implements a simple polling loop for various technologies (NFC-A, NFC-B, NFC-F, and NFC-V), and shows how some simple frames can be exchanged. This polling loop syncs up with various events (timer expiry, interrupt from the ST25R3916) to optimize MCU utilization.

2.3.4 NDEF_RW

This demo utilizes the NDEF middleware on top of RFAL to implement NDEF read/write/format procedures.

For properly executing this demo the user needs some NFC Forum formatted tags that are T2T, T3T, T4AT, T4BT or T5T. The user can use for example tags of the ST25TA and ST25TV family, which can also be formatted using this application as explained in the below procedures.

The application waits for an NFC tag to be detected. By default, it reads its content. The user can press the blue user button to cycle among the different features:

- Write a text record
- Write a URI record and an Android application record (AAR)
- Format an ST tag
- NDEF records are read and decoded, and their content is displayed and tuned to their type, as well as stored in a message and written to the tag.

2.3.5 eDTA

This demo is used to run NFC Forum analog and digital tests for CCC digital key reader devices operating T4AT platform.

It provides an embedded DTA that implements the NFC Forum Type 4A tag platform loop-back functionality. This embedded DTA:

- polls for the NFC-A technologies, and automatically sends the start of test command once a T4T device is selected.
- loops back the data received from the tester
- ends the communication and returns to poll mode when the end of the test command is received.

The ICS/IXIT configuration parameters are available in the document folder of the eDTA demo.

3 RFAL middleware

RFAL (RF abstraction layer) provides several functionalities, which are required to perform the RF/NFC communications.

The RFAL encapsulates the different RF ICs (ST25R3911, ST25R3916, ST25R95, and forthcoming ST25R devices) into a common and easy to use interface.

RFAL is MISRA C® 2012 compliant. Detailed MISRA (Motor Industry Software Reliability Association) compliance report is available on the documentation folder.

The technologies currently supported by RFAL are:

- NFC-A \ ISO14443A (T1T, T2T, T4TA)
- NFC-B \ ISO14443B (T4TB)
- NFC-F \ FeliCa™ (T3T)
- NFC-V \ ISO15693 (T5T)
- P2P \ ISO18092 (NFCIP1, passive-active P2P)
- ST25TB (ISO14443-2 Type B with proprietary protocol)
- PicoPass \ iClass
- B' \ Calypso
- CTS \ CTM

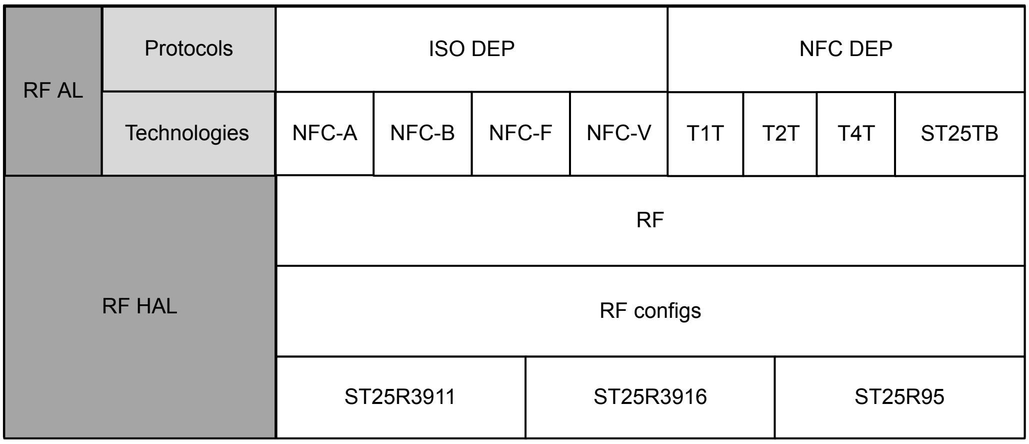
The protocols provided by RFAL are:

- ISO-DEP (ISO14443-4 Data Link Layer, T=CL)
- NFC-DEP (ISO18092 Data Exchange Protocol)

Internally the RFAL is divided into two sub layers:

- RF HL - RF higher layer
- RF HAL - RF hardware abstraction layer
- RF AL - RF abstraction layer

Figure 1. **RFAL block diagram**



Modules inside RF HAL are chip-dependent. They implement the RF IC driver, configuration tables, and the way to handle the hardware specifically to perform the same functionality.

The interface for the caller is a shared RF header file, which provides the same interface for the upper layers (for all chips).

Above the RF AL is also composed by another two different sub layers:

- Technologies - technology modules, which implement all its specifics, framing, timings, etc.
- Protocols - protocol implementation with the all framing, timings, error handling, etc.

So, in addition to these, there is the application layer that uses of the RFAL functionality. For example: *NFC Forum activities* (NFCC), *EMVco*, DISCO/NUCLEO demo, etc.

The RFAL NFC module provides an interface to perform the common activities as Poller/Listener device.

The access to the lowest functionality of the ICs is granted by the RF module. Caller can make direct use of any of the RF, technology, or protocol layer in a non-hardware-specific way.

3.1 RFAL APIs

Detailed technical information on the APIs is available to the user in a compiled CHM file. This file is located inside the rfal\doc folder of the software package, where all the functions and parameters are fully described.

4 NDEF middleware

The NDEF library design is split into an RF technology-dependent layer and an RF technology-independent layer:

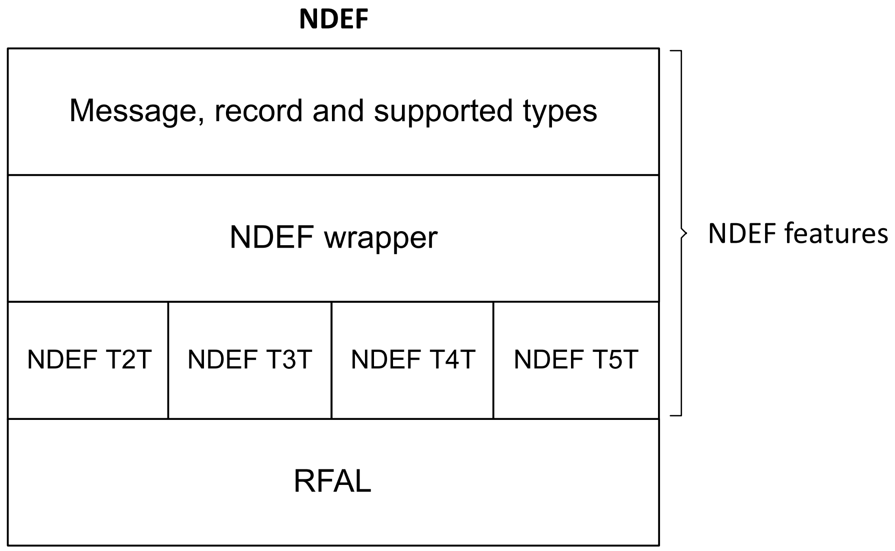
- Message, record, and supported type management layer (technology independent)
- NDEF technology layer defining a common API on top of the RFAL (technology dependent)
- NDEF wrapper layer abstracting the underlying technologies.

The NDEF wrapper on top of the NDEF technology-dependent components allows managing NDEF tags without taking care of the underlying NFC technologies.

The types currently supported are:

- RTD device information
- RTD text
- RTD URI
- Android application record (AAR)
- vCard
- Wi-Fi.

Figure 2. **Figure 2 NDEF block diagram**



4.1 NDEF APIs

Detailed technical information on the NDEF APIs is available to the user in a compiled CHM file. This file is located inside the ndef\doc folders of the software package where all the functions, and parameters are fully described.

Revision history

Table 2. Revision history

Date	Revision	Changes
10-Jul-2020	1	Initial release
22-Apr-2022	2	Updated: • Section 1 Demonstrations overview Added: • Section 2.3.5 eDTA

Contents

List of tables

List of figures

IMPORTANT NOTICE – READ CAREFULLY