Getting started with the X-CUBE-NFC5 high performance HF reader / NFC initiator IC software expansion for STM32Cube

Introduction

The X-CUBE-NFC5 software expansion for STM32Cube provides the complete STM32 middleware to build applications using the ST25R3911B high performance HF reader / NFC initiator IC. The package contains a sample application to detect different types of NFC tags and mobile phones supporting P2P.

The software provides sample implementations using an STM32 Nucleo platform equipped with the X-NUCLEO-NFC05A1 expansion board, featuring the ST25R3911B high performance HF reader / NFC initiator IC.

The software is based on STM32Cube technology, rendering it easily portable across different MCU families.

Contents

List of tables

List of figures

1   Acronyms and abbreviations

Table 1: List of acronyms

Acronym	Description
NFC	Near field communication
RFAL	Rf abstract layer
P2P	Peer to peer
MCU	Microcontroller unit
BSP	Boot support package
HAL	Hardware abstraction layer
LED	Light emitting diode
SPI	Serial peripheral interface
CMSIS	The ARM® Cortex® microcontroller software interface standard

2 What is STM32Cube?

STMCube™ represents the STMicroelectronics initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio.

STM32Cube version 1.x includes:

*   STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
*   A comprehensive embedded software platform specific to each series (such as the STM32CubeF4 for the STM32F4 series), which includes:
    *   the STM32Cube HAL embedded abstraction-layer software, ensuring maximized portability across the STM32 portfolio
    *   a consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
    *   all embedded software utilities with a full set of examples

2.1 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with one another, as described in the diagram below.

Figure 1: Firmware architecture
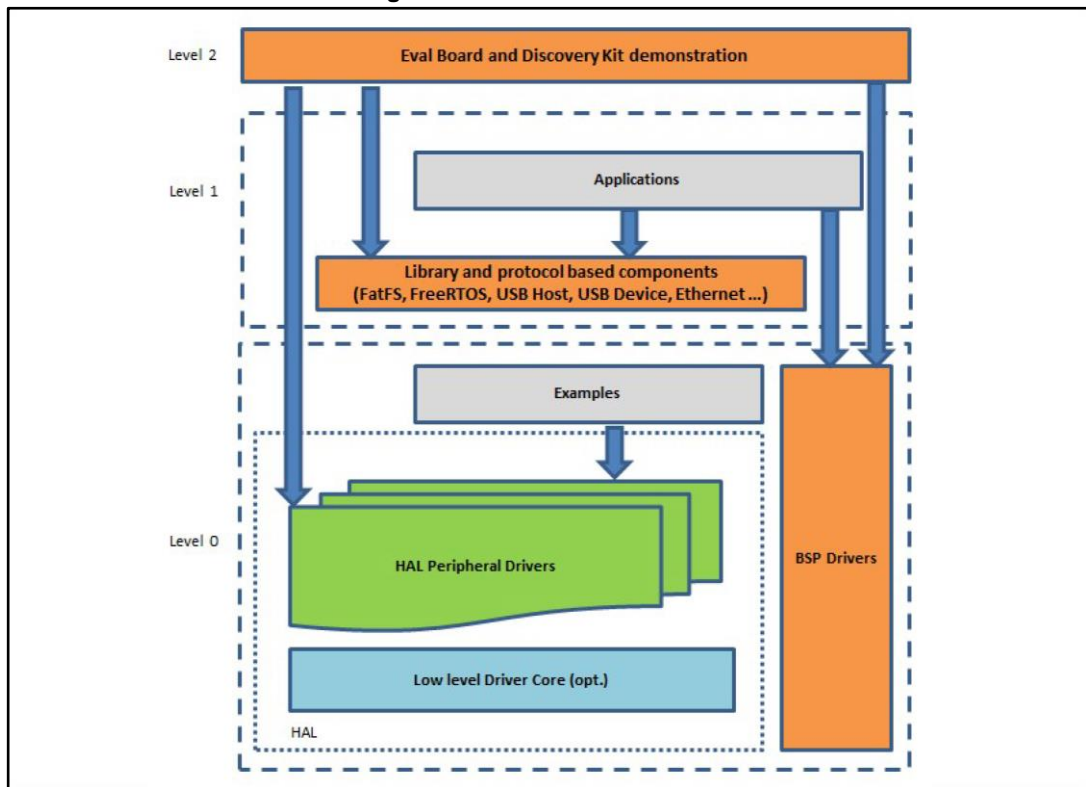
Level 0: This level is divided into three sub-layers:

*   Board Support Package (BSP): this layer offers a set of APIs relative to the hardware components in the hardware boards (Audio codec, IO expander, Touchscreen, SRAM driver, LCD drivers. etc…); it is based on modular architecture allowing it to be easily

ported on any hardware by just implementing the low level routines. It is composed of two parts:

- Component: is the driver relative to the external device on the board and not related to the STM32, the component driver provides specific APIs to the external components of the BSP driver, and can be ported on any other board.
- BSP driver: links the component driver to a specific board and provides a set of easy to use APIs. The API naming convention is BSP_FUNCT_Action(): e.g., BSP_LED_Init(), BSP_LED_On().

- Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes. For example, for the communication peripherals (I²C, UART, etc.) it provides APIs for peripheral initialization and configuration, data transfer management based on polling, interrupt or DMA processes, and communication error management. The HAL Drivers APIs are split in two categories: generic APIs providing common, generic functions to all the STM32 series and extension APIs which provide special, customized functions for a specific family or a specific part number.
- Basic peripheral usage examples: this layer houses the examples built around the STM32 peripherals using the HAL and BSP resources only.

Level 1: This level is divided into two sub-layers:

- Middleware components: set of libraries covering USB Host and Device Libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. Horizontal interaction among the components in this layer is performed directly by calling the feature APIs, while vertical interaction with low-level drivers is managed by specific callbacks and static macros implemented in the library system call interface. For example, FatFs implements the disk I/O driver to access a microSD drive or USB Mass Storage Class.
- Examples based on the middleware components: each middleware component comes with one or more examples (or applications) showing how to use it. Integration examples that use several middleware components are provided as well.

Level 2: This level is a single layer with a global, real-time and graphical demonstration based on the middleware service layer, the low level abstraction layer and basic peripheral usage applications for board-based functions.

3 X-CUBE-NFC5 software expansion for STM32Cube

3.1 Overview

X-CUBE-NFC5 software package expands the STM32Cube functionality.

The key features of the package are:

- Complete middleware to build applications using the ST25R3911B high performance HF reader / NFC initiator IC
- Easy portability across different MCU families, thanks to STM32Cube
- Sample application to detect NFC tags of different types and mobile phones supporting P2P
- Free user-friendly license terms
- Sample implementations on X-NUCLEO-NFC05A1 board plugged onto a NUCLEO-F401RE or NUCLEO-L476RG development board

This software contains high performance HF reader / NFC initiator IC drivers for the ST25R3911B device, running on STM32. It is built on top of STM32Cube software technology to ease portability across different STM32 microcontrollers.

This firmware package includes component device drivers, a board support package and a sample application demonstrating usage of X-NUCLEO-NFC05A1 expansion board with STM32 Nucleo boards.

The sample application configures the ST25R3911B for inductive or capacitive wake up, followed by a polling loop for active and passive device detection. When a passive tag or active device is detected, the shield signals the detected technology by lighting a corresponding LED.

The demo logs all activities with ST-Link Virtual Com Port to the host system.

The supported RFID technologies in this demo are:

- ISO14443A/NFCA
- ISO14443B/NFCB
- Felica/NFCF
- ISO15693/NFCV
- Active P2P

3.2 Architecture

This fully compliant software expansion for STM32Cube lets you develop applications using the ST25R3911B high performance HF reader / NFC initiator IC. It is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube with a board support package (BSP) for the X-NUCLEO-NFC05A1 expansion board.

Application software can access and use the X-NUCLEO-NFC05A1 expansion board through the following layers:

- **STM32Cube HAL layer**: the HAL driver layer provides a simple set of generic, multi-instance APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). These generic and extension APIs are directly built on a common architecture and allow overlying layers like middleware to implement their functions without depending on specific microcontroller unit (MCU) hardware

information. This structure improves the library code reusability and guarantees easy portability across other devices.

- **Board support package (BSP) layer**: provides support for the peripherals on the STM32 Nucleo board (apart from the MCU). This set of APIs provides a programming interface for certain board-specific peripherals like the LED, the user button etc. This interface also helps you identify the specific board version.

- **Middleware RF abstraction layer (RFAL)**: RFAL provides several functions for RF/NFC communication. It groups the different RF ICs (existing ST25R3911B product family and future ST25R391x devices) under a common and easy to use interface. The **technologies** currently supported by RFAL are:
  The **protocols** provided by RFAL are:ISO-DEP (ISO14443-4 Data Link Layer, T=CL)NFC-DEP (ISO18092 Data Exchange Protocol)
    – 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)

Internally, the RFAL is divided into two sub layers:

- RF HAL- RF hardware abstraction layer
- RF AL - RF abstraction layer

Figure 2: RFAL block diagram

* Future devices added to the ST25R family
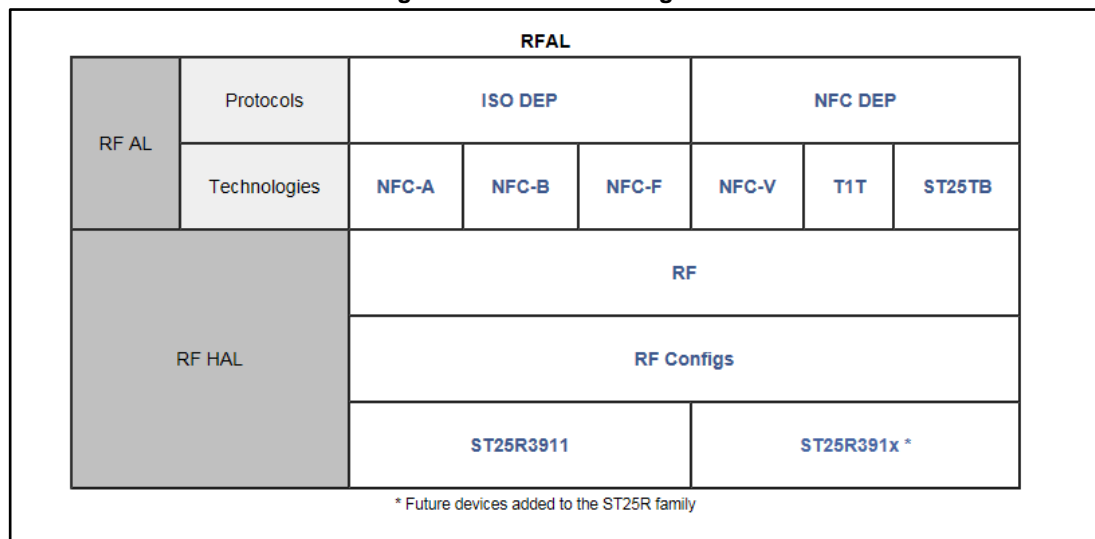
The modules in the RF HAL are chip-dependent, they implement the RF IC driver, configuration tables and specific instructions for the HW to perform the physical RF functions. The interface for the caller is a shared RF header file which provides the same interface for upper layers (for all chips).

The RF AL can be broken down into two further sub layers:

• Technologies: technology modules which implement all the specifics, framing, timings, etc.

• Protocols: protocol implementation including all the framing, timings, error handling, etc

On top of these, the application layer uses RFAL functions like NFC Forum Activities, EMVCo compliant contactless reader, GP demo, etc.
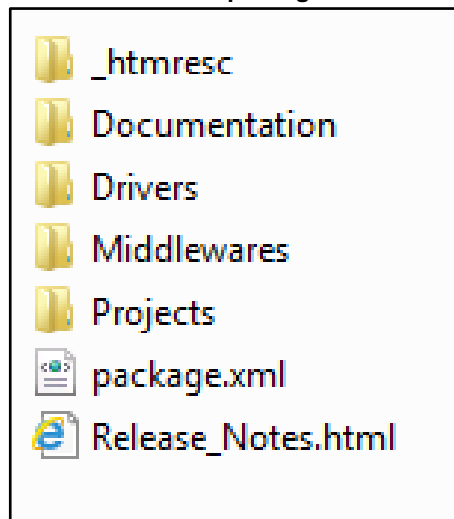
Access to the lowest functions of the ICs is granted by the RF module. The caller can make direct use of any of the RF technology or protocol layers without requiring any specific hardware configuration data.

Figure 3: X-CUBE-NFC5 software architecture
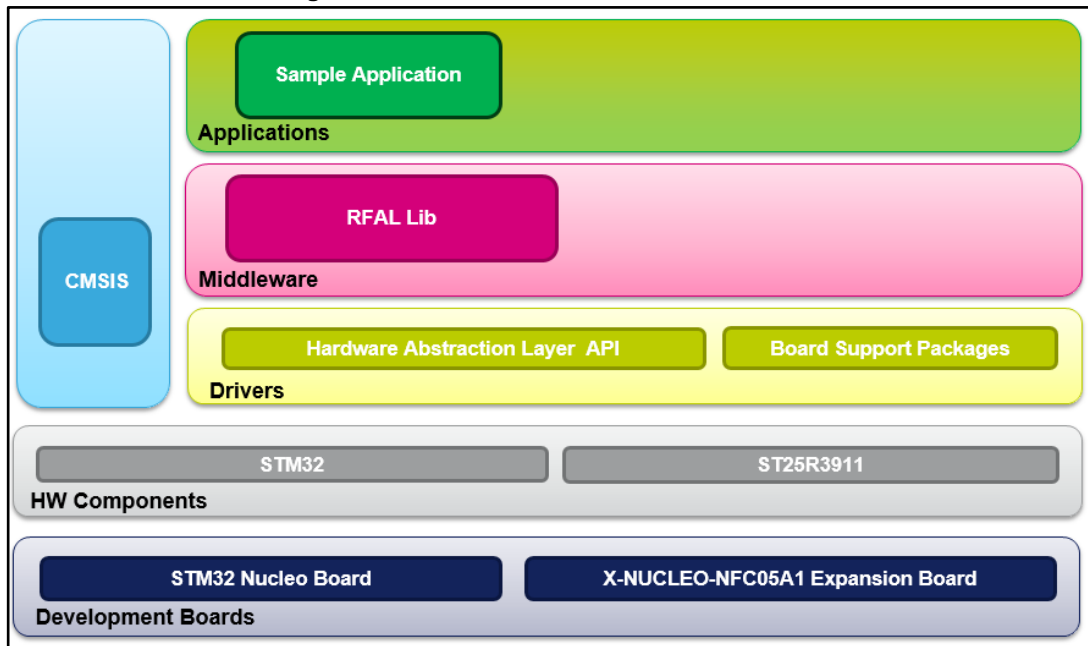
3.3 Folder structure

Figure 4: X-CUBE-NFC5 package folders structure

The following folders are included in the software package:

- **Documentation**: this folder contains a compiled HTML file generated from the source code which details the software components and APIs.
- **Drivers**: this folder contains the HAL drivers, the board-specific drivers for each supported board or hardware platform, including the on-board components, and the CMSIS vendor-independent hardware abstraction layer for the Cortex-M processor series.

- **Middlewares**: this folder contains RFAL (RF abstraction layer). RFAL provides several functions required to perform RF/NFC communication.

The RFAL groups the different RF ICs (ST25R3911 and future ST25R391x devices) under a common and easy to use interface.

- **Projects:** this folder contains a sample application example Tag Detect, provided for the NUCLEO-L476RG and NUCLEO-F401RE platforms with three development environments (IAR Embedded Workbench for ARM, Keil Microcontroller Development Kit (MDK-ARM), and System Workbench for STM32 (SW4STM32)).

An RFAL usage example as a Poller device is provided in **exampleRfalPoller.c**. In this example, different devices are detected and activated, and data is exchanged implementing a presence check mechanism. Once removed or upon error, the device is deactivated and the discovery loop restarts.

3.4   APIs

Detailed technical information about the APIs available to the user can be found in a compiled CHM file located inside the "Documentation" folder of the software package where all the functions and parameters are fully described.

3.5   Sample application

A sample application using the X-NUCLEO-NFC05A1 expansion board with the NUCLEO-F401RE or NUCLEO-L476RG development board is provided in the "Projects" directory. Ready-to-build projects are available for multiple IDEs.

In this application, NFC tags of different types of mobile phones supporting P2P are detected by the ST25R3911B high performance HF reader / NFC initiator IC.

See the CHM documentation file generated from the source code for more details regarding the sample application.

After system initialization and clock configuration, LED101, LED102, LED103, LED104 and LED105 blink 3 times.

When a tag is detected in the vicinity, a LED is lit on the NFC5 shield according to the table below.

Table 2: LED Lit on tag detection

| NFC tag type	LED lit on tag detection
NFC TYPE F	LED101 / Type F
NFC TYPE B	LED102 / Type B
NFC TYPE A	LED103 / Type A
NFC TYPE V	LED104 / Type V
NFC TYPE AP2P	LED105 / Type AP2P

ST's virtual comport interface is also included: following system initialization, the board is configured and enumerated as an ST Virtual comport.

Figure 5: ST Virtual com port enumeration
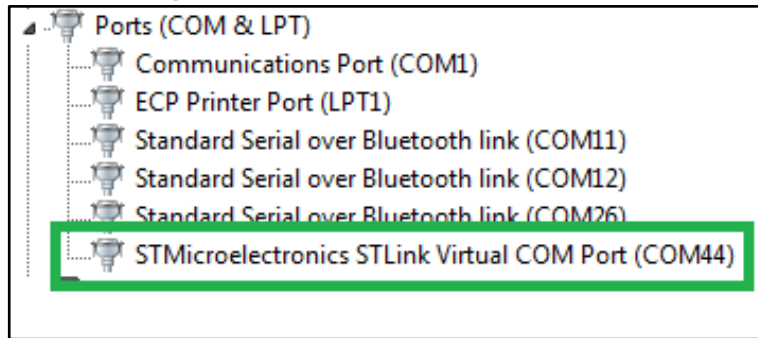
After checking the virtual COM port number, open a connection on Hyperterminal (or similar) with the configuration shown below (enable option: Implicit CR on LF, if available).

Figure 6: UART serial communication configuration
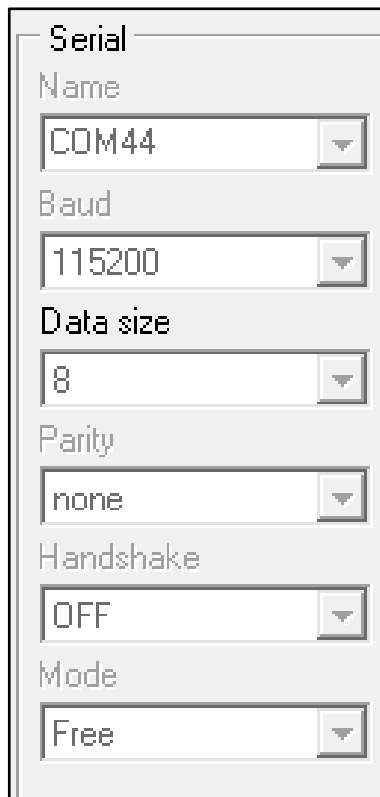
Following successful connection, the user can view the messages on the Hyperterminal, as shown below.

Figure 7: UART serial communication displayed on on Hyperterminal

```
Welcome to X-NUCLEO-NFC05A1
RFAL initialization succeeded..
Going to Wakeup mode.
Inductive Wakeup received.
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
Going to Wakeup mode.
Inductive Wakeup received.
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
ISO15693/NFC-V card found. UID: E00220423819874F
Going to Wakeup mode.
```

4 System setup guide

4.1 Hardware description

This section describes the hardware components required to develop a sensor-based application.
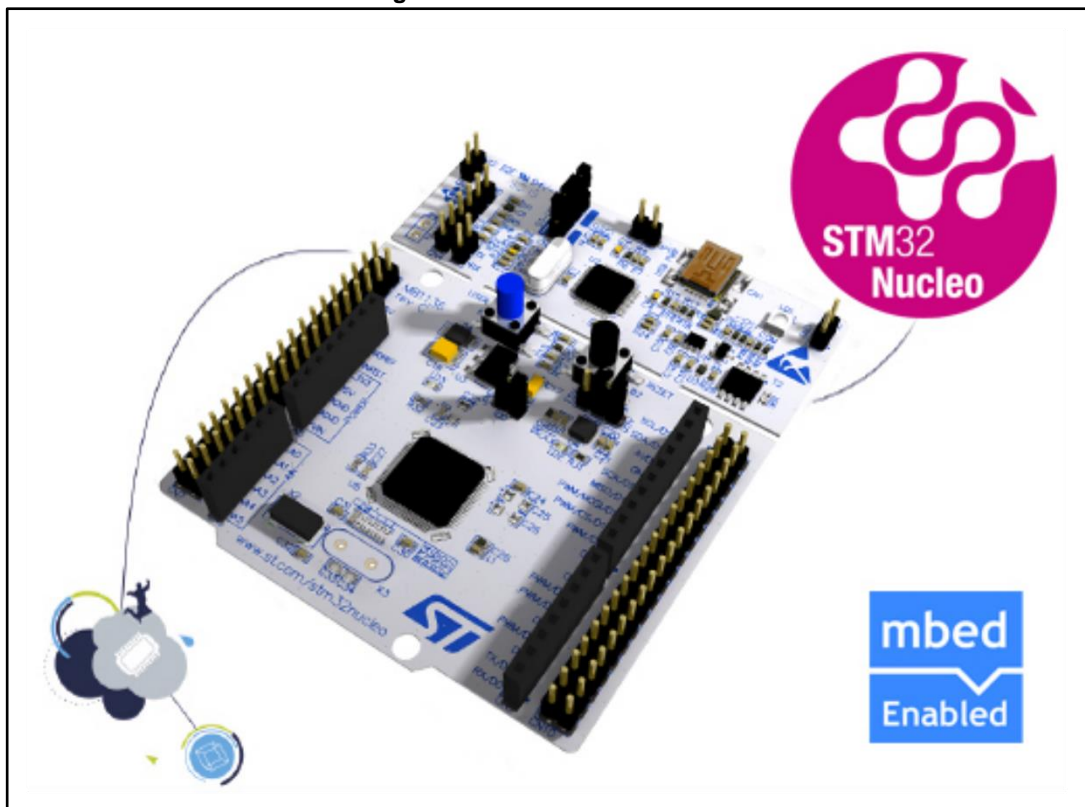
4.1.1 STM32 Nucleo platform

STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino™ connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

Figure 8: STM32 Nucleo board

Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

4.1.2 X-NUCLEO-NFC05A1 expansion board

The X-NUCLEO-NFC05A1 is a high performance HF reader / NFC initiator expansion board usable with the STM32 Nucleo platform. It is also compatible with Arduino™ UNO

R3 connector layout, and is designed around the STMicroelectronics IC ST25R3911B high performance HF reader / NFC initiator (with 1.4 W supporting VHBR and AAT). The X-NUCLEO-NFC05A1 interfaces with the STM32 MCU via SPI.

Figure 9: X-NUCLEO-NFC05A1 expansion board
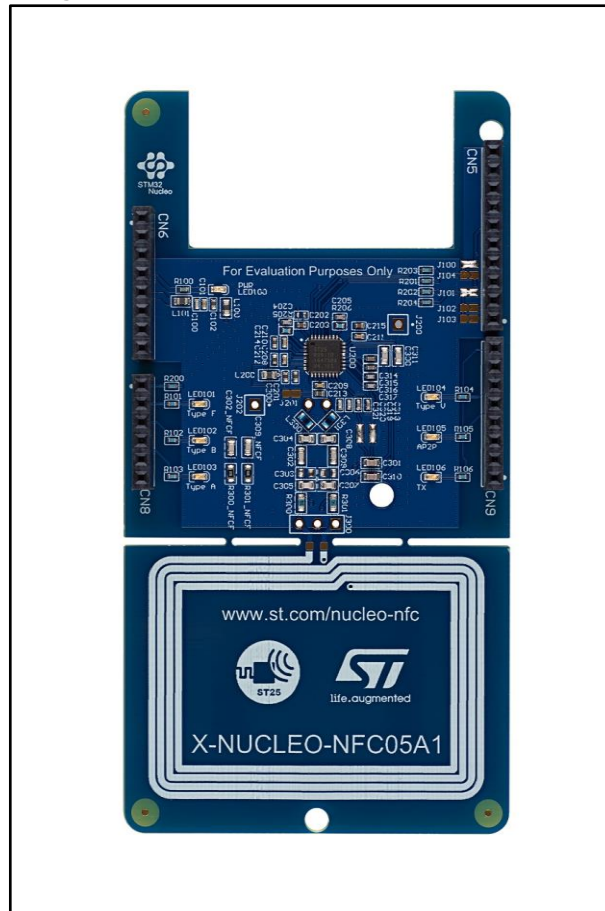
Information regarding the X-NUCLEO-NFC05A1 expansion board is available on *www.st.com* at:

http://www.st.com/x-nucleo

4.2 Software description

The following software components are needed in order to setup the suitable development environment for creating applications for the STM32 Nucleo equipped with the NFC expansion board:

- X-CUBE-NFC5: an expansion for STM32Cube dedicated to NFC applications development. The X-CUBE-NFC5 firmware and related documentation is available on www.st.com.
- Development tool-chain and Compiler: The STM32Cube expansion software supports the three following environments:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-Link
  - Keil Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
  - System Workbench for STM32 (SW4STM32) + ST-LINK

4.3 Hardware and software setup

This section describes the hardware and software setup procedures. It also describes the system setup needed for the above.

4.3.1 Hardware setup

The following hardware components are required:

- One STM32 Nucleo Development platform (suggested order code: either NUCLEO-F401RE or NUCLEO-L476RG)
- One ST25R3911B High performance HF reader / NFC initiator IC expansion board (order code: X-NUCLEO-NFC05A1)
- One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

4.3.2 Software setup

This section lists the minimum requirements for the developer to set up the SDK.

4.3.2.1 Development tool-chains and compilers

Select one of the integrated development environments (IDE) supported by the STM32Cube expansion software and read the system requirements and setup information provided by the IDE provider.

4.3.3 System setup guide

This section describes how to set up the different hardware components before writing and executing an application on the STM32 Nucleo board with the X-NUCLEO-NFC05A1 expansion board.

4.3.3.1 STM32 Nucleo and X-NUCLEO-NFC05A1 expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. You can download the ST-LINK/V2-1 USB driver by searching STSW-LINK009 software on *www.st.com*.

The X-NUCLEO-NFC05A1 expansion board is easily plugged onto the STM32 Nucleo development board through the Arduino™ UNO R3 extension connector.

It interfaces with the STM32 microcontroller on STM32 Nucleo board through the SPI transport layer.

5 Revision history

Table 3: Document revision history

Date	Version	Changes
03-Jul-2017	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY