# AN4548
# Application note

## EVAL-ST95HF firmware description

## Introduction

This document describes the firmware (STSW-ST95HF001) for the EVAL-ST95HF board.

It has the goal of helping the developer to understand how this firmware works, and how to tailor the application.

The ST95HF is an RFID transceiver. It supports ISO/IEC 14443A, ISO/IEC 14443B, ISO/IEC 15693 and ISO/IEC 18092 in reader mode (PCD), and ISO/IEC 14443A in card emulation mode (PICC).

The MCU of the EVAL-ST95HF is the STM32F103. The ST95HF, together with the microcontroller, emulates a PCD or a PICC. The MCU communicates with the ST95HF by means of the SPI bus.

*Note: It supposed that the developer has already read UM1795 "EVAL-ST95HF firmware functionalities", as no informations from user point of view will be given in this document.*

# Contents

# List of tables

# List of figures

# 1 Acronyms and notational conventions

## 1.1 Acronyms

- ISO: International Organization for Standardization
- IEC: International Electrotechnical Commission
- IAP: In Application Programming
- HID: Human Interface Device
- MCU: Micro Controller Unit
- NFC: near field communication
- RF: radio frequency
- RFID: Radio Frequency Identification
- uC: micro Controller
- USB: Universal Serial Bus.

## 1.2 Representation of numbers

The following conventions and notations apply in this document unless otherwise stated:

### Binary numbers

Binary numbers are represented by strings of digits 0 and 1 shown with the most significant bit (MSB) on the left, the least significant bit (LSB) on the right, and a “0b” added at the beginning.

Example: 0b11110101.

### Hexadecimal numbers

Hexadecimal numbers are represented by using the numbers 0 to 9 and the characters A – F, and adding an “0x” at the beginning. The Most Significant Byte (MSB) is shown on the left and the Least Significant Byte (LSB) on the right.

Example: 0xF5.

### Decimal numbers

Decimal numbers are represented as is without any trailing character.

Example: 245.

# 2 Overview

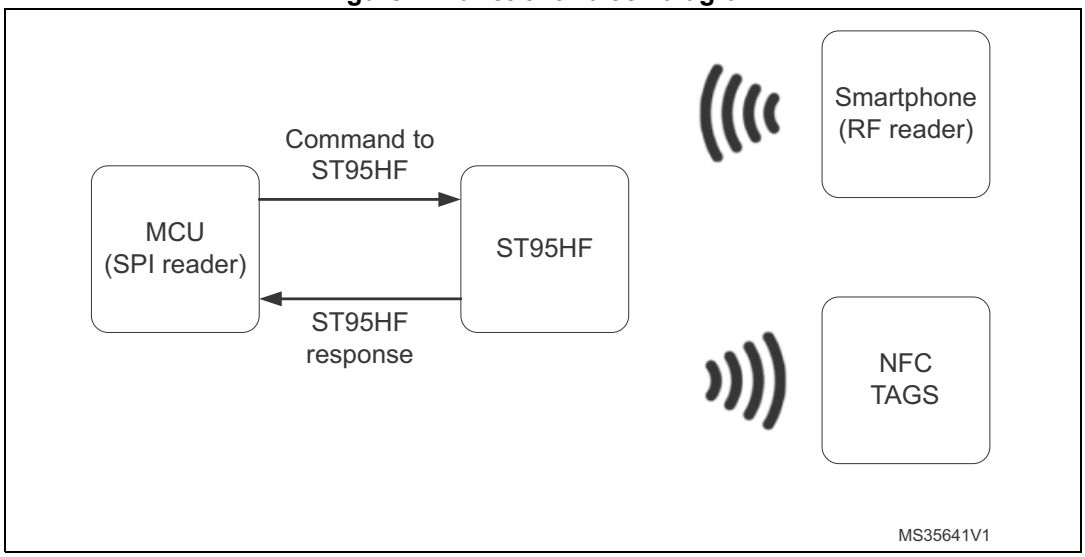
## 2.1 ST95HF overview

ST95HF is a transceiver for contactless application. It includes frame coding and RF modulation, thus allowing the connected MCU to send and receive NFC commands in the supported protocols.

ST95HF is a slave device, hence an host (MCU) is required to control it.

The ST95HF is connected to the MCU using an SPI communication. The ST95HF is able to act as a PCD or a PICC so it can interact with a tag or with a reader, as exemplified in *Figure 1*.

**Figure 1. Functional block diagram**




For more details concerning the ST95HF device, please refer to its datasheet.

## 2.2 STM32F103 overview

The STM32F103xx incorporates the high-performance ARM® Cortex®-M3 32-bit RISC core operating at a 72 MHz frequency, high-speed embedded memories (Flash memory up to 1 Mbyte and SRAM up to 96 Kbytes), and an extensive range of enhanced I/Os and peripherals connected to two APB buses. All devices offer two 12-bit ADCs, three general purpose 16-bit timers plus one PWM timer, as well as standard and advanced communication interfaces: up to two I2Cs and SPIs, three USARTs, an USB and a CAN.

These features make the STM32F103xx microcontrollers suitable for a wide range of applications such as motor drives, application control, medical and handheld equipment, PC and gaming peripherals, GPS platforms, industrial applications, PLCs, inverters, printers, scanners, alarm systems, video intercoms, and HVACs.

## 2.3 EVAL-ST95HF board

The EVAL-ST95HF is a kit which allows to evaluate the ST95HF transceiver performance.

The EVAL-ST95HF is powered through the USB bus and no external power supply is required. It includes a ST95HF, a 47x34 mm 13.56 MHz simple layer inductive etched antenna and its associated tuning components.

By default, the ST95HF communicates with the STM32F103RG 32-bit MCU via the SPI bus.

Pictures of the board are shown in *Figure 2* and *Figure 3*.

**Figure 2. EVAL ST95HF board (front side)**

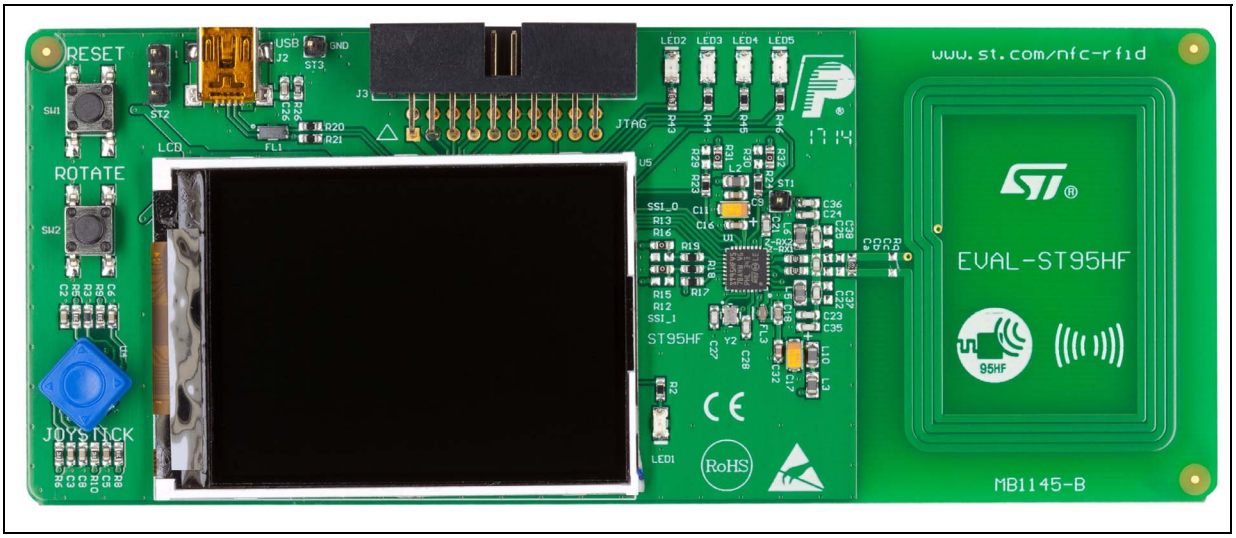
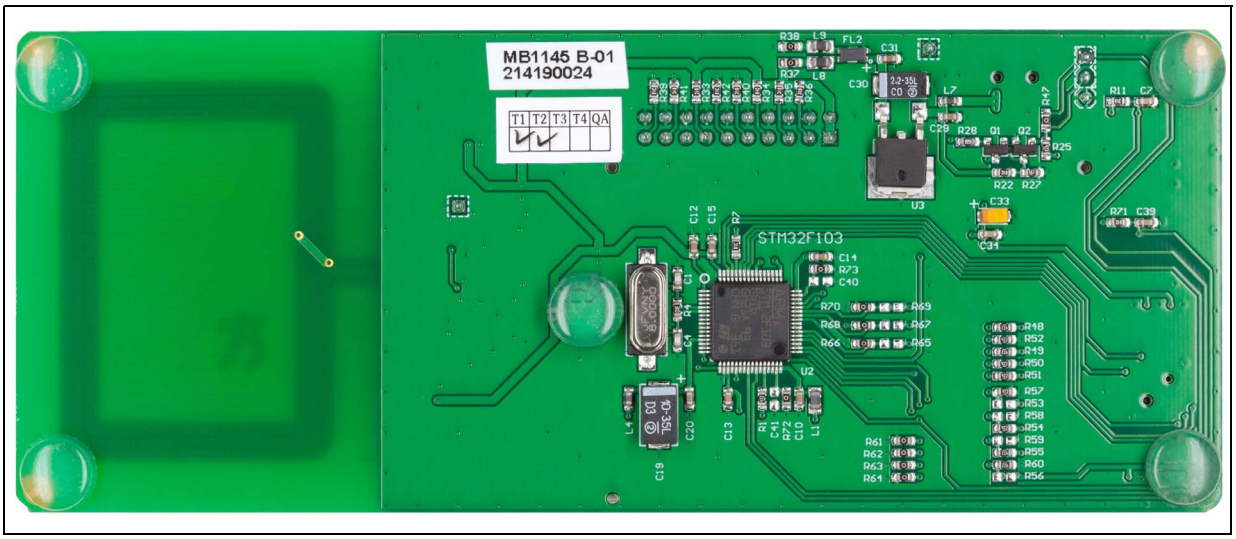



**Figure 3. EVAL ST95HF board (back side)**



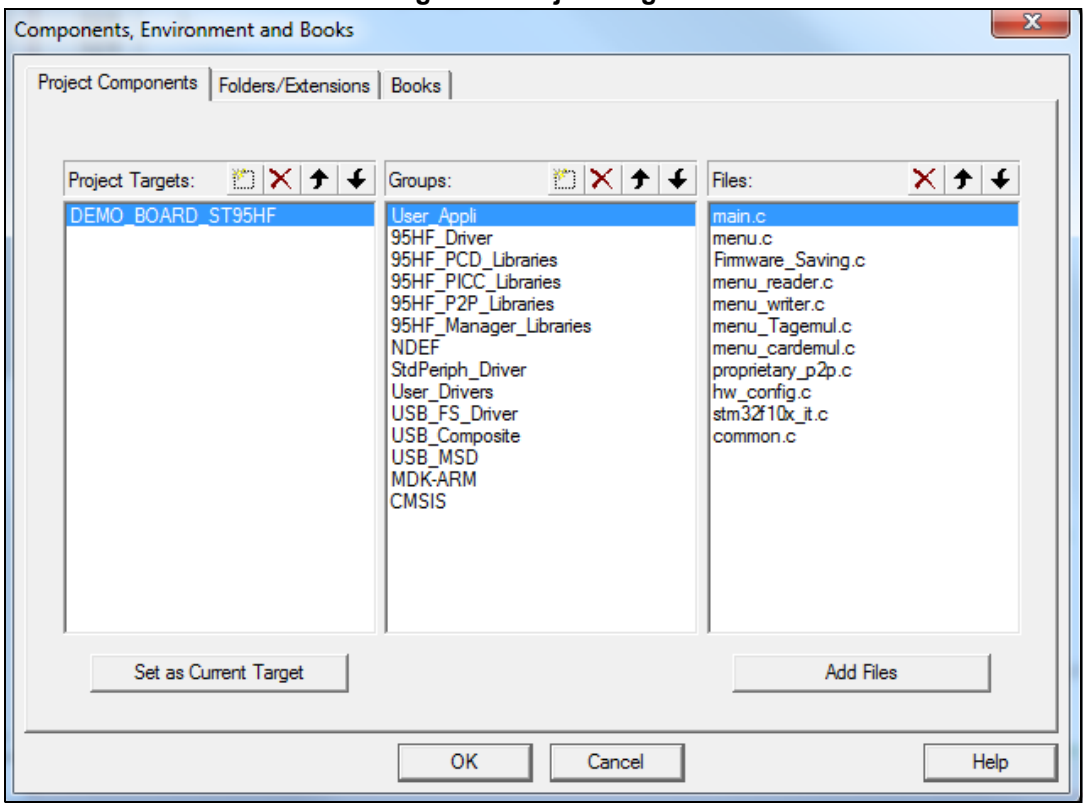
# 3 Firmware overview

## 3.1 The project target

The firmware was developed on the Keil® µvision and can be used on the EVAL-ST95HF.

Thus the Keil project contains the project DEMO_BOARD_ST95HF, as can be seen from *Figure 4*.

**Figure 4. Project targets**

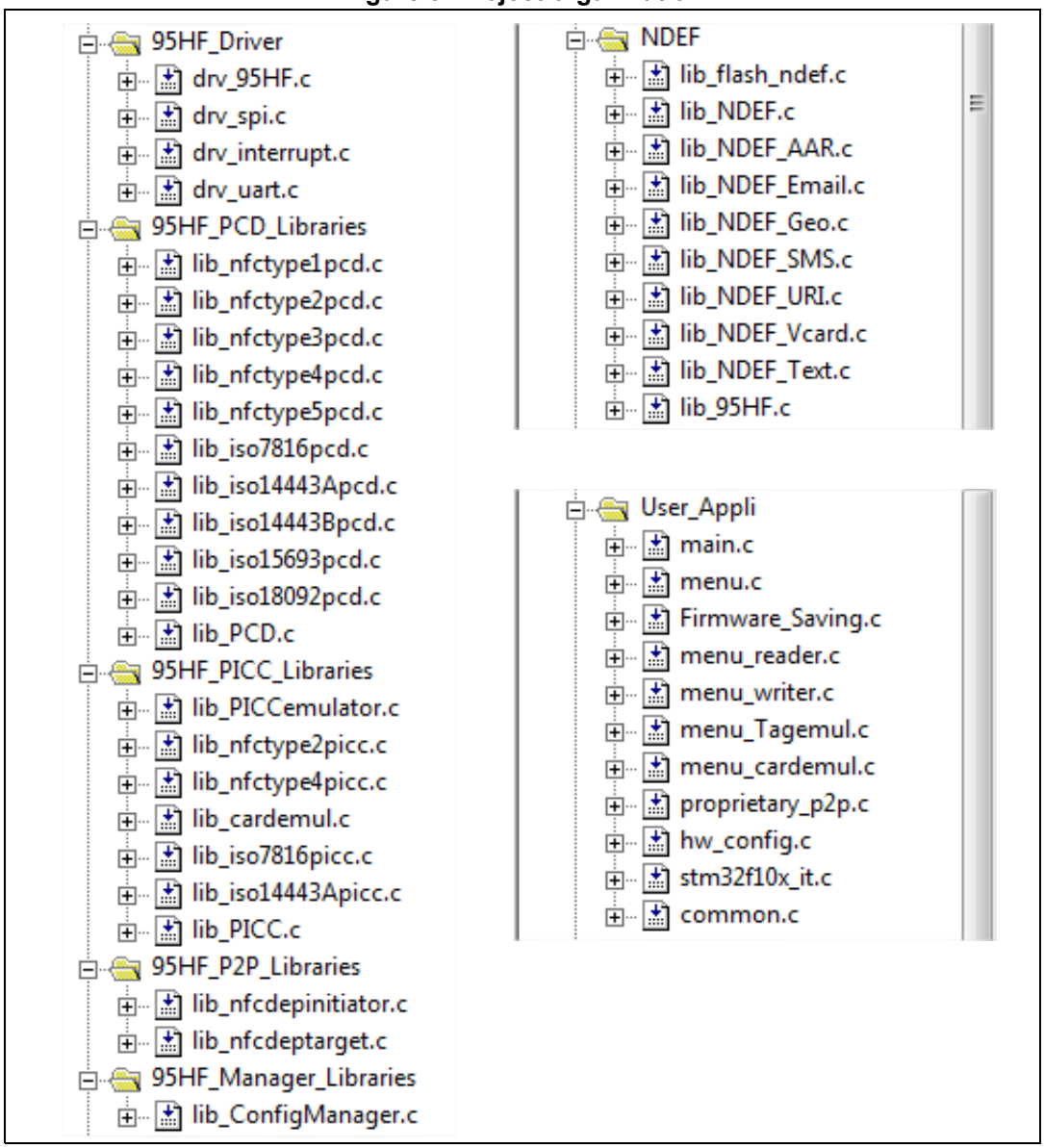



## 3.2 The firmware package

The firmware, developed using Keil µvision 4.71, is delivered in a ZIP file and contains all the subdirectories, the .h and .c source code files that make up the core of the application. The related Keil workspace/project files are included too.

As shown in *Figure 5*, the project is organized in project folders.

**Figure 5. Project organization**



The firmware contains all the application task source files and the related modules files and consists of the following project folders:

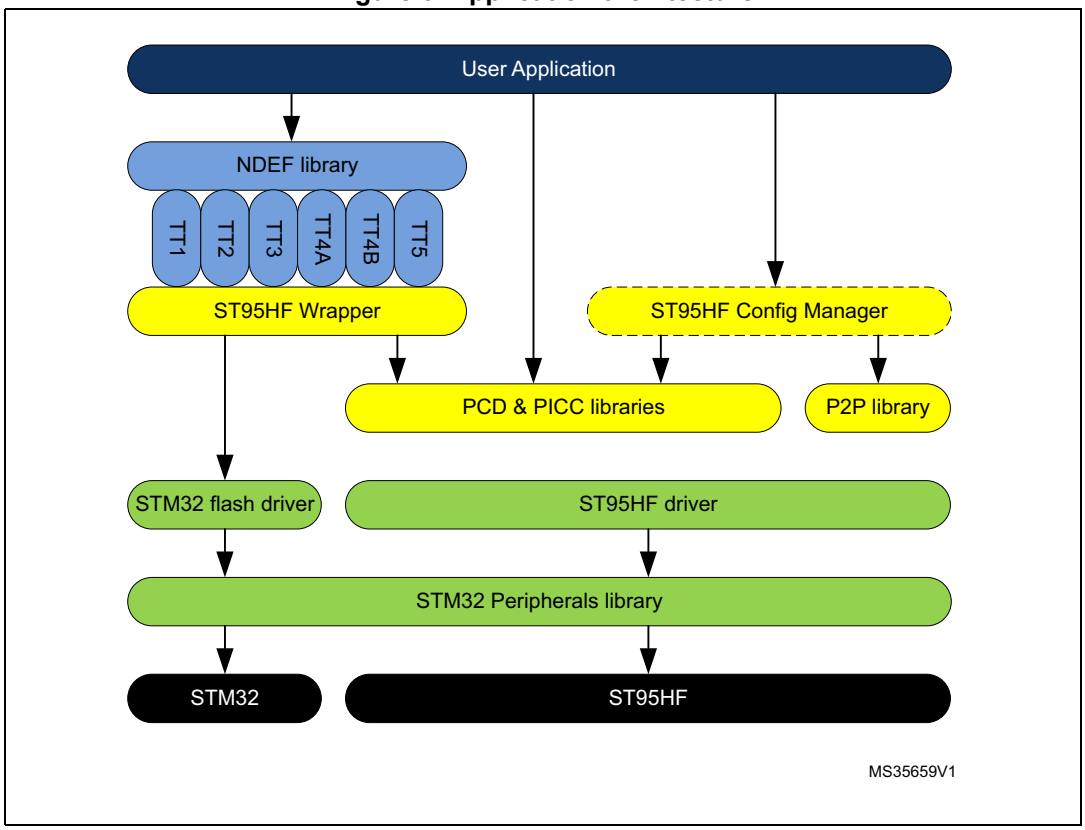
- **User_Appli**: the application layer, all the menu developed for the demonstration are available in this folder.
- **95HF_Driver**: the drivers manages the GPIO of the MCU to communicate to the 95HF transceiver via the SPI bus. Offer the basic function to send/receive data with ST95HF.
- **95HF_PCD_Libraries**: this library includes the function to provide PCD capabilities. All the commands accepted by ST95HF in this mode are available, as well as the supported protocols in PCD mode. Files to manage tag type are present in this library.
- **95HF_PICC_Libraries**: this library includes the function to provide PICC capabilities. All the commands accepted by ST95HF in this mode are available and the supported ISO14443A protocol in PICC mode is available. File to emulate TT2 and TT4A are present in this library.
- **95HF_P2P_Libraries**: this library includes the function to provide P2P capabilities. (Actually NFC-DEP is present which allow to make some proprietary P2P demonstration.)
- **95HF_Manager_Libraries**: this library is optional, it can be used to automate some actions, like switching in different mode PCD/PICC/P2P, selecting the protocol supported, etc…
- **NDEF**: this library contains files to manage NDEF message.
- **StdPeriph_Driver**: the STM32 MCU standard library
- **User_Drivers**: this is linked to the feature supported by the board hardware. This folder contains the source file that manages the LCD of the EVAL-95HF board and also the management of the LED.

## 3.3 Application firmware architecture

The architecture is sketched in *Figure 6*, to better understand it please refer to the following color code:

- in **Blue**: files not directly linked to ST95HF, but to the NFC protocol. They have been created for the demonstration, you can leave them unchanged, improve or replace altogether according to your own application.
- in **Yellow**: this is the heart of the firmware, these files are directly linked to the protocol supported by ST95HF, that must be implemented in the microcontroller that drives the ST95HF. The ST95HF wrapper is here to adapt the NDEF library, and ST95HF Config Manager is optional but can be useful in more complex cases.
- in **Green**: these are the files you have to modify regarding your HW platform, unless you are using a microcontroller of the STM32F1 series to drive the ST95HF (in this case you can keep the files without any changes).

**Figure 6. Application architecture**



# 4 Principle of the application

In the firmware the STM32 drives the ST95HF to:

1. read/write tag (PCD mode);
2. emulate card/tag (PICC mode); or
3. allow P2P data exchange.

The ST95HF can receive a command only from STM32, but data can be received either from STM32 or from an external reader or tag. If the data has been sent by STM32, it has been done through SPI channel, if it comes from an external reader or tag it has been sent by RF.

All the protocols supported by ST95HF in reader or emulation mode have been implemented in this evaluation software.

To illustrate them, the firmware can configure the ST95HF to read/write tag with TEXT, URI, SMS, Email, Vcard, Geolocation information (TT1, TT2, TT3, TT4A, TT4B and vicinity card are supported). It can also simulate tag with the same content (TT2 and TT4A).

You will be also able to download file in card emulation mode, this is a good way to proceed if you want to download a firmware or a picture.

To illustrate ST95HF proprietary P2P (but NFC forum protocol compliant using NFC-DEP) application, a demonstration has been developed between 2 EVAL-ST95HF boards. The principle is a data exchange between a client and a server to simulate a tennis game.

*Note: It's also possible to take the control of the board through USB, in this case a PC software is needed.*

## 4.1 Low layer

This part is covered by the ST95HF datasheet, here only a short reminder.

### 4.1.1 Data exchange synchronization

The synchronization of the data exchange between ST95HF and host can be achieved in two ways:

1. polling
2. interrupt mode

By default the interrupt mode is selected by the flag SPI_INTERRUPT_MODE_ACTIVATED in the drv_interrupt.h file.

## 4.2 Driver and Lib_ST95HF layers

In order to simplify design and code integration with the 95HF family, the driver contains the file which will be commonly used by ST95HF, CR95HF and RX95HF.

*Note: UART mode is only available with CR95HF, so drv_uart.c file is not used with ST95HF or RX95HF.*

Over this driver layer and moreover, to keep 95HF family consistency, all the commands supported by ST95HF have been split into 2 main files: lib_PCD.c and lib_PICC.c. Some commands are only used in emulation mode (for instance LISTEN) so they are only available in the PICC.c file. Other commands are present in both files.

This has been done to ease code divisibility for the 95HF family, the PCD library can be used with CR95HF, PICC library can be used with RX95HF.

As in reader/writer mode 14443A/B, 18092 and 15693 protocols are supported you will find these protocol developed in the 95HF_PCD_Libraries. In addition to these files, to manage all the tag types, you will found lib_nfctypexpcd.c file.

In emulation mode as only 14443A is supported, only the file for this protocol and those for TT2 and TT4A are present.

The 95HF_P2P_Librariries has been added to support P2P mode, ST95HF can be initiator or target. As defined in the NFC Forum P2P mode are based on NFC-DEP.

## 4.3 NFC library layer

For the purpose of the demonstration with the EVAl-ST95HF a “NFC library” has been developed to manage tag content. Indeed the content of the tag read or emulated must be formatted in respect to the NFC forum standard to enable native interaction with smartphone (no need to install specific applications on it).

For more information please refer to NFCForum-TS-NDEF_1.0 documentation.

This library has been created to manage this content Read, Parse and Identify content type or Format and Write NDEF message.

So this part of the software is not directly linked to ST95HF but to any NFC Forum tag. That’s the reason why a lib_wrapper.h file has been created. Inside the NFC library all the function call made to the lower layer are done through this API:

```
ReadData();
WriteData();
```

The file lib_wrapper.h allows redirecting function call to ST95HF functions.

For the purpose of the demonstration the NFC library manage NDEF messages that represent:

- TEXT
- URI
- SMS
- Email
- Vcard
- Geolocation

It’s also possible to add AAR record.

To ease NDEF message generation, specific functions have been developed:

```
u16 NDEF_ReadNDEF(u8 *pNDEF);
u16 NDEF_WriteNDEF(u8 *pNDEF);
u16 NDEF_ReadURI (sURI_Info *pURI);
```

```
u16 NDEF_WriteURI (sURI_Info *pURI);
u16 NDEF_ReadSMS(sSMSInfo *pSMS);
u16 NDEF_WriteSMS(sSMSInfo *pSMS);
u16 NDEF_ReadEmail(sEmailInfo *pEmailStruct);
u16 NDEF_WriteEmail(sEmailInfo *pEmailStruct);
u16 NDEF_ReadVcard(sVcardInfo *pVcard);
u16 NDEF_WriteVcard(sVcardInfo *pVcard);
u16 NDEF_ReadGeo(sGeoInfo *pGeo);
u16 NDEF_WriteGeo(sGeoInfo *pGeo);
```

Tag memory structure are different between TT1, TT2, TT3, TT4A, TT4B and vicinity card, the library needs to know which tag will be written or emulated to works. For this reason two global variables are used (namely st95mode and st95tagtype), and they can take the following values:

- st95mode
  - PICC
  - PCD
- st95tagtype
  - TT1
  - TT2
  - TT3
  - TT4A
  - TT4B
  - TT5.

Note that these values are automatically set when using higher level functions like “PICCEmul_InitPICCEmulation” or “ConfigManager_TagHunting”.

Examples are available in “menu_reader.c”, “menu_writter.c” and “menu_Tagemul.c” files.

## 4.4 RAM organization

Every tag type supported has its own buffer inside the RAM, they are all defined in the “main.c” file. They are used to store the content of the read/written tag (when using PCD functions) and to store the emulated tag content (when using PICC functions).

Content of the tag emulated can be saved in the flash memory of the STM32F1, as described in *Section 4.5: Flash management*.

## 4.5 Flash management

In order to use the flash for the card emulation the flag DISABLE_NDEF_FLASH has to be disabled (this is the case by default).

Before running the emulation, the function *initFlashNDEF()* has to be called first. Its role is to check if an existing message is available in the flash (magic number defined to manage first boot issue), if yes it will load it, otherwise it will format the memory by writing a default content.

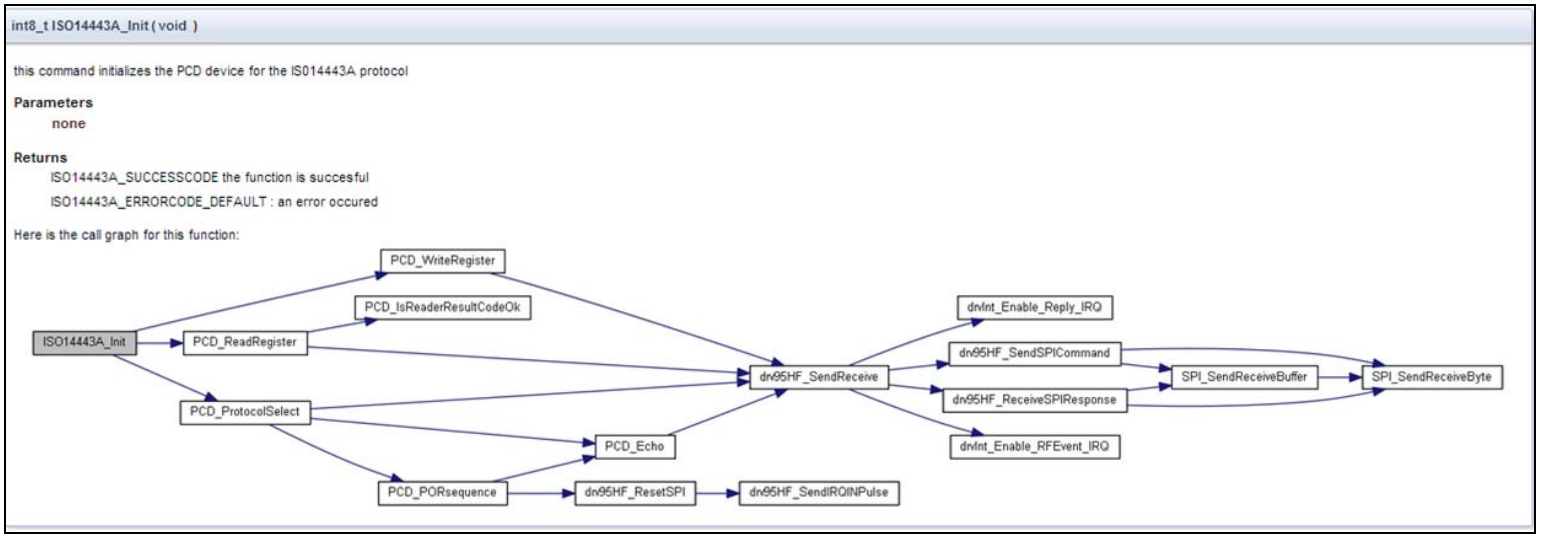
When a function to write the emulated tag is called (using RF or NDEF lib) the flag updateFlash is set. To save the changes to the flash, the function *manageFlashNDEF()* has to be called. It will check if the updateFlash flag is set, then it will write the selected tag to the flash (st95tagtype).

# 5 Firmware functions

You will find documentation (in html format) in the directory “Public document/doxygen” inside the firmware delivery: simply launch the file OpenST95HFDocumentation.bat, and your browser will display it.

An extract to illustrate the available content of this documentation is shown in *Figure 7*.

**Figure 7. Extract from doxygen**



# 6 Revision history

Table 1. Document revision history

| Date | Revision | Changes |
|---|---|---|
| 30-Jun-2014 | 1 | Initial release. |