UM2954

User manual

Getting started with X-LINUX-NFC6 package for the ST25R3916 high performance NFC front-ends

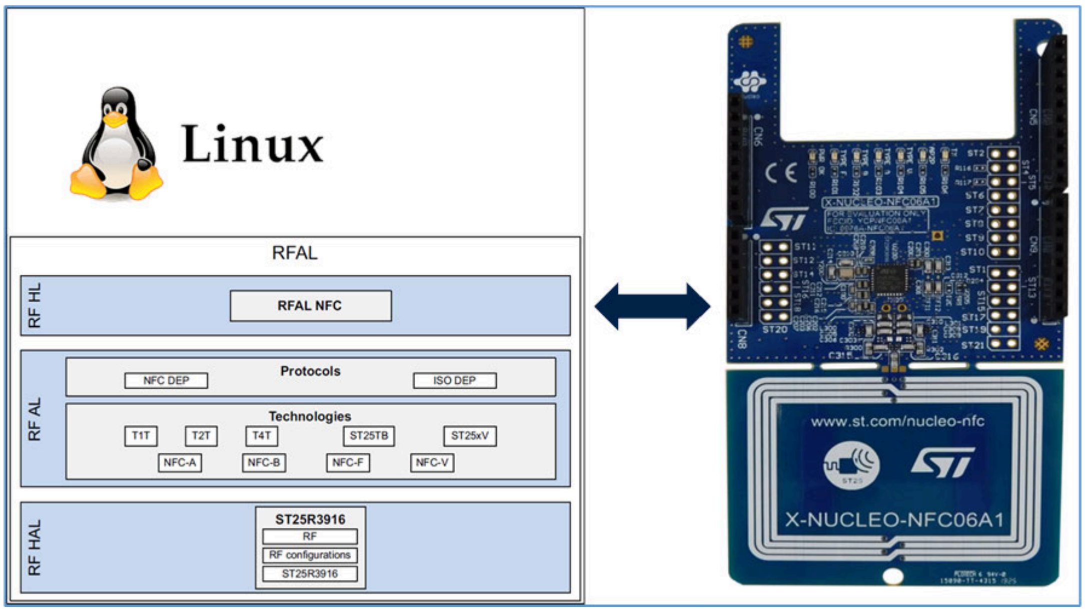
Introduction

This STM32 MPU OpenSTLinux software expansion package demonstrates the NFC/RF communication for a standard Linux system using our radio frequency abstraction library (RFAL). The RFAL common interface driver ensures that user function and application software is compatible with ST25R3916 NFC universal device.

The X-LINUX-NFC6 package ports the RFAL onto a discovery kit with STM32MP1 Series microprocessor running Linux to drive an ST25R3916 NFC front end on an STM32 Nucleo expansion board. The package includes a sample application to help you understand detection of different types of NFC tags and NFC-enabled mobile phones.

The source code is designed for portability across a wide range of processing units running Linux and supports all lower layers and some higher layer protocols of ST25R ICs to abstract RF communication.

Figure 1. Radio frequency abstraction library for Linux



1 Overview

1.1 Main features

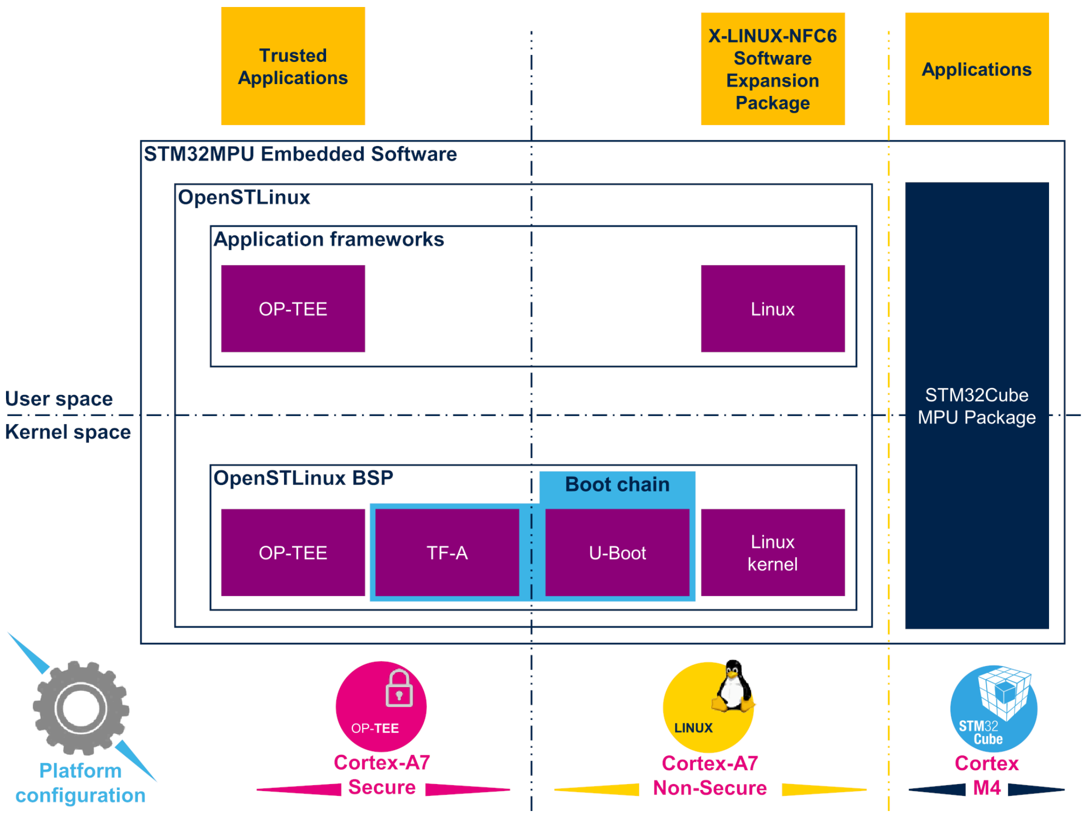
The X-LINUX-NFC6 software expansion package includes the following features:

- Complete Linux user space driver (RF abstraction layer) to build NFC enabled applications using the ST25R3916 NFC front-end IC.
- Linux host communication with the ST25R3916 via high-speed SPI or I²C interface (optionally configurable)
- Complete RF/NFC abstraction (RFAL) for all major technologies and higher layer protocols:
  - NFC-A (ISO14443-A)
  - NFC-B (ISO14443-B)
  - NFC-F (FeliCa)
  - NFC-V (ISO15693)
  - P2P (ISO18092)
  - ISO-DEP (ISO data exchange protocol, ISO14443-4)
  - NFC-DEP (NFC data exchange protocol, ISO18092)
  - Proprietary technologies (ST25TB, Kovio, B', iClass, Calypso, etc.)
- Sample implementation available for the X-NUCLEO-NFC06A1 expansion board plugged on an STM32MP157F-DK2
- Sample application to detect several NFC tag types and mobile phones supporting P2P

1.2 Package architecture

The software package runs on the A7 core of the STM32MP1 series microprocessor. The X-LINUX-NFC6 interacts with the lower layers libraries and SPI/I²C lines exposed by the Linux software framework.

Figure 2. X-LINUX-NFC6 application architecture in Linux environment



2 Hardware setup

Hardware requirements:

- PC/virtual-machine with Ubuntu® 18.04 or higher
- STM32MP157F-DK2 board (discovery kit)
- X-NUCLEO-NFC06A1
- 8 GB micro SD card to boot the STM32MP157F-DK2
- SD card reader/LAN connectivity
- USB Type-A to Type-micro-B USB cable
- USB Type-A to Type-C USB cable
- USB PD compliant 5 V 3 A power supply

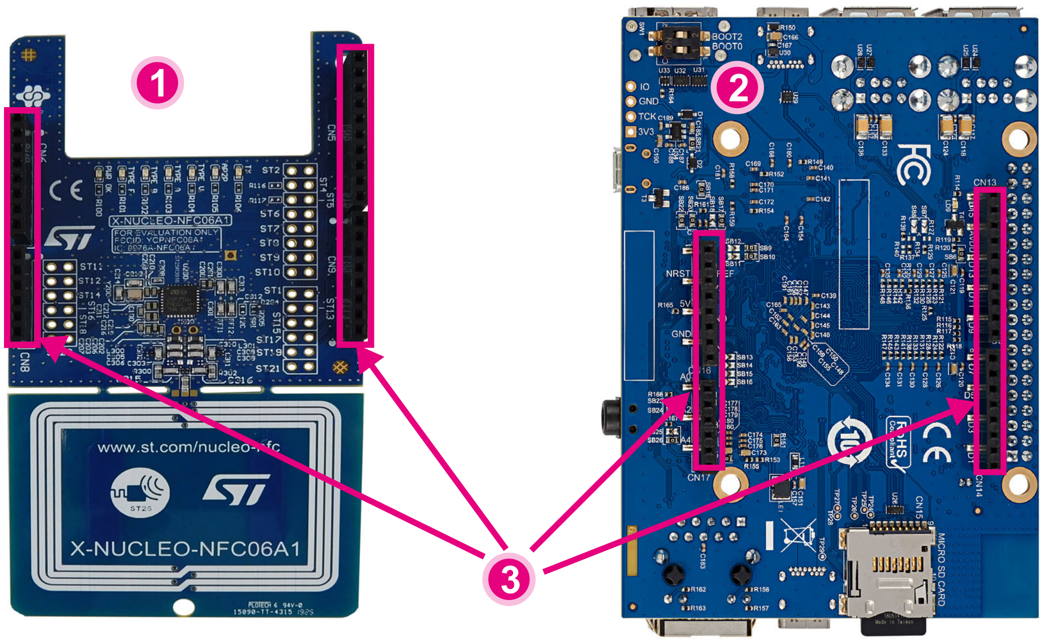
The PC/Virtual-machine forms the cross-development platform to build the RFAL library and application code to detect and communicate with NFC devices through the ST25R3916 IC.

2.1 How to connect the hardware using the SPI interface

Step 1. Plug the X-NUCLEO-NFC06A1 expansion board onto the Arduino connectors on the bottom side of the STM32MP157F-DK2 discovery board.

Figure 3. STM32 Nucleo expansion board and discovery board Arduino connectors

1. X-NUCLEO-NFC06A1 expansion board
2. STM32MP157F-DK2 discovery board
3. Arduino connectors



Step 2. Connect the ST-LINK programmer/debugger embedded on the discovery board to your host PC via the USB micro B type port (CN11).

Step 3. Power the discovery board through the USB Type C port (CN6).

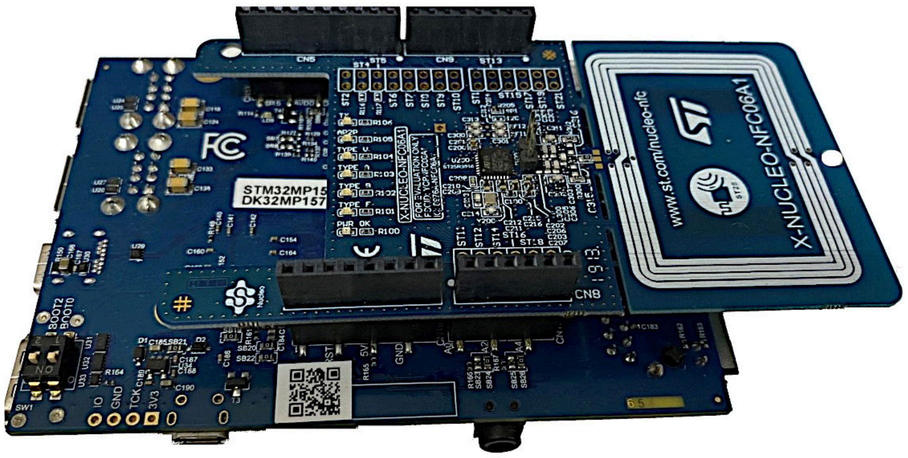


Figure 4. **Full hardware connection setup**

2.2 How to connect the hardware using the I²C interface

Step 1. Make below modifications on the X-NUCLEO-NFC06A1 expansion board so that the I²C interface pins are enabled.

a. ST2 and ST4 – soldered

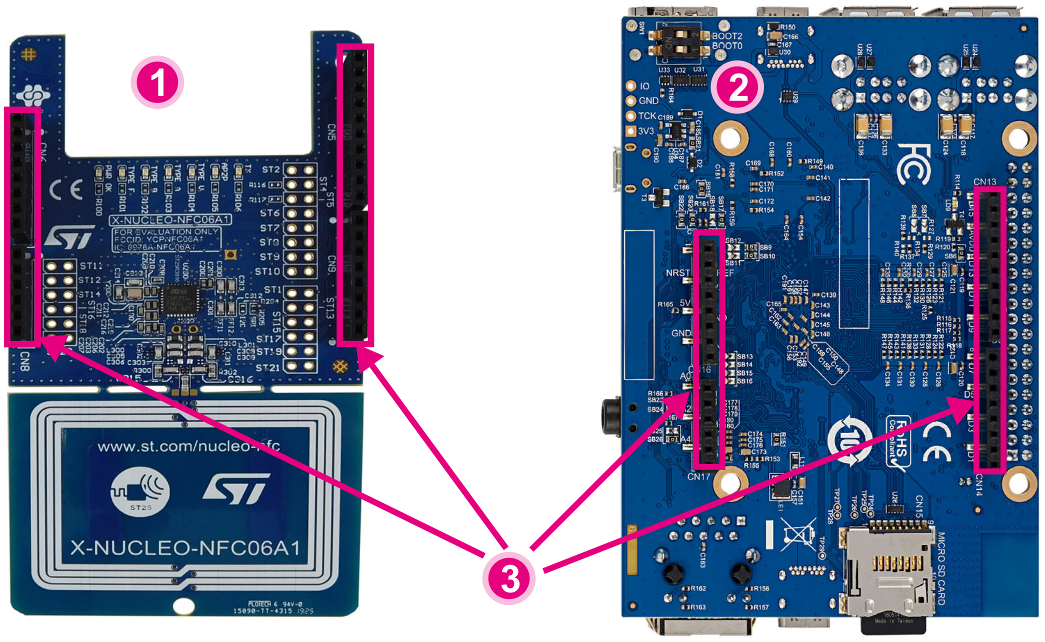
b. R204 – soldered 0 ohm resistor

c. R205 – de-soldered 0 ohm resistor

Step 2. Plug the X-NUCLEO-NFC06A1 expansion board onto the Arduino connectors on the bottom side of the STM32MP157F-DK2 discovery board.

Figure 5. **STM32 Nucleo expansion board and discovery board Arduino connectors**

1. X-NUCLEO-NFC06A1 expansion board
2. STM32MP157F-DK2 discovery board
3. Arduino connectors



Step 3. Connect the ST-LINK programmer/debugger embedded on the discovery board to your host PC via the USB micro B type port (CN11).

3 Software setup

Before you begin, power the STM32MP157F-DK2 discovery kit via a USB PD compliant 5 V, 3 A power supply and install the Starter Package according to the instructions in the STM32MP157x-DK2 Getting Started wiki. You need a minimum 8 GB microSD card to flash the bootable images.

To run the application, update the platform configuration by loading the correct device tree to enable the relevant peripherals. You can do this quickly by using the prebuilt images available, or you can compile the device tree and images from sources using the OpenSTLinux developer package.

You can also (optionally) build this software package by including the Yocto layer (meta-nfc6) in the ST distribution package. This operation builds the source code and includes the device-tree modifications along with compiled binaries in the final flashable images. For detailed steps describing the process, see Section 3.5 .

You can connect to the discovery kit from the host PC via TCP/IP network using `ssh` and `scp` commands, or through serial UART or USB links using tools like minicom for Linux or Tera Term for Windows.

Related links

Refer to these instructions for installing the starter package onto the discovery kit

3.1 Steps for quick evaluation of software (run the prebuilt binary) on page 5

3.2 How to update the platform configuration in the developer package on page 6

4 How to transfer files using Tera Term on page 10

Refer to this wiki page for alternative ways of estashing PC communication with the board

3.1 Steps for quick evaluation of software (run the prebuilt binary)

Step 1. Flash the starter package on the SD card.

Step 2. Boot the board with the starter package.

Step 3. Enable network connectivity on the board via Ethernet or Wi-Fi.

Step 4. Download the X-LINUX-NFC6 package, containing the prebuilt binaries and sources, from the ST website.

Step 5. Use the following commands to copy the device tree blob and update the new platform configuration.

If network connectivity is not available, you can transfer the files locally from your Windows PC to the discovery kit using Tera Term.

```
PC $> cd X-LINUX-NFC6_v2.0/STM32MP157F-DK2_DeviceTree/Binaries
PC $> scp stm32mp157f-dk2.dtb root@<ip address of board>:/boot/
PC $> ssh root@<ip address of board>
Board $> /sbin/depmod -a
Board $> sync
Board $> reboot
```

Step 6. To evaluate the demo application using the SPI interface, follow the below mentioned instructions. After the board boots up, copy the application binary and the shared lib to the discovery kit.

```
PC $> cd X-LINUX-NFC6_v2.0\NFCPollerApplication\Binaries\SPI
PC $> scp ./nfcpoller_st25r3916 root@<ip address of board>:/usr/bin
PC $> scp ./librfal_st25r3916.so root:<ip address of board>:/usr/lib
PC $> ssh root@<ip address of board>
Board $> cd /usr/bin
Board $> chmod +x nfc_poller_st25r3916
Board $> ./nfc_poller_st25r3916
```

The application starts running once these commands are executed.

Note: *In case network connectivity is not available, refer to Section 4 for instructions on how to transfer the file over serial connection. Alternatively, you can also use the USB drive to transfer the files.*

Step 7. To evaluate the demo application using the I²C interface, follow the below mentioned instructions. Ensure that the hardware has already been modified as mentioned in Section 2.2 How to connect the hardware using the I²C interface. After the board boots up, copy the application binary and the shared lib to the discovery kit.

```
PC $> cd X-LINUX-NFC6_v2.0\NFCPollerApplication\Binaries\I2C
PC $> scp ./nfcpoller_st25r3916 root@<ip address of board>:/usr/bin
PC $> scp ./librfal_st25r3916.so root:<ip address of board>:/usr/lib
PC $> ssh root@<ip address of board>
Board $> cd /usr/bin
Board $> chmod +x nfc_poller_st25r3916
Board $> ./nfc_poller_st25r3916
```

3.2 How to update the platform configuration in the developer package

In case there is a mismatch between the kernel version available on the discovery kit and the one used to create prebuilt images, instructions in Section 3.1 Steps for quick evaluation of software (run the prebuilt binary) might not work. Then, it becomes necessary to compile the images from the sources. This method also allows further customization of the device tree. Follow the procedure below to set up the development environment.

Step 1. Download the developer package and install the SDK in the default folder structure on your Ubuntu machine.

You can find the instructions here: Install SDK

Step 2. Open the device tree file ‘stm32mp157f-dk2.dts’ in the developer package source code and add the code snippet below to the file:

This updates the device tree to enable and configure the SPI4 and I2C5 driver interface.

```
&i2c5 {
    status = "okay";
};

&spi4 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&spi4_pins_b>;
    pinctrl-1 = <&spi4_sleep_pins_b>;
    /*status = "disabled";*/
    cs-gpios = <&gpioe 11 0>;
    status = "okay";
    spidev@0x00 {
        compatible = "semtech,sx1301";
        spi-max-frequency = <5000000>;
    reg = <0>;
    };
};
```

Step 3. Compile the developer package to get the stm32mp157f-dk2.dtb file and copy it to the /boot folder on the STM32MP157F-DK2 board.

Refer to the following link for help: Modify, rebuild and reload the Linux® kernel.

3.3 How to build the RFAL Linux application code

Before you begin, you have to download the SDK, install and enable it. Download the application from X-LINUX-NFC6.

Step 1. Open the "Linux_RFAL_st25r3916_v2.4.0\linux_demo\platform\Inc\st25r3916\platform.h" file and comment/uncomment the line-number 54, which states the `#define RFAL_USE_I2C` for using SPI or I²C as the communication interface between the MPU and the ST25R3916 IC.

Step 1a. Uncomment the #define to use the I²C interface.

Step 1b. Comment the #define to use the SPI interface.

Step 2. Run the commands below to cross-compile the code:

These commands build the following files:

- the example application: nfc_poller_st25r3916
- shared lib for running the example application: librfal_st25r3916.so

```
PC $> sudo apt-get install cmake
PC $> cd X-
LINUXNFC6_v2.0\NFCPollerApplication\Source\Linux_RFAL_st25r3916_v2.4.0\linux_demo\build
PC $> cmake ..
PC $> make
```

3.4 How to run the RFAL Linux application on STM32MP157F-DK2

Step 1. Ensure that the hardware has already been modified according to the choice of using the SPI or the I²C interface as mentioned in Section 2.2 How to connect the hardware using the I²C interface. Copy the generated binaries into the discovery kit using the below commands.

```
PC $> scp X-LINUX-NFC6_v2.0/NFCPollerApplication/Source/
Linux_RFAL_st25r3916_v2.4.0/linux_demo/build/nfc_poller/nfc_poller_st25r3916
root@<board ip address>:/usr/bin
PC $> scp X-LINUX-NFC6_v2.0/NFCPollerApplication/Source/
Linux_RFAL_st25r3916_v2.4.0/linux_demo/build/rfal/st25r3916/
librfal_st25r3916.so root@<board ip address>:/usr/lib
```

Step 2. Copy generated binaries onto the discovery kit using the below commands.

```
PC $> scp X-LINUX-NFC6_v1.0.0/NFCPollerApplication/Source/
Linux_RFAL_st25r3916_v2.4.0/linux_demo/build/nfc_poller/nfc_poller_st25r3916
root@<board ip address>:/usr/bin
PC $> scp X-LINUX-NFC6_v1.0.0/NFCPollerApplication/Source/
Linux_RFAL_st25r3916_v2.1.0/linux_demo/build/rfal/st25r3916/
librfal_st25r3916.so root@<board ip address>:/usr/lib
```

Step 3. Open terminal on the discovery kit board or use `ssh` login and run the application using the following commands.

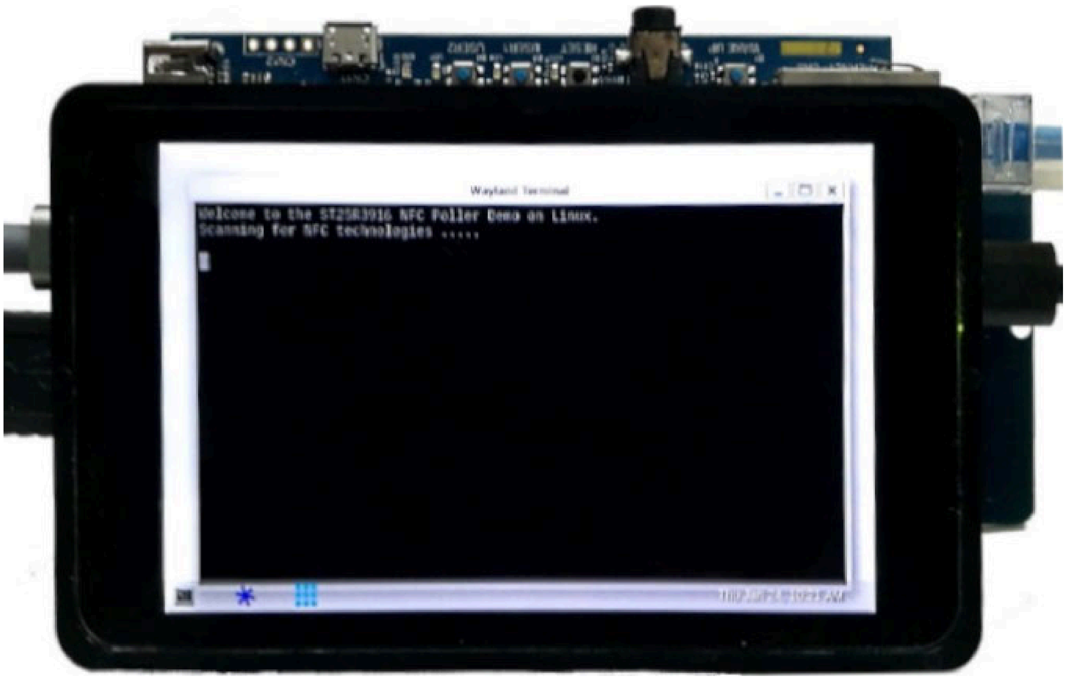
```
PC $> ssh root@<board ip address>
Board $> nfc_poller_st25r3916 # Run the application
```

After the execution of this code, the below message appears on the screen:

```
Welcome to the ST25R3916 NFC Poller Demo on Linux. Scanning for NFC
Technologies ......
```

Step 4. Bring an NFC tag near the NFC receiver to make the screen display the UID and NFC tag type.

Figure 6. **Discovery kit running the nfcPoller application**



```
Welcome to the ST25R3916 NFC Poller Demo on Linux.
Scanning for NFC technologies .....
```

3.5 How to include meta-nfc6 layer in the distribution package

Step 1. Download and compile the distribution package on your Linux machine.

Step 2. Follow the default directory structure suggested by ST wiki page to follow this document synchronously.

Step 3. Download the X-LINUX-NFC6 application package:

```
PC$> tar -xvf X-LINUX-NFC6_v2.0.tar.xz
PC$> cp -rf X-LINUX-NFC6_v2.0/NFCPollerApplication/Source/meta-
nfc6/ STM32MP15-Ecosystem-v4.0.0/Distribution-Package/openstlinux-5.15-yocto-
kirkstone-mp1-v22.06.15/layers
PC$> cd STM32MP15-Ecosystem-v4.0.0/Distribution-Package/openstlinux-5.15-yocto-
kirkstone-mp1-v22.06.15
```

Step 4. Open the "Linux_RFAL_st25r3916_v2.4.0.zip\Linux_RFAL_st25r3916_v2.4.0\linux_demo\platform\Inc\st25r3916\platform.h" file in the path Distribution-Package/openstlinux-5.15-yoctokirkstone-mp1-v22.06.15/layers/meta-nfc6/recipes-nfc6/nfc6/nfc6/ and comment/uncomment the line-number 54, which states `#define RFAL_USE_I2C` for using the SPI or the I²C as the communication interface between the MPU and the ST25R3916 IC.

Step 4a. Uncomment the #define to use the I²C interface.

Step 4b. Comment the #define to use the SPI interface.

Save the .zip file and proceed with the above steps.

Step 5. Set up the build configuration.

```
PC$> DISTRO=openstlinux-weston MACHINE=stm32mp1 source layers/meta-st/scripts/
envsetup.sh
```

Step 6. Add the meta-nfc6 layer to the build configuration of the distribution package configuration.

```
PC$> bitbake-layers add-layer ../layers/meta-nfc6
```

Step 7. Update the configuration to add new components in your image.

```
echo 'IMAGE_INSTALL:append = "nfc6"' >> ../layers/meta-st/meta-st-openstlinux/conf/
layer.conf
```

Step 8. Build your layer separately and then build the complete distribution layer.

```
PC$> bitbake st-image-weston
```

Note: *Building the distribution page for the first time may take several hours. However, it takes only few minutes to build meta-nfc6 layer and install the executables in the final images. Once the build is complete, the images are present in the following directory: build-<distro>-<machine>/tmp-glibc/deploy/images/stm32mp1.*

Step 9. Follow instructions on ST wiki page Flashing the built image to flash the new-built images onto the discovery kit.

Step 10. Run the application as mentioned in step 2 of Section 3.4 .

4 How to transfer files using Tera Term

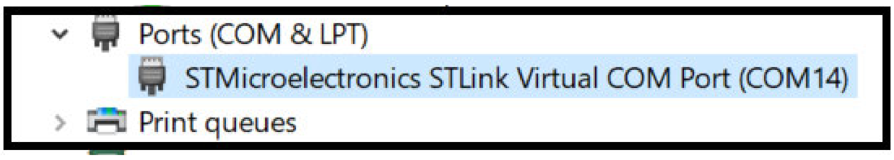
You can use a Windows terminal emulator application like Tera Term to transfer files from your PC to the discovery kit.

Step 1. Plug the power cable to power the board.

Step 2. Connect the discovery kit to your PC via the USB micro-B type connector (CN11).

Step 3. Check the virtual COM port number in the device manager.
In the screenshot below, the COM port number is 14.

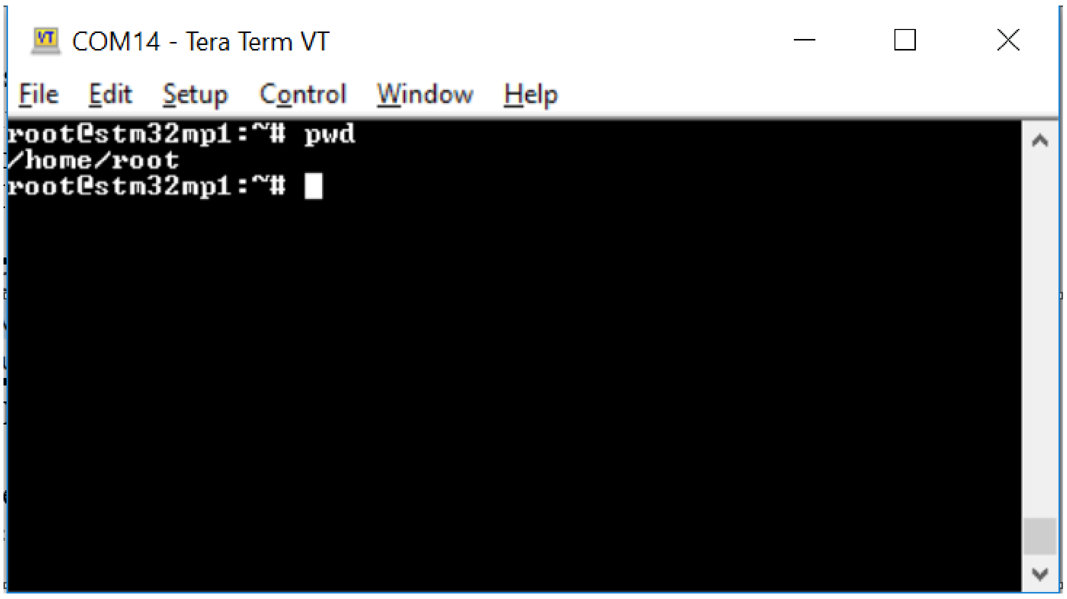
Figure 7. Screenshot of device manager showing virtual com port



Ports (COM & LPT)
STMicroelectronics STLink Virtual COM Port (COM14)
Print queues

Step 4. Open Tera Term on your PC and select the COM port identified in the previous step.
The baud rate should be 115200 baud.
The virtual terminal (remote access) appears as shown below.

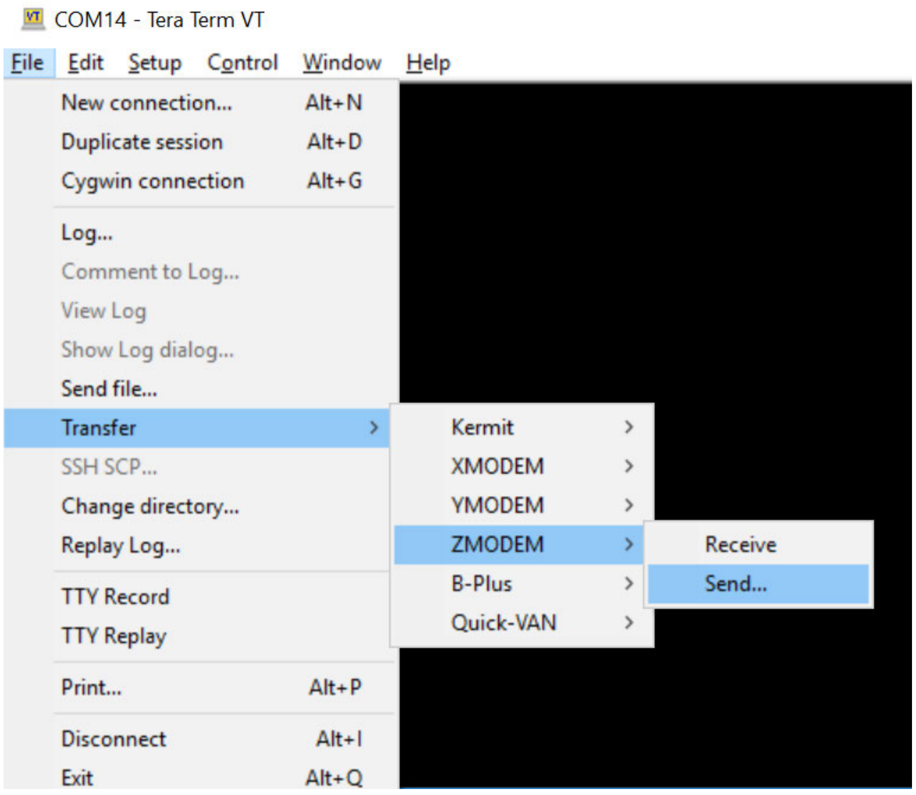
Figure 8. Snapshot of remote terminal via Tera Term



```
root@stm32mp1:~# pwd
/home/root
root@stm32mp1:~#
```

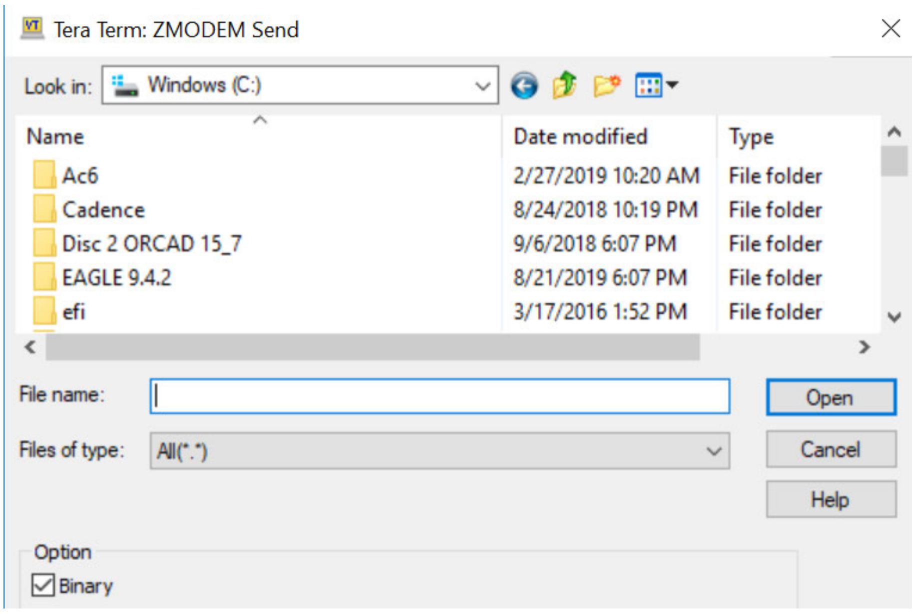
Step 5. To transfer a file from the host PC to the discovery kit, select [**File**]>[**Transfer**]>[**ZMODEM**]>[**Send**] in the top left corner of the Tera Term window.

Figure 9. **Tera Term file transfer menu**



Step 6. Select the file to be transferred in the file browser and select [**Open**].

Figure 10. File browser window for sending files



Tera Term: ZMODEM Send

Look in: Windows (C:)

Name	Date modified	Type
Ac6	2/27/2019 10:20 AM	File folder
Cadence	8/24/2018 10:19 PM	File folder
Disc 2 ORCAD 15_7	9/6/2018 6:07 PM	File folder
EAGLE 9.4.2	8/21/2019 6:07 PM	File folder
efi	3/17/2016 1:52 PM	File folder

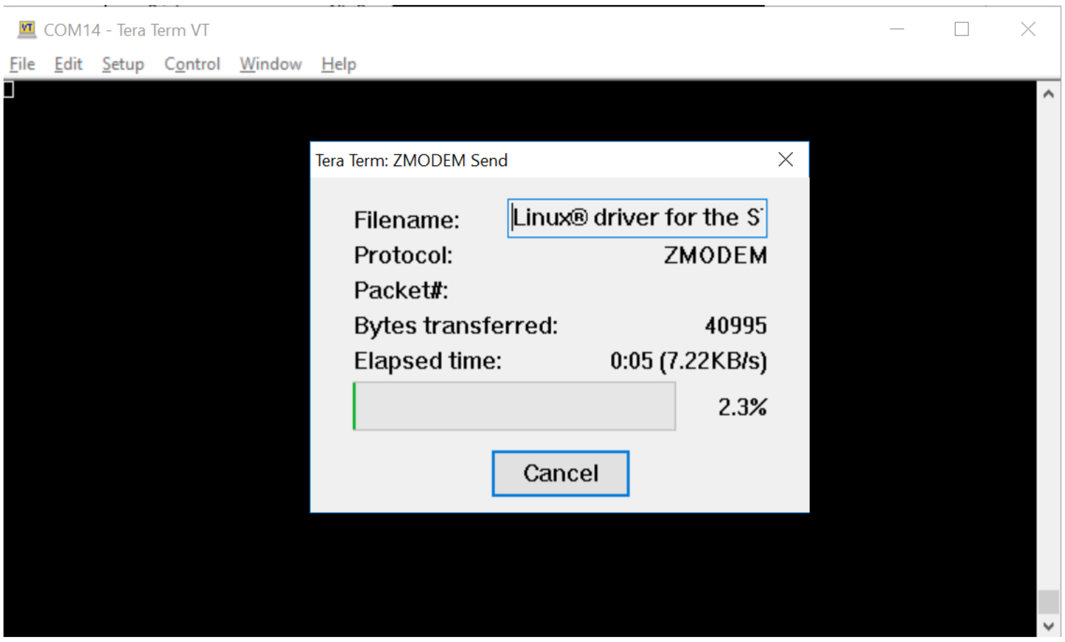
File name:

Files of type: All(*.*)

Option
☑ Binary

A progress bar shows the status of file transfer.

Figure 11. File transfer progress bar



COM14 - Tera Term VT

Tera Term: ZMODEM Send

Filename: Linux® driver for the S
Protocol: ZMODEM
Packet#:
Bytes transferred: 40995
Elapsed time: 0:05 (7.22KB/s)
2.3%

Revision history

Table 1. **Document revision history**

Date	Revision	Changes
02-Nov-2021	1	Initial release.
01-Dec-2022	2	Updated Section 1.1 Main features, Section 1.2 Package architecture, Section 2.1 How to connect the hardware using the SPI interface, Section 3.1 Steps for quick evaluation of software (run the prebuilt binary), Section 3.2 How to update the platform configuration in the developer package, Section 3.3 How to build the RFAL Linux application code, Section 3.4 How to run the RFAL Linux application on STM32MP157F-DK2, and Section 3.5 How to include meta-nfc6 layer in the distribution package. Added Section 2.2 How to connect the hardware using the I²C interface.

Contents

List of tables

List of figures