UM2954 X-LINUX-NFC6 Package for the ST25R3916 High Performance NFC Front-Ends User Manual

UM2954
User manual
Getting started with the X-LINUX-NFC6 package for the ST25R3916 high-performance NFC front-ends

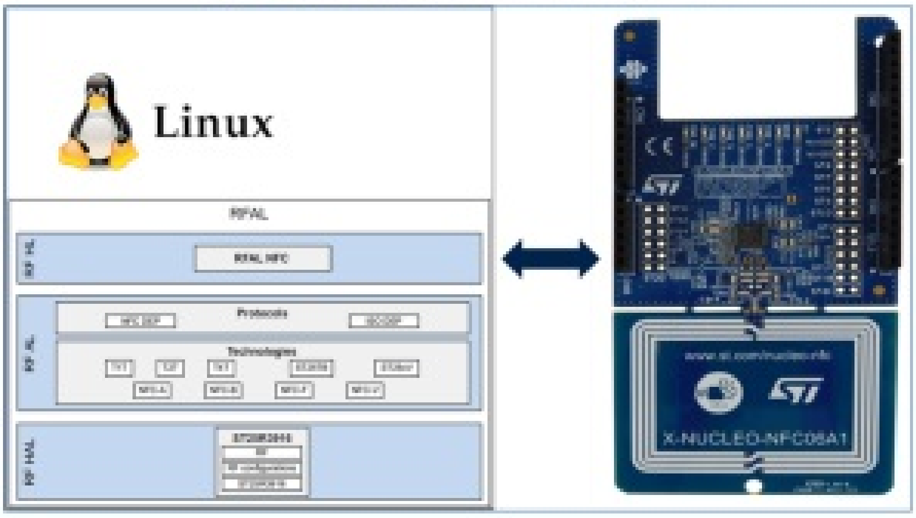
Contents

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 the 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



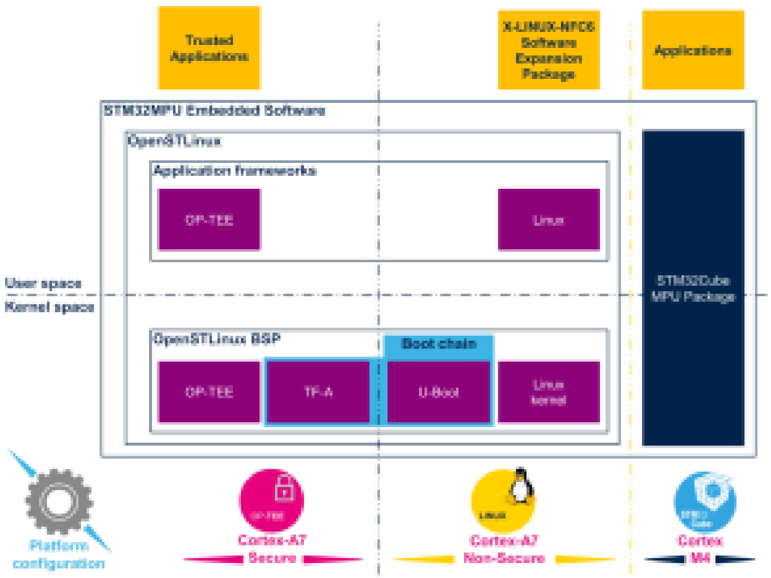
Overview

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 interface.
- 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 with 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 lines exposed by the Linux software framework.

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



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

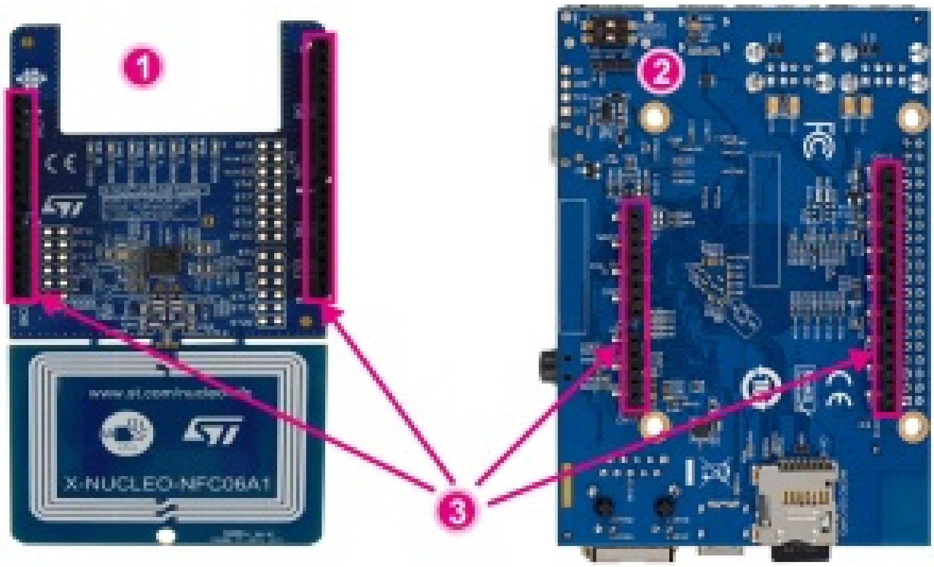
The PC/Virtual-machine forms the cross-development platform to build the RFAL library and application code to detect and communicate with NFC devices through the ST25R3916 IC.

2.1 How to connect the hardware

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**

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 updating the device tree to enable the relevant 0peripherals. You can do this quickly by using the prebuilt images available, or you can develop the device tree and build your own kernel images.
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 on page 5
3.2 How to update the platform configuration in the developer package on page 5
4 How to transfer files using Tera Term on page 8
Refer to this wiki page for alternative ways of stashing PC communication with the board
3.1 Steps for quick evaluation of software
Step 1. Flash the **starter package** on the SD card.

Step 2. Boot the board with the starter package.
Step 3. Enable Internet connectivity on the board via Ethernet or Wi-Fi.
Step 4. Download prebuilt images from the [X-LINUX-NFC6]() web page on 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_v1.0.0/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. After the board boots up, copy the application binary and the shared lib to the discovery kit.

```
PC $> cd X-LINUX-NFC6_v1.0.0\NFCPollerApplication\Binaries
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 of network connectivity is not available, refer to **Section 4** for instructions on how to transfer the file locally.

3.2 How to update the platform configuration in the developer package

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 driver interface.

```
&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.
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-LINUXNFC6.

Step 1. 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-LINUX-
NFC6_v1.0.0\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. 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 2. Open the 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 $> cd /usr/bin/  #enter directory where where binaries were copied
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 3. Bring an NFC tag is near the NFC receiver to make the screen display the UID and NFC tag type.

Figure 5. Discovery kit running the nfcPoller application

3.5 How to include a 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 the ST wiki page to follow this document synchronously.
Step 3. Download the [X-LINUX-NFC6](#) application package:

```
PC$> tar -xvf X-LINUX-NFC6_v1.0.0.tar.xz
PC$> cp -rf X-LINUX-NFC6_v1.0.0/NFCPollerApplication/Source/meta-nfc6/ STM32MP15-Ecosystem-v3.0.0/Distribution-Package/openstlinux-5.10-dunfell-mp1-21-03-31/layers
PC$> cd STM32MP15-Ecosystem-v3.0.0/Distribution-Package/openstlinux-5.10-dunfell-mp1-21-03-31/
```

Step 4. Set up the build configuration.

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

Step 5. Add the meta-nfc6 layer to the build configuration of the distribution package configuration.

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

Step 6. Update the configuration to add new components to your image.

```
PC$> echo 'IMAGE_INSTALL_append += "nfc6"' >> ../layers/meta-st/meta-st-openstlinux/conf/layer.conf
```

Step 7. 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 a few minutes to build the 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 8. Follow instructions on the ST wiki page **Flashing the built image** to flash the new-built images onto the discovery kit.
Step 9. Run the application as mentioned in step 2 of **Section 3.4**.

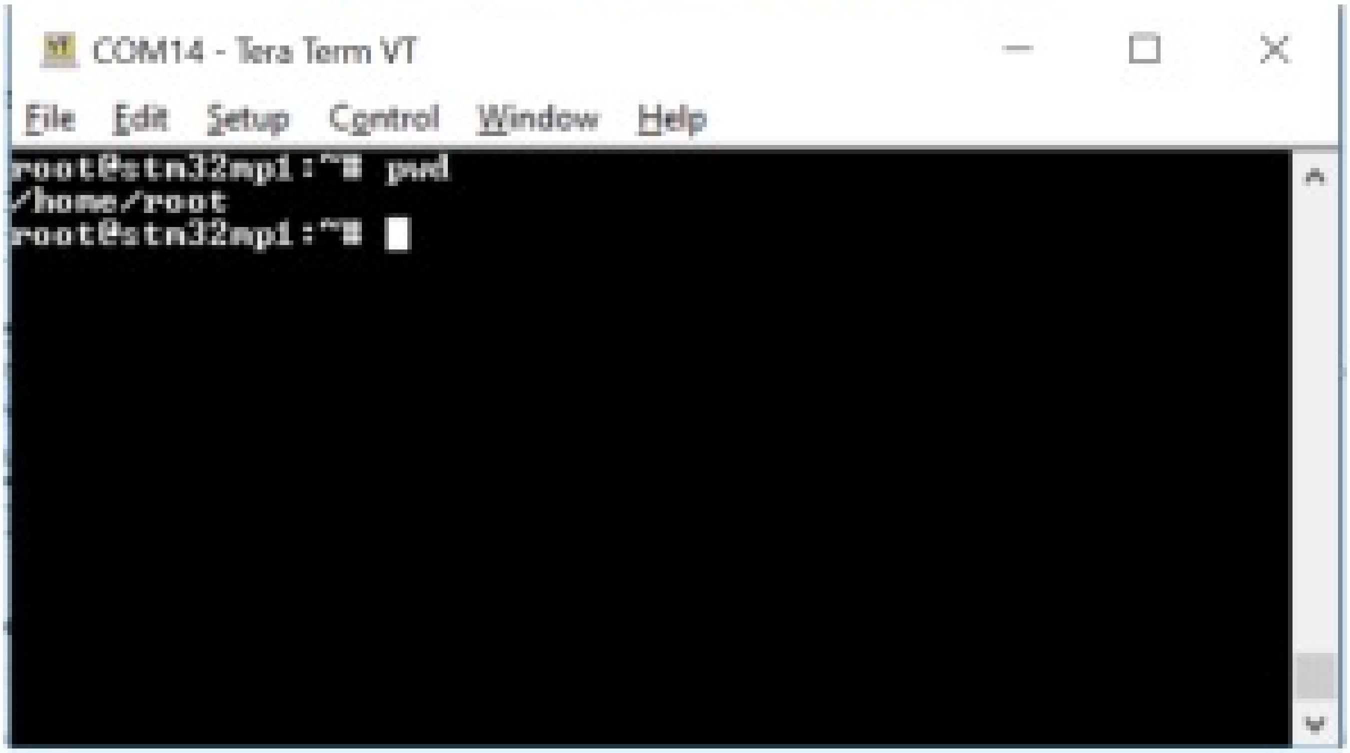
How to transfer files using Tera Term

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 6. **Screenshot of device manager showing virtual com port**

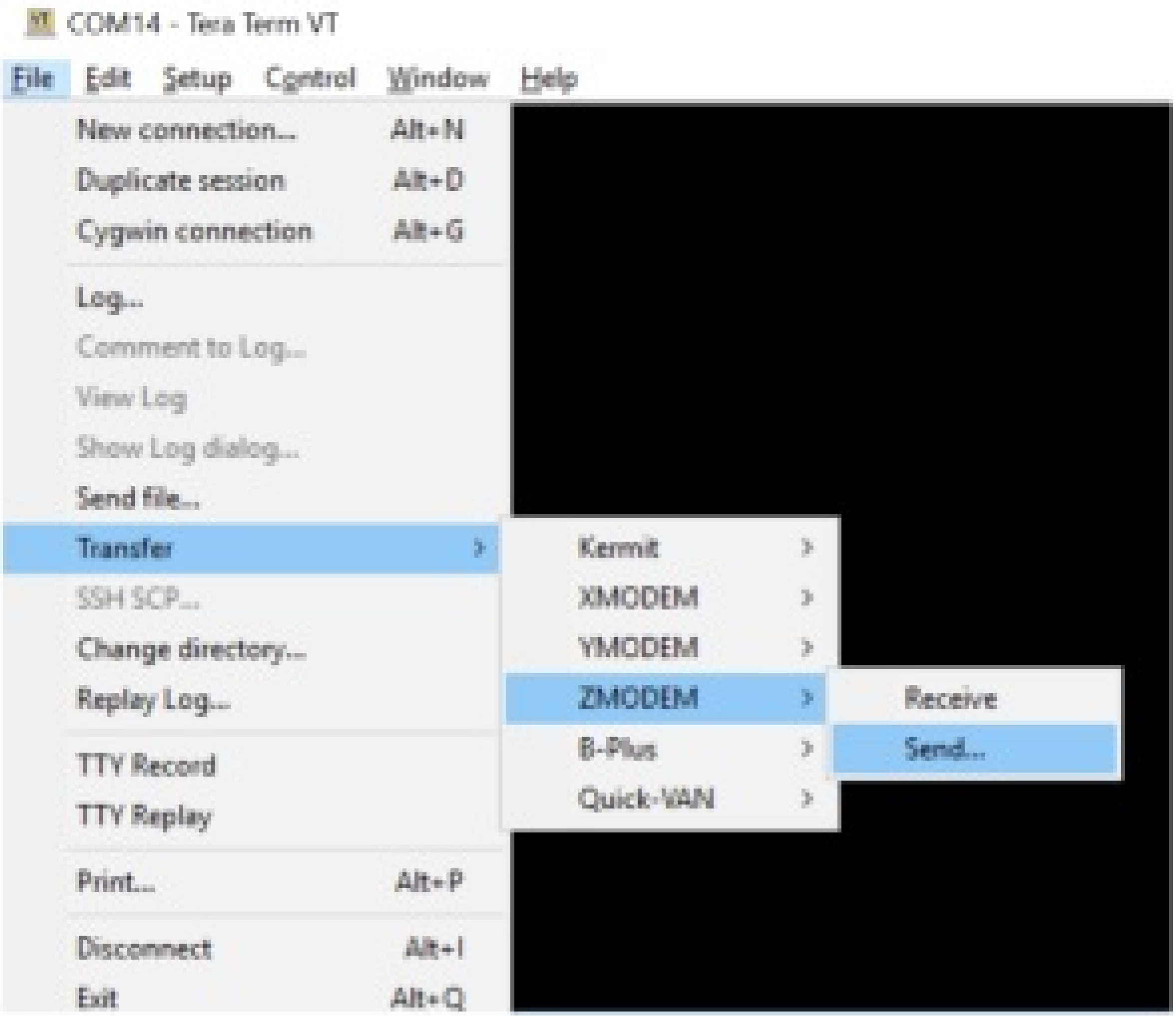
Step 4. Open Tera Term on your PC and select the COM port identified in the previous step.
The baud rate should be 115200 baud.
The virtual terminal (remote access) appears as shown below.

Figure 7. **Snapshot of remote terminal via Tera Term**



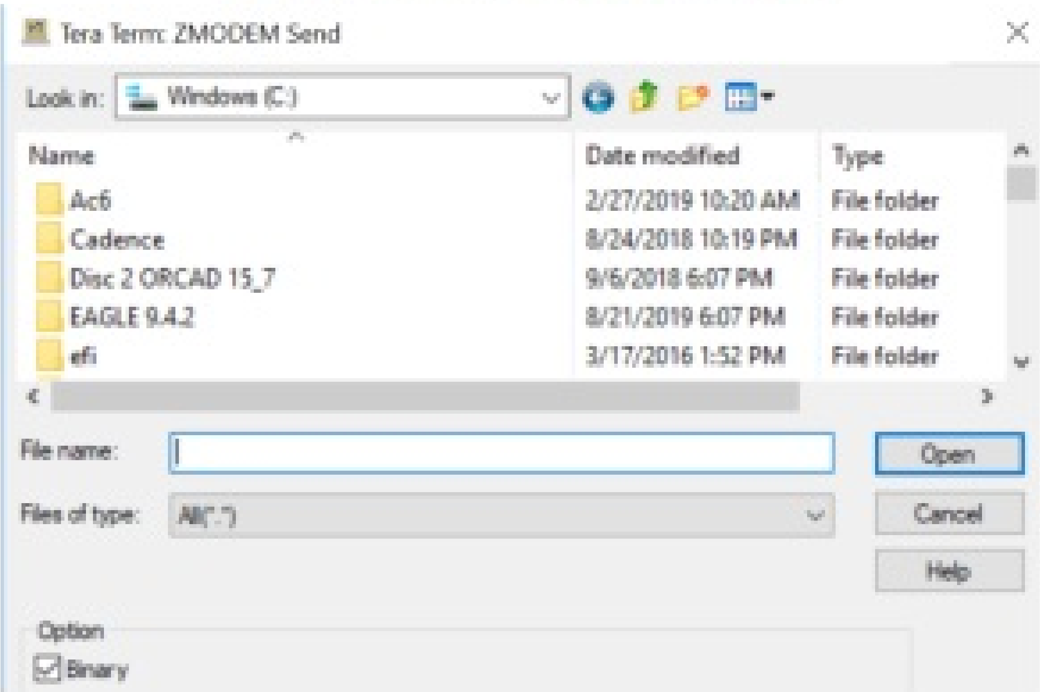
Step 5. To transfer a file from the host PC to the discovery kit, select [**File]>[Transfer]>[ZMODEM]>[Send]** in the top left corner of the Tera Term window.

Figure 8. **Tera Term file transfer menu**



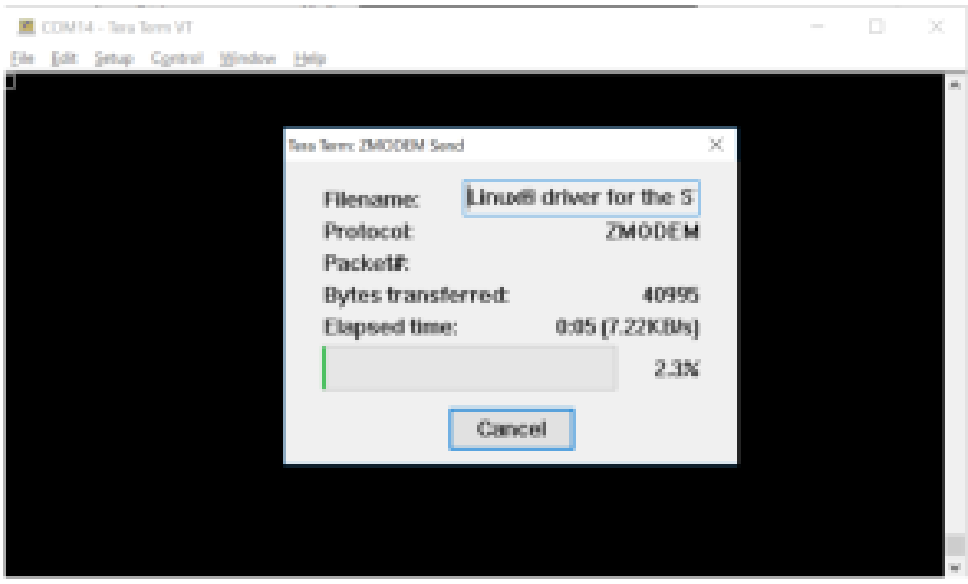
Step 6. Select the file to be transferred in the file browser and select **[Open]**.

Figure 9. **File browser window for sending files**



A progress bar shows the status of the file transfer.

Figure 10. File transfer progress bar



Revision history

Table 1. Document revision history

Date	Revision
2-Nov-21	1

List of tables

List of figures

IMPORTANT NOTICE – PLEASE READ CAREFULLY

For further information contact your local STMicroelectronics sales office.
www.st.com

Documents / Resources

	**ST UM2954 X-LINUX-NFC6 Package for the ST25R3916 High Performance NFC Front-Ends** [pdf] User Manual UM2954, X-LINUX-NFC6 Package for the ST25R3916 High Performance NFC Front-Ends, UM2954 X-LINUX-NFC6 Package for the ST25R3916 High Performance NFC Front-Ends, ST25R3916 High Performance NFC Front-Ends, High Performance NFC Front-Ends, NFC Front-Ends