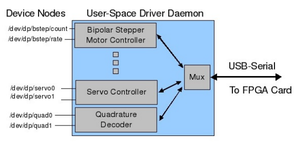
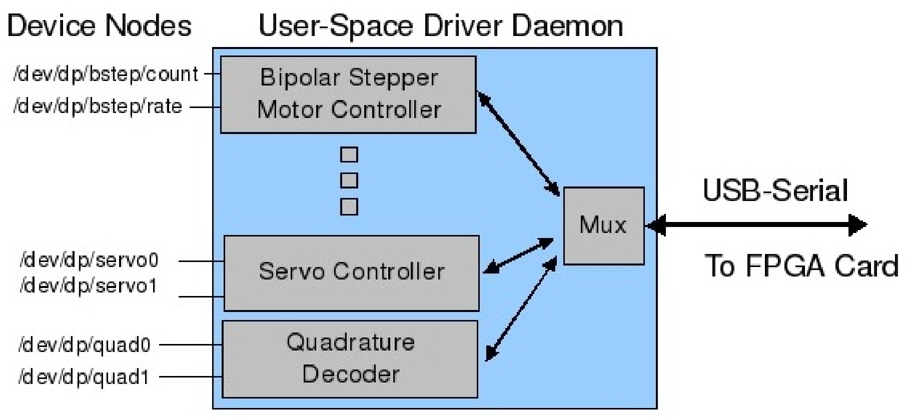


UM3078 ST25DVXXKC Linux User Space Driver User Manual

Contents

UM3078 ST25DVXXKC Linux User Space Driver



Introduction

This document shows how to use the STSW-ST25DV009 software package to control a ST25DVXXKC dynamic NFC tag from a Linux® platform. The STSW-ST25DV009 software package provides Linux® user space driver and some examples that can be configured to run on any Linux® platform. The ST25DVXXKC is an NFC dynamic tag, which can be managed by an RFID reader or by an NFC phone, it also has an I2C interface to communicate with an MCU or MPU. The ST25DVXXKC is available, for example, on the X-NUCLEO-NFC07A1 expansion board. Information and documentation related to the NFC components, the X-NUCLEO-NFC07A1 expansion board and the STSWST25DV009 software are available on www.st.com.

Purpose

ST25DVXXKC dynamic NFC/RFID tags are integrated circuits that can communicate with both:

- RFID readers and NFC phones, based on the ISO/IEC 15693 and NFC Forum Type 5 tag specifications.
- An MCU or MPU using an I2C interface.

These devices can be used on a Linux platform to enable a wireless communication, to easily transfer data from a Linux platform to a smartphone (for instance: URL, GPS coordinates, Out-Of-Band pairing data, and so on). The STSW-ST25DV009 software package provides the required code to control a ST25DVXXKC device from the user space of a Linux platform having an I2C controller.

Software structure

The STSW-ST25DV009 software is divided in several layers:

- ST25DVXXKC component driver
- Board support package
- NDEF library middleware
- Sample project codes

ST25DVXXKC component driver

The ST25DVXXKC component driver provides the methods to configure and control a ST25DVXXKC device. This part of the code is independent from the hardware, and it requires some basic IO functions to be implemented (see Section 2.2 Board support package) such as I2C read/write, gpio control. The ST25DVXXKC component driver files are in the Drivers/BSP/Components/ST25DVxxKC directory.

Board support package

The board support package implements two different aspects:

- The low-level IO functions called by the ST25DVXXKC component driver
- An API to the ST25DVXXKC component driver methods

The board support package files are in the Drivers/BSP/Linux directory.

Low-level IO functions

The low-level IO layer implements all the low-level functions required by the ST25DVXXKC driver. This layer is implemented in the Drivers/BSP/Linux directory with the files described in Table 1.

Files	Description
	These files implement the functions to:
st25dv-i2c_linux.c	• Configure, read and write the I2C interface
st25dv-i2c_linux.h	• Get a millisecond tick
	This code relies on the /dev/i2c-X file to take control of the I2C peripheral.
st25dv-i2c-gpo.c st25dv-i2c-gpo.h	These files implement the functions to configure and receive interruptions from the GPO pin of the ST25DVXXKC. This code uses the /dev/gpiochipX file and a dedicated thread to poll for an event on the GPIO .
st25dv-i2c-lpd.c st25dv-i2c-lpd.h	These files implement the functions to configure and control the low power down pin of the ST25DVXXKC. This code uses the following files to control the GPIO: •/sys/class/gpio/export •/sys/class/gpio/gpioXX/direction • /sys/class/gpio/gpioXX/value.

These functions are specific to the board used and must be adapted to the platform on which they are used (see Section 3 How to configure a board).

API to the ST25DVXXKC driver methods
This API is only a wrapper around the ST25DVXXKC component driver. It is implemented in Drivers/BSP/Linux/bsp_nfctag.c and Drivers/BSP/Linux/bsp_nfctag.h files.

NDEF library middleware
The NFC Forum defines a standard format to use when reading/writing an NFC device. This format is known as NDEF messages. The NDEF library implements high level methods to easily format data into a NDEF compliant manner. This STMicroelectronics library is delivered as a middleware, fully independent from the hardware and coming with an interface file to be implemented for the targeted platform.
In the STSW-ST25DV009 software package these interface files are implemented in:

- Projects\NDEF_URI\Src\lib_NDEF_config.c
- Projects\NDEF_BLUETOOTH\Src\lib_NDEF_config.c

The NDEF library middleware files are in the Middlewares/ST/lib_nfc directory

Sample projects
In this section, a short overview of the sample projects included in the STSW-ST25DV009 pack is provided. The sample projects:

- must be adapted to the targeted Linux platform (as explained in Section 3 How to configure a board)
- show to the user how to use the APIs to correctly initialize and use the dynamic NFC/RFID tag IC (ST25DVxxKC device)

The sample projects are in the ./Projects directory.

- **NDEF_URI**
  This application shows how to write a simple URI NDEF message to the ST25DVXXKC EEPROM using the NDEF lib middleware. A message is displayed when the message has been successfully written. A smartphone or a NFC reader can be used to read the NDEF_URI message.
- **NDEF_BLUETOOTH**
  This application shows how to write a Bluetooth® OOB NDEF message to the ST25DVXXKC EEPROM using the NDEF lib middleware. A message is displayed when the message has been successfully written. A smartphone or a NFC reader can be used to read the NDEF_BLUETOOTH message.
- **GPO (general purpose output)**
  This example shows how to enable and use the GPO. After initialization, an interrupt is programmed to detect field changes in proximity of the ST25DVXXKC. A message is displayed when the field is detected and when the field disappears.
- **I2CPROTECTION**
  This example shows how to create areas in the ST25DVXXKC and how to protect them. Text is displayed on the console.
- **LPD (low power down)**
  This example shows how to activate low power down (LPD) pin. By entering “1” or “0”, the LPD pin is activated or deactivated. When the LPD pin is activated, the ST25DVXXKC VCC is cut off, the power consumption is minimum and communication via I2C is not available.
  Note: This test cannot be run with the X-NUCLEO-NFC07A1 expansion board as the board does not connect such pin.
- **Mailbox**
  This example shows how to write a message into the mailbox and how to read mailbox status register of ST25DVXXKC device. The text is displayed.
- **I2CChannel**
  This example shows how to change I2C slave address and shows that writing a message into the mailbox and reading mailbox status register of the ST25DVXXKC device both work with new slave address. The text is displayed and I2C slave address is reverted to default value.
  Note: If user stops the application before its end, appropriate I2C slave address has to be used for subsequent communications with ST25DVXXKC.
- **I2CMode**
  This example shows how to change I2C slave mode (Normal/RF Off) and that with I2C slave mode set to ‘RFOFF’ no more NFC communication is handled whereas with I2C slave mode set to ‘Normal’ the NFC communication is processed.

How to configure a board

The board support package layer must be slightly adapted to the targeted Linux platform, in order to select the I2C peripheral to be communicate with ST25DVXXKC and the GPIOs is connected to the ST25DVXXKC GPO and LPD pins. All the required definitions are listed in the following file: Drivers/BSP/Linux/hwconfig.h.

Table 2. Hardware configuration definition:

Feature	**Define**	**Description**
I2C	ST25DV_I2C_NR	It defines the I2C peripheral number used to communicate with the ST25DVXXKC. The value is used to complete the path to the /dev/i2c-X file.
GPO	ST25DV_GPO_GPIOCHIP	It defines the GPIOCHIP number connected to the ST25DVXXKC GPO pin. The value is used to complete the path to the /dev/gpiochipX file.
	ST25DV_GPO_PIN	It defines the GPIO pin number of the GPIOCHIP connected to the ST25DVXXKC GPO pin.
LPD	ST25DV_LPD_PIN	It defines the global GPIO pin number connected to the ST25DVXXKC LPD pin. It is used to export this GPIO.
	ST25DV_LPD_DIRECTION	It defines the path to the Linux file defining the GPIO direction connected to the ST25DVXXKC LPD pin such as: /sys/class/gpio/gpioXX/direction
	ST25DV_LPD_VALUE	It defines the path to the Linux file defining the GPIO value connected to the ST25DVXXKC LPD pin such as: /sys/class/gpio/gpioXX/value

Hardware setup

Hardware requirements:

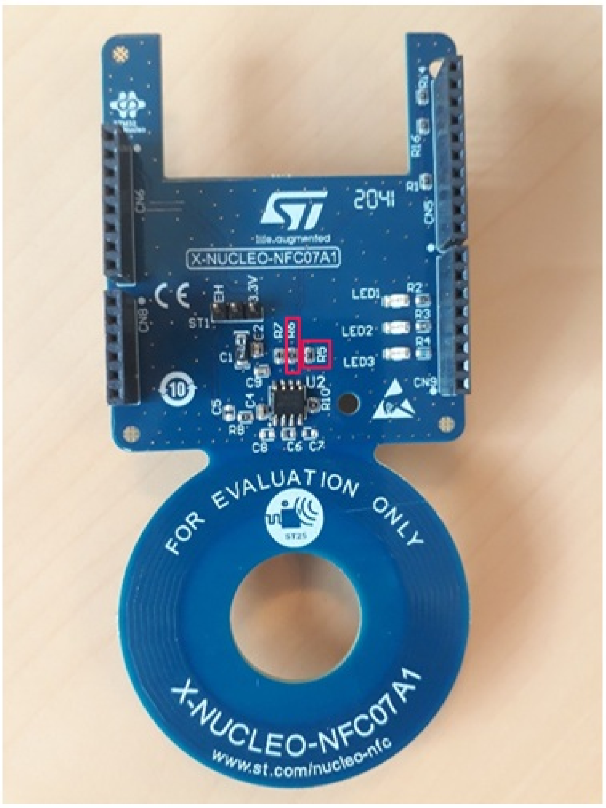
- Ubuntu-based PC/Virtual-machine version 16.04 or higher
- STM32MP157F-DK2 board (discovery kit)
- X-NUCLEO-NFC07A1
- 8 GB micro SD card to boot the STM32MP157F-DK2
- SD card reader / LAN connectivity
- USB Type-A to Type-micro B USB cable (optional)
- USB Type A to Type-C USB cable
- USB PD-compliant 5V 3A power supply

The PC/Virtual-machine forms the cross-development platform to build the sample projects application code. The hardware is connected as follows:

1. Depending on STM32MP157F-DK2 discovery board I2C configuration mode (normal, fast, fast+), the X-

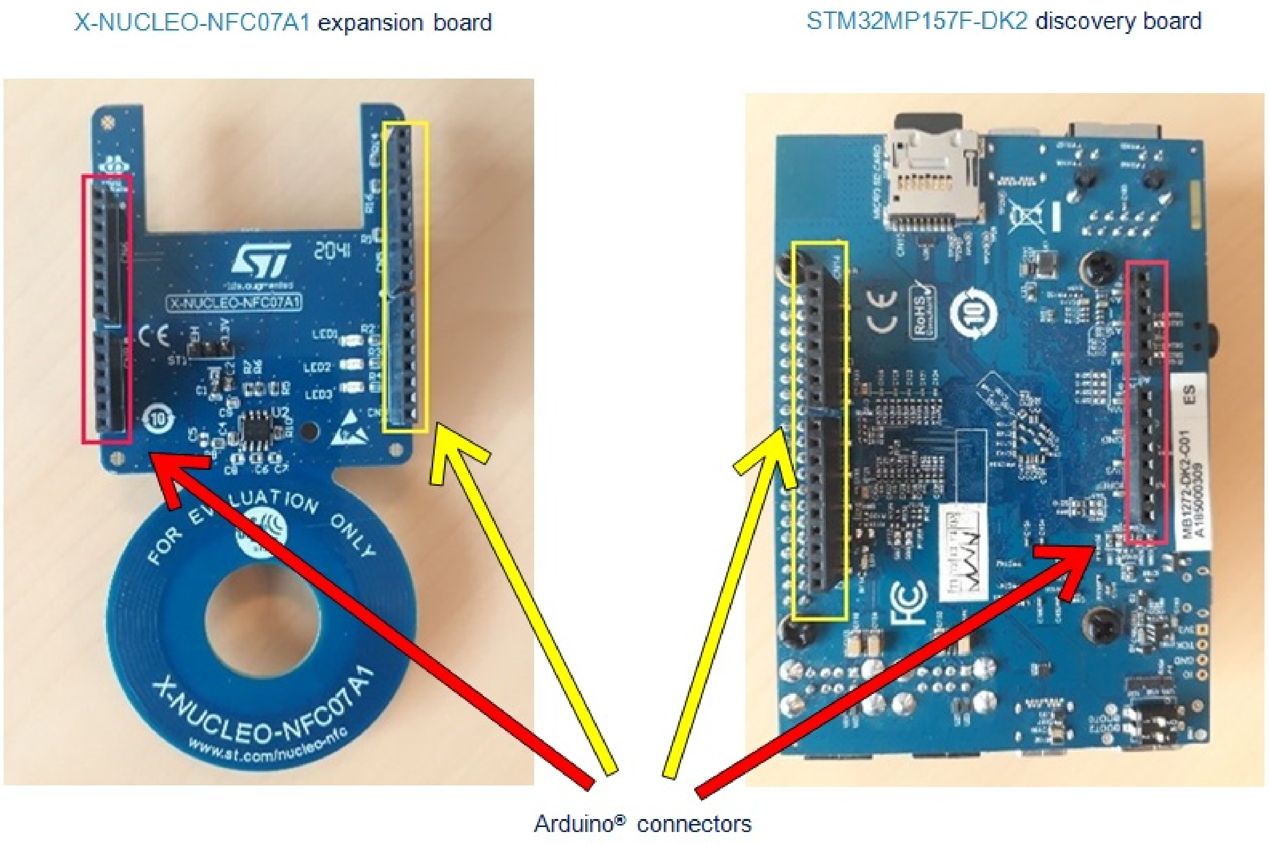
NUCLEO-NFC07A1 expansion board I2C pullup resistors (R5 and R6) may be increased up to 10 kΩ.

Figure 1. X-NUCLEO-NFC07A1 pull up resistors



2. Plug the X-NUCLEO-NFC07A1 expansion board onto the Arduino® connectors on the bottom side of the STM32MP157F-DK2 discovery board.

Figure 2. Nucleo board and discovery board Arduino® connectors


X-NUCLEO-NFC07A1 expansion board
STM32MP157F-DK2 discovery board
Arduino® connectors

3. If required, connect the ST-LINK programmer/debugger embedded on the discovery board to host PC via the

USB micro B type port (CN11).

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

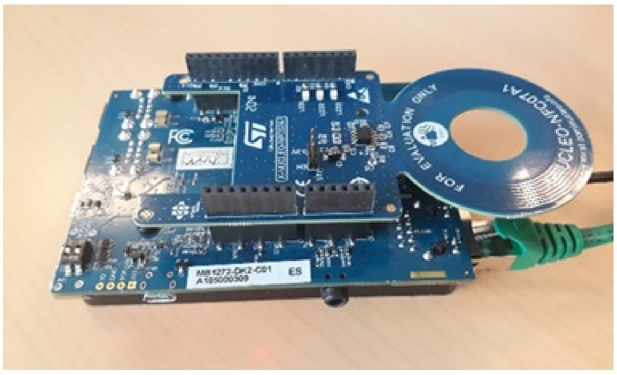


Figure 3. Full hardware connection setup

Compiling and running the sample projects

Each STSW-ST25DV009 sample project comes with a makefile and can be compiled using a C compiler like GCC. The pthread Linux library is used to create a thread detecting an event on the GPO line, this library is required for a correct linking at compilation time. Compilation and run procedures:

1. **On PC host:**
   - **copy all ST25DVLinux tree files to PC/Virtual-machine:** scp -r <user@linuxmachine>:.
2. **On PC/Virtual machine:**
   - **cross-compile the application (this generates statically linked executable file):** cd ~//Projects/ make clean all
   - **copy exe file to Linux target board (RPi, STM32MP157F-DK2, …):** scp ~//Projects//st25dv-i2c_ root@:.
3. **On STM32MP157F-DK2 board:**
   - **run the copied exe file:** chmod +x st25dv-i2c_ ./st25dv-i2c_

Revision history

IMPORTANT NOTICE – READ CAREFULLY

Documents / Resources

UM3078 User manual ST25DVXXKC Linux® user space driver **Introduction** This document shows how to use the STSW-ST25DV009 software package to control a ST25DVXXKC dynamic NFC tag from a Linux® platform. The STSW-ST25DV009 software package provides Linux® user space driver and some examples that can be configured to run on any Linux® platform. The ST25DVXXKC is an NFC dynamic tag, which can be managed by an RFID reader or by an NFC phone, it also has an I²C interface to communicate with an MCU or MPU. The ST25DVXXKC is available, for example, on the X-NUCLEO-NFC07A1 expansion board. Information and documentation related to the NFC components, the X-NUCLEO-NFC07A1 expansion board and the STSW-ST25DV009 software are available on www.st.com. UM3078 - Rev 1 - October 2022 For further information contact your local STMicroelectronics sales office. www.st.com	**ST UM3078 ST25DVXXKC Linux User Space Driver** [pdf] User Manual UM3078 ST25DVXXKC Linux User Space Driver, UM3078 ST25DVXXKC, UM3078, ST25DVXXKC, Linux User Space Driver, UM3078 Linux User Space Driver, ST25DVXXKC Linux User Space Driver, Linux User Driver, User Space Driver, Space Driver, Linux Driver, Driver

References

- **STMicroelectronics: Our technology starts with you**
- **STMicroelectronics Trademark List - STMicroelectronics**
- **STM32MP157x-DK2 - stm32mpu**

Manuals+,