AN5514
Application note

ST25 tags embedded library

Introduction

This document describes the ST25Tags middleware used in embedded C developments for reader applications based on ST25Rxx devices.

ST25Tags library is part of the ST25 NFC embedded library package (STSW-ST25R-LIB).

STSW-ST25R-LIB embedded software provides several middleware stacks and their associated examples, which can be reused when developing an application with ST25Rxx products. Refer to the STSW-ST25R-LIB webpage on *www.st.com* for more information.

Contents

List of tables

List of figures

1 ST25Tags library

The ST25Tags middleware is made up by independent files that provide easy access to ST25 tag features above the RFAL level.

The public ST25Tags libraries are described in the following sections of this document: the API for each library is explained, and application examples are presented.

The goal of files in the ST25Tags library is to offer to developers of applications based on STM32 microcontrollers (based on Arm®(a) Cortex® cores) a C application programming interface (API) that adds ST25 tag features on top of the RFAL RF abstraction layer for ST25R reader products.

The latest version of the library supports st25dv_pwm for the ST25DV-PWM series.

arm

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Example application setup

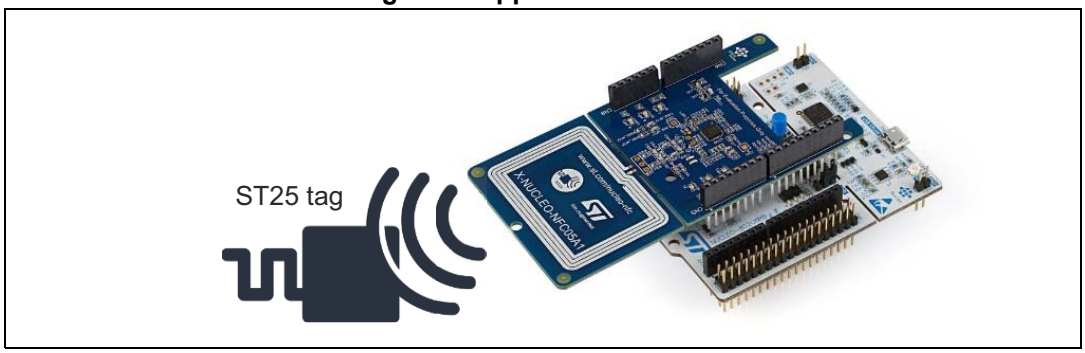
Example code typically runs on the NUCLEO-L476RG development board with an NFC reader shield board on top of it:

- X-NUCLEO-NFC03 for the ST25R95 reader
- X-NUCLEO-NFC05 for the ST25R3911B reader
- X-NUCLEO-NFC06 for the ST25R3916 reader

Some applications communicate via the UART. To see the traces, connect the USB port of the NUCLEO board to a PC and monitor communications via a serial console.

STM32CubeIDE project files are available for each supported Nucleo shield.

Figure 1. Application scheme



3 ST25DV-PWM API

For more information on how the ST25DV-PWM behaves refer to the device datasheet, available on *www.st.com.*

The ST25DV_PWM API is made up of two files located in Middlewares/ST/ST25Tags:

- Inc/st25dv_pwm.h: header file for the library, to include in your application
- Src/st25dv_pwm.c: source code for the ST25DV-PWM API

The ST25DV-PWM library is built on top of the ST25 reader RFAL stack.

3.1 API enums

Table 1. API enums

Enum name	Description
ST25TAGS_DVPWM_TagType_t	Type of ST25DV-PWM tag according to their EEPROM size and the number of PWM outputs
ST25TAGS_DVPWM_PwmNumber_t	Name of the PWM outputs
ST25TAG_DVPWM_SecuritySession_t	List of all possible security sessions
ST25TAG_DVPWM_UserAreaName_t	List of all possible user area names
ST25TAG_DVPWM_AccessProtection_t	List of options to protect access to areas
ST25TAGS_DVPWM_OutputPowerLevel_t	PWM power level
ST25TAGS_DVPWM_PwmCoexistenceWithRf_t	Defines behavior of PWM signal during RF commands

3.2 API global variables

The only global variable for this library is an array of characters that contains names of the different ST25DV-PWM tag types:

- char ST25TAGS_ST25DVPWM_tagName[][16];

ST25TAGS_ST25DVPWM_tagName contains the String equivalent of the ST25TAGS_DVPWM_TagType_t enum.

3.3 API macro

ST25TAGS_DVPWM_HAS_TWO_OUTPUTS(x) is the only macro defined for the ST25DV-PWM library.This macro takes a ST25TAGS_DVPWM_TagType_t parameter and returns true if the tag type supports two PWM outputs, false if there is only one PWM output.

3.4 API methods

This section describes all the methods declared in the st25dv-pwm.h header file.

ST25TAGS_DVPWM_isTagSt25DvPwm

Parameters [in] **uid**: pointer to UID (Unique IDentifier) byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.

Returns
- **true** if the UID belongs to a ST25DV-PWM product
- **false** if the UID does not belong to a ST25DV-PWM product

Description Determines if the UID parameter belongs to a ST25DV-PWM product according to the value of the ST product byte.

ST25TAGS_DVPWM_GetSt25PwmTagType

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.

Returns ST25TAGS_DVPWM_TagType_t value corresponding to the tag reference

Description Determines if the tag with the UID as parameter belongs to a ST25DV-PWM product.

ST25TAGS_DVPWM_OpenSecuritySession

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.

[in] **sessionType**: Value taken from ST25TAG_DVPWM_SecuritySession_t

[in] **pPassword**: pointer to a 4-byte or 8-byte character array.
- pPassword length used is 8 bytes for SINGLE_USER_AREA_SECURITY_SESSION
- pPassword length used is 4 bytes for all other sessions

Returns
- ERR_WRONG_STATE: RFAL not initialized or incorrect mode
- ERR_PARAM: Invalid parameters
- ERR_IO: Generic internal error
- ERR_CRC: CRC error detected
- ERR_FRAMING: Framing error detected
- ERR_PROTO: Protocol error detected
- ERR_TIMEOUT: Timeout error
- ERR_NONE: No error

Description Opens a security session.

Sends a presentPassword RF command through rfalST25xVPollerPresentPassword() in addressed mode using the UID provided as the first parameter and the password passed as the third parameter.

The security session is closed upon taking the tag off the RF field, or when presenting an incorrect password.

When AREA1 and AREA2 user areas are merged, using the command ST25TAGS_DVPWM_SetNumberOfUserAreas(uid, 1) the password for the session must be set to 64-bit length.

ST25TAGS_DVPWM_SetNumberOfUserAreas

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.
[in] **numberOfUserAreas**:
– set to 1 to merge Area1 and Area2 into a single user area configuration
– set to 2 to keep two distinct user areas

Returns
– ERR_WRONG_STATE: RFAL not initialized or incorrect mode
– ERR_PARAM: Invalid parameters
– ERR_IO: Generic internal error
– ERR_CRC: CRC error detected
– ERR_FRAMING: Framing error detected
– ERR_PROTO: Protocol error detected
– ERR_TIMEOUT: Timeout error
– ERR_REQUEST: Cannot be executed at the moment
– ERR_NONE: No error

Description Configures the number of user areas. Possible values are 1 and 2 (AREA0 and PWM are not counted).

Note A security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method may return ERR_REQUEST (unless no write is required).

ST25TAGS_DVPWM_ProtectUserArea

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.
[in] **areaName**: Area to protect
[in] **accessControl**: Level of protection to apply to the given area

Returns
– ERR_WRONG_STATE: RFAL not initialized or incorrect mode
– ERR_PARAM: Invalid parameters
– ERR_IO: Generic internal error
– ERR_CRC: CRC error detected
– ERR_FRAMING: Framing error detected
– ERR_PROTO: Protocol error detected
– ERR_TIMEOUT: Timeout error
– ERR_REQUEST: Cannot be executed at the moment
– ERR_NONE: No error

Description Sets Read/Write area access protection for a given area.

Note A PWM_CTRL_SECURITY_SESSION security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method may return ERR_REQUEST (unless no write is required).

ST25TAGS_DVPWM_EnablePwmOutputSignal

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.
[in] **pwmName**: Select PWM1 or PWM2 output.
[in] **enableOutput**:
– Set to true to enable the given PWM output.
– Set to false to disable the given PWM output.

Returns
– ERR_WRONG_STATE: RFAL not initialized or incorrect mode
– ERR_PARAM: Invalid parameters
– ERR_IO: Generic internal error
– ERR_CRC: CRC error detected
– ERR_FRAMING: Framing error detected
– ERR_PROTO: Protocol error detected
– ERR_TIMEOUT: Timeout error
– ERR_REQUEST: Cannot be executed at the moment
– ERR_NONE: No error

Description Enables or disables PWM1 or PWM2 output signals.

Note A PWM_CTRL_SECURITY_SESSION security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method may return ERR_REQUEST (unless no write is required).

ST25TAGS_DVPWM_SetPwmOutputSignal

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.
[in] **pwmName**: PWM signal selection [PWM1, PWM2]
[in] **frequencyInHertz**: PWM output frequency in range [489Hz, 31250Hz]
[in] **dutyCycle**: PWM signal duty cycle in range [0, 100]

Returns
– ERR_WRONG_STATE: RFAL not initialized or incorrect mode
– ERR_PARAM: Invalid parameters
– ERR_IO: Generic internal error
– ERR_CRC: CRC error detected
– ERR_FRAMING: Framing error detected
– ERR_PROTO: Protocol error detected
– ERR_TIMEOUT: Timeout error
– ERR_REQUEST: Cannot be executed at the moment
– ERR_NONE: No error

Description Programs a given PWM output signal with the wanted characteristics.

Note A PWM_CTRL_SECURITY_SESSION security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method returns ERR_REQUEST.

ST25TAGS_DVPWM_ConfigurePwmOutputPower

Parameters [in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.

[in] **pwmName**: PWM signal selection [PWM1, PWM2]

[in] **outputPowerLevel**: FULL (default), 75%, 50% or 25% of Imax level

Returns
- ERR_WRONG_STATE: RFAL not initialized or incorrect mode
- ERR_PARAM: Invalid parameters
- ERR_IO: Generic internal error
- ERR_CRC: CRC error detected
- ERR_FRAMING: Framing error detected
- ERR_PROTO: Protocol error detected
- ERR_TIMEOUT: Timeout error
- ERR_REQUEST: Cannot be executed at the moment
- ERR_NONE: No error

Description Configures PWM output driver level.

If the application does not require full power, it is possible to reduce the output drive capability independently through PWM_CFG trimming registers (PWM_CFG bits b1-b0 for PWM1 and PWM_CFG bits b3-b2 for PWM2).

Note A PWM_CTRL_SECURITY_SESSION security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method may return ERR_REQUEST (unless no write is required).

ST25TAGS_DVPWM_ConfigurePwmOutputDuringRfCommand

Parameters
[in] **uid**: pointer to UID byte array of the tag to be targeted. The expected length is 8 bytes and the last byte of the array must be equal to 0xE0.

[in] **pwmBehavior**: can take one of the following five values:

- PWM_OUTPUT_UNCHANGED: PWM output unchanged during RF command
- PWM_OUTPUT_QUARTER_POWER_FULL_FREQUENCY: PWM output drive level trimmed at 25%
- PWM_OUTPUT_FULL_POWER_LOW_FREQUENCY: PWM output frequency lowered below minimum
- PWM_OUTPUT_QUARTER_POWER_LOW_FREQUENCY: PWM output drive level and frequency reduced
- PWM_OUTPUT_HIGH_IMPEDANCE_DURING_RF: PWM output in high impedance during RF

Returns
- ERR_WRONG_STATE: RFAL not initialized or incorrect mode
- ERR_PARAM: Invalid parameters
- ERR_IO: Generic internal error
- ERR_CRC: CRC error detected
- ERR_FRAMING: Framing error detected
- ERR_PROTO: Protocol error detected
- ERR_TIMEOUT: Timeout error
- ERR_REQUEST: Cannot be executed at the moment
- ERR_NONE: No error

Description
Configures PWM output behavior during RF command.

Reduces the impact of PWM noise during RF commands.

Note
A PWM_CTRL_SECURITY_SESSION security session must be active for this command to execute correctly. If ST25TAGS_DVPWM_OpenSecuritySession() has not been called prior to this command, the method may return ERR_REQUEST (unless no write is required).

4 Demonstration application for the ST25DV-PWM library

An example of using the ST25DV-PWM API can be found in file Src/demo.c of directory ST25NFCLib\Projects\STM32L476RG-Nucleo\Applications\X-NUCLEO-NFC06A1\ST25TAGS_ST25DV_PWM\.

Method demoSt25DvPwm() calls all the available API methods for reference. It is invoked in the RFAL listener polling loop when a ST25DV-PWM tag is detected.

4.1 Code example: detecting the tag type

Two API methods are available to detect if a tag UID belongs to a ST25DV-PWM product:

```
uint8_t uid = nfcDevice->nfcid
/* Test if device is a ST25DV-PWM tag */
if ( ST25TAGS_DVPWM_isTagSt25DvPwm( uid ) ) {
    platformLog(" ST25DV-PWM tag detected.\r\n");
    demoSt25DvPwm(&nfcDevice->dev.nfcv);
}
```

In the example above, ST25TAGS_DVPWM_isTagSt25DvPwm(uid) returns true if the UID identifies the tag as a ST25DV-PWM product.

The second API gives more details about the ST25DV-PWM tag type:

```
ST25TAGS_DVPWM_TagType_t tagType;

/* Test if tag is ST25DV-PWM */
tagType = ST25TAGS_DVPWM_GetSt25PwmTagType( uid );
if (tagType == UNKNOWN_DVPWM_TAG) {
    platformLog("ST25DV PWM Tag not recognized.\r\n");
    return;
}
```

ST25TAGS_DVPWM_GetSt25PwmTagType() returns the type of detected ST25DV-PWM product, a feature useful to determine if it has one or two PWM outputs, or in getting its EEPROM size.

In addition, the macro ST25TAGS_DVPWM_HAS_TWO_OUTPUTS(tagType) is provided, it returns true if the ST25DV-PWM tagType has two PWM output signals available.

For convenience a global character array ST25TAGS_ST25DVPWM_tagName[tagType] returns a pointer to a preset string of characters matching the ST25TAGS_DVPWM_TagType_t name.

4.2 Code example: opening a security session

To change settings in the ST25DV-PWM configuration, or to update the value of any register, a security session must be opened:

```
uint8_t configPassword[] = {0, 0, 0, 0};
```

```
/* Open ST25DV-PWM Security Session to access configuration commands and
registers */
platformLog("Calling OpenSecuritySession to open a Configuration
session...\r\n");
errCode = ST25TAGS_DVPWM_OpenSecuritySession( uid,
CONFIGURATION_SECURITY_SESSION, configPassword );
```

Note that security sessions can also be opened on user areas and PWM signal control. Refer to the ST25DV-PWM datasheet for more details.

Without a prior call to ST25TAGS_DVPWM_OpenSecuritySession(), the other API methods return an error.

4.3 Code example: interacting with the user memory areas

Once the configuration session is open, the layout of the user memory can be changed.

Standard read/write block operations can be achieved using the rfalNfcvPoller API. The read and write operations may require their own security session to be open on the targeted user area, if the protection has been set to access memory only after a password presentation.

```
ReturnCode err;
uint16_t   rcvLen;
uint8_t    blockNum = 1;
uint8_t    rxBuf[ 1 + DEMO_NFCV_BLOCK_LEN + RFAL_CRC_LEN ];    /* Flags +
Data + CRC */
uint8_t *  uid;

/* RFAL example use: Read block using Read Single Block command */
err = rfalNfcvPollerReadSingleBlock( RFAL_NFCV_REQ_FLAG_DEFAULT, uid,
blockNum, rxBuf, sizeof(rxBuf), &rcvLen );
platformLog( " Read Block %d: %s %s\r\n\r\n", blockNum, (err != ERR_NONE) ?
"FAIL": "OK - Data = ", (err != ERR_NONE) ? "" : hex2Str( &rxBuf[1],
DEMO_NFCV_BLOCK_LEN) );
```

Above is an example of a readSingleBlock command at block address 0x01. Data read from tag memory is stored in the rxBuf byte array.

API methods are available to configure the user memory areas:

```
/* Configure memory into a single user area */
platformLog( "Set a single user area...\r\n" );
errCode = ST25TAGS_DVPWM_SetNumberOfUserAreas( uid, 1 );

/* Set AREA1 protection to Read: Always / Write: Always */
platformLog( " Set AREA1 to set Read: always / Write: protected by
password...\r\n" );
errCode = ST25TAGS_DVPWM_ProtectUserArea( uid, AREA_USER_1,
READABLE_AND_WRITE_PROTECTED_BY_PWD );
```

The example code above shows how to set one or two user memory areas. Access constraints for each area can be specified with the ST25TAGS_DVPWM_ProtectUserArea() method.

4.4 Code example: setting the shape of the PWM signal

The PWM signal generated by the ST25DV-PWM can be configured easily with the ST25TAGS_DVPWM_SetPwmOutputSignal() API. This method takes the frequency value in Hz and the duty cycle ratio necessary to generate a PWM signal:

```
    /* Configure the PWM outputs */
    platformLog( " Configuring PWM1 output...\r\n" );
    errCode = ST25TAGS_DVPWM_SetPwmOutputSignal( uid, DVPWM_OUTPUT_PWM1,
20000, 50 );
```

In the example above the tag generates a 50% duty cycle signal at 20 kHz on PWM1.

The output must be enabled to see the electric signal.

```
/* Enable PWM outputs */
platformLog( " Enabling PWM1 output...\r\n" );
errCode = ST25TAGS_DVPWM_EnablePwmOutputSignal( uid, DVPWM_OUTPUT_PWM1,
true );

if ( ST25TAGS_DVPWM_HAS_TWO_OUTPUTS(tagType) ) {
    platformLog( " Enabling PWM2 output...\r\n" );
   errCode = ST25TAGS_DVPWM_EnablePwmOutputSignal( uid, DVPWM_OUTPUT_PWM2,
true );
}
```

5 Running demonstrations on STM32NUCLEO hardware in the SMT32CubeIDE

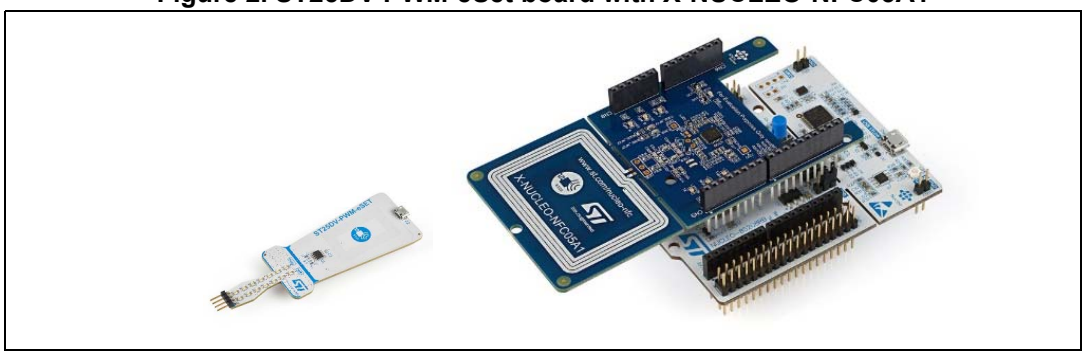
The STM32CubeIDE can be downloaded from *www.st.com*.

To open a project in the STM32CubeIDE environment, double click on either the .cproject or .project file located in the STM32CubeIDE directory for the corresponding Nucleo board. For example, the NFC06 (ST25R3916) project can be found at: ST25NFCLib\Projects\STM32L476RG-Nucleo\Applications\X-NUCLEO-NFC06A1\ inside folder ST25TAGS_ST25DV_PWM\STM32CubeIDE.

ST25DV-PWM example applications are best run with the ST25DV-PWM-eSet board to see the board LEDs turn on and off (any ST25DV-PWM tag behaves similarly).

Power on the ST25DV-PWM-eSet board by connecting it to a USB voltage supply, then connect the STM32-NUCLEO_L476RG + NFC shield (X-NUCLEO-NFC05 or X-NUCLEO-NFC06) to a USB port on your development PC.

Figure 2. ST25DV-PWM-eSet board with X-NUCLEO-NFC05A1



After opening the project file in STM32CubeIDE, build the project, then run or debug as STM32 Cortex-M C/C++ application (both operations can be performed by right-clicking on the project name in the Project Explorer window).

The example application waits for a ST25DV-PWM tag to be detected by the NFC reader. It then runs through each API method sequentially and displays the changes of the output signal duty cycle in the LED bar located on the eSet board.

6 Conclusion

STSW-ST25R-LIB embedded software provides middleware stacks and their associated examples.

This document describes the middleware and provides examples that can be reused when developing applications based on ST25Rxx products.

7 Revision history

Table 2. Document revision history

Date	Revision	Changes
16-Jun-2020	1	Initial release.