Application note

ST25DV64KC-DISCO firmware documentation

Introduction

The purpose of this firmware is to run demonstration boards based on the ST25DVxxKC dynamic NFC tag.

The ST25DVxxKC is a dynamic NFC tag with two different interfaces:

- NFC RF interface, compliant with standards:
  - NFC Forum Type 5 Tag
  - NFC Forum Type 5 Tag (ISO/IEC 15693)
- An I²C interface, controlled by an MCU (executing this firmware).

This tag also supports fast transfer mode (FTM) to enable faster communication between the MCU and the RF part.

1 ST25 discovery board

The ST25 discovery board is built around the STM32L476 MCU running this firmware. The board also embeds:

- an LCD display with a touchscreen to display and control the demonstration
- two USB connectors, one for the ST-Link, the second for user applications
- a connector for a tag antenna daughterboard (such as one of the ST25DV64KC antenna boards)
- optional modules:
  - an ST Bluetooth® Low Energy module
  - an ST WiFi® module.

1.1 ST25DV64KC antenna boards

These boards are built around the ST25DV64KC dynamic NFC tag and different NFC antennas.

1.2 Demonstration overview

The firmware implements three categories:

- The FTM demonstration boards, with different use cases benefiting from the faster communication between the MCU and a reader:
  - FW upgrade
  - picture download/upload
  - data transfers (from or to the reader)
  - stopwatch display
- The NDEF messages:
  - NFC types: URL, phone number, SMS, email, etc.
  - vCard
  - Android™ application record
  - proprietary record (MyAPP)
  - pairing with out of band records for Bluetooth LE and WiFi
- other features of the ST25DVxxKC demonstration boards:
  - RF interrupt through a dedicated GPO
  - energy harvesting from the RF to power an additional device
  - different states (RF off, RF sleep and low power down)
  - memory mapping and password protection
  - EEPROM writing time comparison between ST25DVxxK (emulated) and ST25DVxxKC. The firmware also implements a user interface consisting of:
    - a menu to select the demonstration boards
    - several screens to display instructions and status of the demonstration boards.

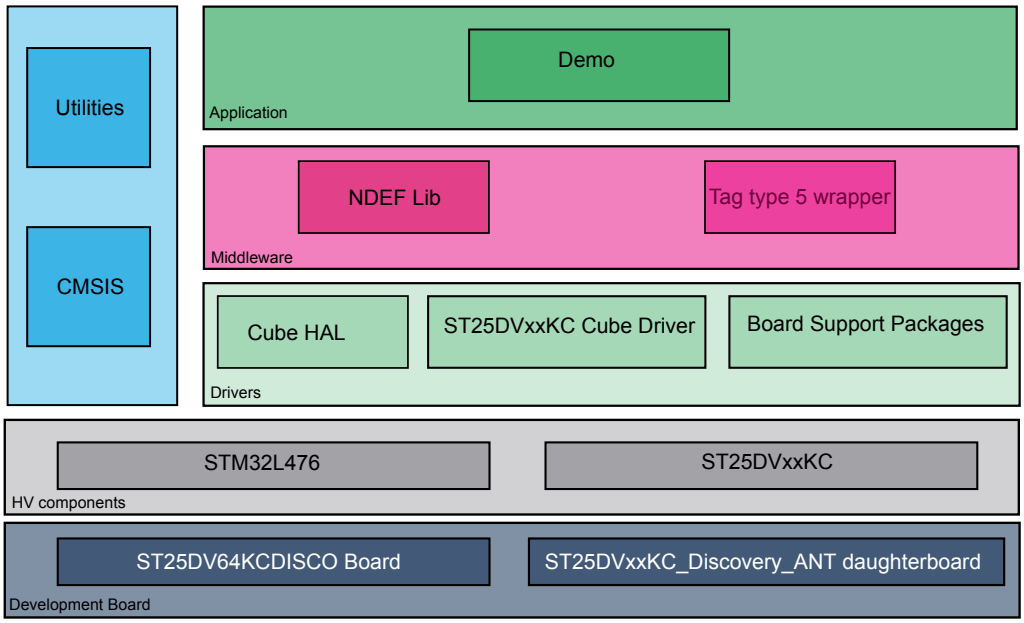
2 STM32Cube methodology

This firmware is designed to specifically run on the STM32L476 MCU embedded on the ST25 discovery board. Two important elements are specifically designed to make their reuse as straightforward as possible:

- The ST25DVxxKC driver, which implements the functions to control the ST25DVxxKC from its I²C interface, is completely independent from the MCU.

  This driver can be easily reused in other projects based on the STM32Cube methodology and in the HW solution interacting with an ST25DVxxKC.
- The NDEF library, which implements the standard NDEF protocol, is provided as STM32Cube middleware, fully independent from the HW.

Figure 1. ST25DVxxKC firmware overview



2.1 Firmware documentation structure

2.1.1 Main demonstration board modules

- **ST25 discovery demonstration board**: main module for all the ST25 discovery demonstration boards.
- **FTM**: this module provides functions to manage the protocol for the FTM demonstration boards.
- **NDEF**: this module implements the NDEF demonstration board functions.
- **ST25DV-I2C features**: this module implements the functions to execute specific features of the ST25DVxxKC.

2.1.2 FTM demonstration related modules

- **ST25DVxxKC mailbox functions**: this module offers a common API (application programming interface) for the ST25DVxxKC mailbox.
- **Flash command**: this module implements high level functions to write firmware or data to Flash memory.
- **Flash memory API** : this module defines an API to access the internal Flash memory.

2.1.3 ST25DVxxKC board support package and driver

- **Board support package**: container module for the BSP modules.
- **ST25 discovery NFC tag board support package**: this module provides high level functions to access the ST25DVxxKC NFC dynamic tag.
- **ST25DVxxKC driver**: this module implements the functions to drive the ST25DVxxKC NFC dynamic tag.

2.1.4 Menu of the demonstration board

- **ST25DV-I2C menu definition**: this module defines the structure and content of the ST25DV-I2C demonstration board menu.
- **ST25DV-I2C menu interface configuration**: interface file for the `menu_demo` middleware.
- **LibJPEG decode wrapper**: calls the libJPEG Cube middleware to decode jpeg pictures.

2.1.5 MCU support modules

- **ST25 discovery interrupt routines**: this module defines all the required interrupt routines for the ST25DVxxKC demonstration board.
- **`ST25Discovery_MSP`**:
  - `OPTIMIZE_OUTPUT_FOR_C= YES`
  - `INLINE_SIMPLE_STRUCT=YES`
  - `MAX_INITIALIZER_LINES=1`
  - `HIDE_UNDOC_MEMBERS=1`

3 Middlewares used in this firmware

This firmware relies on middlewares provided by ST or third party. These middlewares are hardware independent softwares implementing a generic feature.

This firmware uses the following middlewares:

- LibNDEF
- LibJPEG
- STM32 BlueNRG
- STM32 SPWF01SA
- Menu demonstration

3.1 LibNDEF

This library provides functions to read and write NDEF messages to a tag. It supports a variety of NDEF records, such as:

- URI record (includes URLs).
- SMS record.
- Email record.
- vCard record.
- Geolocation record.
- Bluetooth OOB record.
- WiFi OOB record.
- Android application record (AAR).

The library also defines a NFC-Forum Type 5 Tag wrapper to comply with the NFC-Forum Type5 Tag specification.

3.2 LibJPEG

This library implements the JPEG code. This firmware only includes the JPEG decoding part of the library.

3.3 STM32 BlueNRG

This middleware provides the communication stack for the ST BlueNRG module (Bluetooth Low Energy). A HID profile is used on top of it to remotely control a mouse pointer on a paired device.

3.4 STM32 SPWF01SA

This middleware provides the communication stack for the ST SPWF01 WiFi module.

It is used to set the WiFi module as a mini access point, to receive connections from remote devices.

3.5 Menu demonstration

This middleware implements functions to display an icon and text based menu.

It manages inputs from a touchscreen, a joystick and a button, to interact with the user.

4 ST25 discovery board support package

The board support package software (BSP) is defined by the STM32Cube methodology as the abstraction layer for the board specific features.

It implements all the functions required to access:

- the components on the board.
- the MCU peripherals requiring a board specific configuration.

The different parts of the BSP are described in the following sections.

4.1 IOBus

This part of the BSP implements the low-level functions interfacing between the components drivers and the MCU peripherals (by calling the STM32Cube HAL).

4.1.1 High level APIs

This parts of the BSP provides high level functions called by the application or middlewares to access the component drivers.

This layer acts as a bridge between the application and the component drivers.

4.2 Components

The ST25DV64KC discovery kit embeds different components requiring specific softwares to be correctly driven.

4.2.1 ST25DVxxKC driver

This driver provides all the required functions to access the ST25DVxxKC NFC dynamic tag.

4.2.2 ILI9341 driver

This driver implements functions to access the LCD display of the ST25 discovery board.

4.2.3 STMPE811 driver

This driver implements functions to access the touch screen of the ST25 discovery board.

4.2.4 AD5112 driver

This driver is used to access the digital potentiometer of the ST25 discovery board.

5 Module documentation

5.1 ST25DV-I2C common functions

This module proposes a generic API used for the ST25DV-I2C.

Data structures

- `struct IT_GPO_STATUS`
  ST25DV-I2C GPO interrupt status structure.

Macros

- `#define ST25_RETRY(cmd)`
  Iterate ST25DV-I2C command depending on the command return status.

Functions

- `ST25DV_I2CSSO_STATUS PresentPasswd (const bool passwd)`
  Present password to the ST25DV-I2C to open or close an I²C session.
- `int32_t InitITGPOMode (const uint16_t ITConfig)`
  Enable and initialize the GPO interrupt.
- `void DeInitITGPOMode (void)`
  Disable the GPO interrupt.
- `void ManageGPO (IT_GPO_STATUS *const gpo)`
  Reads the GPO interrupt source.
- `void st25_error (int32_t status)`
  Display an error message depending on the return value of a NFCTAG driver function.

Variables

- `uint8_t GPO_Activated`
  Polling variable for the ST25DV-I2C GPO interrupt, updated from GPO interrupt callback.

5.1.1 Detailed description

This module proposes a generic API used for the ST25DV-I2C.
Common functions for the ST25DV-I2C management.
The module covers password presentation, GPO init and management.

5.1.2 Data structure documentation

struct IT_GPO_STATUS

ST25DV-I2C GPO interrupt status structure.
This structure is used to return the event(s) that raised the GPO interrupt.

Table 1. Data fields

uint8_t	FieldOff	RF field becomes inactive
uint8_t	FieldOn	RF field becomes active
uint8_t	MailboxMsgRead	Fast Transfer Mode: message read by the RF
uint8_t	MsgInMailbox	Fast Transfer Mode: message from the RF received

uint8_t	RfActivity	The tag is processing on RF side
uint8_t	RfInterrupt	Interrupt generated by a dedicated RF command
uint8_t	Rfuser	GPO level controlled through the RF
uint8_t	WriteInEEPROM	EEPROM has been written by the RF

5.1.3 Macro definition documentation

ST25_RETRY

`#define ST25_RETRY(cmd)`

Iterate ST25DV-I2C command depending on the command return status.

Table 2. ST25_RETRY parameters

cmd	A ST25DV-I2C function returning a int32_t status.

5.1.4 Function documentation

DeInitITGPOMode()

`void DeInitITGPOMode (void )`

Disable the GPO interrupt.

Table 3. DeInitITGPOMode() parameters

None	No parameters.

Returns

None.

InitITGPOMode()

`int32_t InitITGPOMode (const uint16_t ITConfig )`

Enable and initialize the GPO interrupt.

Table 4. InitITGPOMode() parameters

ITConfig	Value of the interrupt register to configure.

Returns

int32_t status.

ManageGPO()

`void ManageGPO (IT_GPO_STATUS *const gpo )`

Reads the GPO interrupt source.

This function reads the interrupt status register from the ST25DV-I2C to report which interrupt(s) occured.

Table 5. ManageGPO() parameters

gpo	Pointer on IT_GPO_STATUS structure, to return the status of the GPO irq.

Returns

None.

PresentPasswd()

`ST25DV_I2CSSO_STATUS PresentPasswd (const bool passwd )`

Present password to the ST25DV-I2C to open or close an I²C session.

Table 6. **PresentPasswd() parameters**

passwd	TRUE open session, FALSE close session.

Returns

ST25DV_I2CSSO_STATUS status.

st25_error()

`void st25_error (int32_t status )`

Display an error message depending on the return value of a NFC tag driver function.

Table 7. **st25_error() parameters**

status	Return value from a NFCTAG driver function.

5.2 Flash memory API

This module defines an API to access the internal Flash memory.

Functions

- `void FLASH_If_FlashLock (void)`
  Locks the Flash memory to disable the flash control register access.
- `void FLASH_If_FlashUnlock (void)`
  Unlocks the Flash memory to enable the flash control register access.
- `FlagStatus FLASH_If_ReadOutProtectionStatus (void)`
  Gets Flash memory readout protection status.
- `uint32_t FLASH_If_GetPage (uint32_t Addr)`
  Gets the page of a given address.
- `uint32_t FLASH_If_GetBank (uint32_t Addr)`
  Gets the bank of a given address.
- `uint32_t FLASH_If_MassErase (const uint32_t Address)`
  Erases the required Flash memory sectors computed with destination address.
- `uint32_t FLASH_If_PageErase (const uint32_t Address, const uint32_t LastAddress)`
  Erases the required Flash memory pages computed with destination address.
- `void FLASH_If_DMA_Init (void)`
  Configure the DMA controller for SRAM to Flash memory transfer.
- `void FLASH_If_DMA_DeInit (void)`
  UnConfigure the DMA controller for SRAM to Flash memory transfer.
- `uint32_t FLASH_If_DMA_WriteBuffer (const uint32_t Address, const uint32_t *const pData, const uint32_t Size)`
  Writes a data buffer in the Flash memory through the DMA.
- `uint32_t FLASH_If_DMA_IT_WriteBuffer (const uint32_t Address, const uint32_t *const pData, const uint32_t Size)`
  Writes a data buffer in the Flash memory through the DMA.
- `uint32_t FLASH_If_WriteDWord (const uint32_t Address, const uint64_t pData)`
  Writes a data in Flash memory (data are 32-bit aligned).
- `uint32_t FLASH_If_WriteBuffer (const uint32_t Address, const uint64_t *const pData, const uint32_t Size)`
  Writes a data buffer in the Flash memory.
- `uint32_t FLASH_If_ConfBFB2 (void)`
  Set/Reset BFB2 bit to select boot from Flash memory bank.

- `void HAL_DMA_MspInit (void)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for DMA.
- `void HAL_DMA_MspDeInit (void)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for DMA.

5.2.1 Detailed description

This module defines an API to access the internal Flash memory. The module covers following functions:

- Lock Flash memory modification.
- Unlock Flash memory modification.
- Read Flash memory protection status.
- Erase Flash memory sectors.
- Write Flash memory data.

5.2.2 Function documentation

FLASH_If_ConfBFB2()

```
uint32_t FLASH_If_ConfBFB2 (void )
```

Set/Reset BFB2 bit to select boot from Flash memory bank.

Table 8. FLASH_If_ConfBFB2() Return values

0	BFB2 successfully written to Flash memory.
1	Error occurred while BFB2 bit in Flash memory.

FLASH_If_DMA_DeInit()

```
void FLASH_If_DMA_DeInit (void )
```

Unconfigure the DMA controller for SRAM to Flash transfer.

Parameters

None

Return values

None

FLASH_If_DMA_Init()

```
void FLASH_If_DMA_Init (void )
```

Configure the DMA controller for SRAM to Flash memory transfer.

Note: This function is used to :

- *Enable DMA clock*
- *Select the DMA functional parameters*
- *Select the DMA instance to be used for the transfer*
- *Select callbacks functions called after transfer complete and transfer error interrupt detection*
- *Initialize the DMA channel*
- *Configure NVIC for DMA transfer complete/error interrupts*
- *Start the DMA transfer using the interrupt mode*

Parameters

None

Return values

None

FLASH_If_DMA_IT_WriteBuffer()

`uint32_t FLASH_If_DMA_IT_WriteBuffer (const uint32_t Address, const uint32_t *const pData, const uint32_t Size )`

Writes a data buffer in the Flash memory through the DMA.

Table 9. FLASH_If_DMA_IT_WriteBuffer() parameters

Address	Start address for writing data buffer (must be DWORD aligned).
pData	Pointer on data buffer.
Size	Size of the data.

Table 10. FLASH_If_DMA_IT_WriteBuffer() return values

0	Data successfully written to Flash memory.
1	Error occurred while writing data in Flash memory.

FLASH_If_DMA_WriteBuffer()

`uint32_t FLASH_If_DMA_WriteBuffer (const uint32_t Address, const uint32_t *const pData, const uint32_t Size )`

Writes a data buffer in the Flash memory through the DMA.

Table 11. FLASH_If_DMA_WriteBuffer() parameters

Address	Start address for writing data buffer (must be DWORD aligned).
pData	Pointer on data buffer.
Size	Size of the data.

Table 12. FLASH_If_DMA_WriteBuffer() return values

0	Data successfully written to Flash memory.
1	Error occurred while writing data in Flash memory.

FLASH_If_FlashLock()

`void FLASH_If_FlashLock (void )`

Locks the Flash memory to disable the Flash memory control register access.

Parameters

None

Returns

None.

FLASH_If_FlashUnlock()

`void FLASH_If_FlashUnlock (void )`

Unlocks the Flash memory to enable the Flash memory control register access.

Parameters

None

Returns

None.

FLASH_If_GetBank()

`uint32_t FLASH_If_GetBank (uint32_t Addr )`

Gets the bank of a given address.

Table 13. FLASH_If_GetBank() parameters

Addr	Address of the Flash memory

Table 14. FLASH_If_GetBank() returns

The	bank of a given address

FLASH_If_GetPage()

`uint32_t FLASH_If_GetPage (uint32_t Addr )`

Gets the page of a given address.

Table 15. FLASH_If_GetPage() parameters

Addr	Address of the Flash memory

Table 16. FLASH_If_GetPage() returns

The	page of a given address

FLASH_If_MassErase()

`uint32_t FLASH_If_MassErase (const uint32_t Address )`

Erases the required Flash memory sectors computed with destination address.

Table 17. FLASH_If_MassErase() parameters

Address	Start address for erasing data.
LastAddress	End address of Flash memory area.

Table 18. FLASH_If_MassErase() return values

0	Erase sectors done with success.
1	Erase error.

FLASH_If_PageErase()

`uint32_t FLASH_If_PageErase (const uint32_t Address, const uint32_t LastAddress )`

Erases the required Flash memory pages computed with destination address.

Table 19. FLASH_If_PageErase() parameters

Address	Start address for erasing data.
LastAddress	End address of Flash memory area.

Table 20. **FLASH_If_PageErase() return values**

0	Erase sectors done with success.
1	Erase error.

FLASH_If_ReadOutProtectionStatus()

```
FlagStatus FLASH_If_ReadOutProtectionStatus ( void )
```

Gets Flash memory readout protection status.

Parameters

None

Table 21. **FLASH_If_ReadOutProtectionStatus() return values**

ReadOut	protection status.

FLASH_If_WriteBuffer()

```
uint32_t FLASH_If_WriteBuffer (const uint32_t Address, const uint64_t *const pData,
const uint32_t Size )
```

Writes a data buffer in the Flash memory.

Table 22. **FLASH_If_WriteBuffer() parameters**

Address	Start address for writing data buffer.
LastAddress	End address of Flash memory area.
pData	Pointer on data buffer.
Size	Size of the data.

Table 23. **FLASH_If_WriteBuffer() return values**

0	Data successfully written to Flash memory.
1	Error occurred while writing data in Flash memory.

FLASH_If_WriteDWord()

```
uint32_t FLASH_If_WriteDWord (const uint32_t Address, const uint64_t pData )
```

Writes a data in Flash memory (data are 32-bit aligned).

Table 24. **FLASH_If_WriteDWord() parameters**

Address	Start address for writing data buffer.
LastAddress	End address of Flash memory area.
Data	Word data value to write.

Table 25. **FLASH_If_WriteDWord() return values**

0	Data successfully written to Flash memory.
1	Error occurred while writing data in Flash memory.

HAL_DMA_MspDeInit()

```
void HAL_DMA_MspDeInit (void )
```

DeInitializes the low level hardware: GPIO, CLOCK, NVIC for DMA.

Parameters

None

Return values

None

HAL_DMA_MspInit()

```
void HAL_DMA_MspInit (void )
```

Initializes the low level hardware: GPIO, CLOCK, NVIC for DMA.

Parameters

None

Return values

None

5.3 Simple filesystem API

Simple implementation of a filesystem API.

Files

- `file ff.h`
  Simple file system API header.
- `file fs_api.c`
  Simple filesystem API implementation.

Data structures

- `struct FIL`
  File object structure.

Macros

- `#define FA_READ 0x01`
  File access control: Read only (no other value supported)

Enumerations

- `enum FRESULT { FR_OK = 0 , FR_DENIED = 7 , FR_INVALID_OBJECT = 9 }`
  Function return codes.

Functions

- `FRESULTf_open (FIL *fp, const char *start, uint8_t mode)`
  Open a stream of data from an address in memory.
- `FRESULTf_close (FIL *fp)`
  Close a stream of data from an address in memory.
- `FRESULTf_read (FIL *fp, void *buff, uint32_t btr, uint32_t *br)`
  Reads bytes from a stream of data.
- `FRESULTf_write (FIL *fp, const void *buff, uint32_t btw, uint32_t *bw)`
  Do nothing.

5.3.1 Detailed description

The purpose of this module is to provide a standard filesystem API to feed data to the libJPEG.
This module implements the functions to open, read and close a file.

It does not implements a real filesystem. A file is considered being a starting address in memory, from where the API reads a stream of data.

5.3.2 Data structure documentation

struct FIL

File object structure.

Table 26. struct FIL data fields

uint8_t	open	Boolean, true if the file is currently opened .
const char *	ptr	Pointer to the next data to be read.

5.3.3 Enumeration type documentation

FRESULT

`enum FRESULT`

Function return codes.

Table 27. FRESULT enumerator

FR_OK	(0) Succeeded
FR_DENIED	(7) Access denied due to prohibited access
FR_INVALID_OBJECT	(9) The file object is invalid

5.3.4 Function documentation

f_close()

`FRESULT f_close (FIL * fp )`

Close a stream of data from an address in memory.

This function mimic the close standard function from a filesystem. The usage of such function in the ST25DV demonstration board is to feed data to the jpeg decoder module, without having a real filesystem.

Table 28. f_close() parameters

fp	Pointer to FIL structure - simulate a filehandle.

Table 29. f_close() return values

FR_OK	Data stream has been successfully closed.
FR_INVALID_OBJECT	fp is not allocated or not open.

f_open()

`FRESULT f_open (FIL * fp,const char * start, uint8_t mode )`

Open a stream of data from an address in memory.

This function simulates the open standard function from a filesystem. The usage of such function in the ST25DV demonstration board is to feed data to the jpeg decoder module, without having a real filesystem.

Table 30. **f_open() parameters**

fp	Pointer to FIL structure - simulate a filehandle.
start	Pointer to an address in memory, that will be used as source of the data.
mode	File access control, only FA_READ is supported (read-only).

Table 31. **f_open() return values**

FR_OK	Data stream is opened.
FR_INVALID_OBJECT	fp is not allocated/initialized.
FR_DENIED	mode is different from FA_READ.

f_read()

```
FRESULT f_read (FIL * fp, void * buff, uint32_t btr, uint32_t * br )
```

Reads bytes from a stream of data.

This function simulates the read standard function from a filesystem. The usage of such function in the ST25DV demonstration board is to feed data to the jpeg decoder module, without having a real filesystem.

Table 32. **f_read() parameters**

fp	Pointer to FIL structure - mimisc a filehandle.
buff	Memory buffer to store read data.
btr	Number of bytes to read.
br	Number of bytes actually read.

Table 33. **f_read() return values**

FR_OK	Data stream has been successfully read.
FR_INVALID_OBJECT	fp is not allocated or not open.

f_write()

```
FRESULT f_write (FIL * fp, const void * buff, uint32_t btw, uint32_t * bw )
```

Do nothing.

This function mimics the write standard function from a filesystem. The usage of such function in the ST25DV demonstration board is to feed data to the jpeg decoder module, without having a real filesystem.

Table 34. **f_write() parameters**

fp	Pointer to FIL structure - mimic a filehandle.
buff	Memory buffer with source of data.
btw	Number of bytes to read.
bw	Number of bytes actually written.

Table 35. **f_write() returns values**

FR_OK	Always.

5.3.5 Flash command

This module implements high level functions to write firmware or data to Flash memory.

Functions

- `uint32_t COMMAND_EraseFlash (const uint32_t Address)`
  Command To erase specific Flash memory area.
- `uint32_t Command_WriteBufferToFlash (const uint32_t StartAddress, const uint32_t offset, const uint8_←, t *const pData, const uint32_t size)`
  Writes buffer to Flash memory.
- `void COMMAND_Jump (void)`
  Jump to user program.

5.3.6 Detailed description

This module implements high level functions to write firmware or data to Flash. The module covers following functions:

- Erase Flash command.
- Write buffer to Flash.
- Jump to firmware command.

5.3.7 Function documentation

COMMAND_EraseFlash()

`uint32_t COMMAND_EraseFlash (const uint32_t Address )`

Command To erase specific Flash memory area.

Table 36. COMMAND_EraseFlash() parameters

Address	Start address for erasing data.

Table 37. COMMAND_EraseFlash() return values

0	Erase sectors done with success.
1	Erase error.

COMMAND_Jump()

`void COMMAND_Jump (void )`

Jump to user program.

Table 38. COMMAND_Jump() parameters

None	No parameters.

COMMAND_Jump() returns

None.

Command_WriteBufferToFlash()

`uint32_t Command_WriteBufferToFlash (const uint32_t StartAddress, const uint32_t offset, const uint8_t *const pData, const uint32_t size )`

Writes buffer to Flash memory.

Table 39. **Command_WriteBufferToFlash() parameters**

StartAddress	Start address for writing data.
offset	Offset of data to write.
pData	Buffer pointer to write.
size	Size of data to write.

Table 40. **Command_WriteBufferToFlash() return values**

0	Write Success.
1	Write Error.

5.4 LibJPEG decode wrapper

Wrapper calling the libJPEG Cube middleware to decode JPEG pictures.

Files

- `file jpeg_decode.c`
  This file contains the JPEG decompressing methods.

Macros

- `#define IS_JPEG(ptr) ((ptr[0] == 0xFF) && (ptr[1] == 0xD8))`
  Macro checking if a pointer points to the beginning of a JPEG picture.

Functions

- `uint32_t jpeg_decode (const char *jpeg, uint8_t(*callback)(uint8_t *, uint32_t))`
  Decode a jpeg formatted picture calling the Cube libJPEG middleware.
- `void jpeg_getsize (const char *jpeg, uint32_t *Width, uint32_t *Height)`
  Get the geometry of a JPEG picture.
- `uint32_t jpeg_GetBufferSize (uint8_t *jpeg)`
  Gut not decompressed Jpeg buffer size.
- `void jpeg_decode_exit (j_common_ptr cinfo)`
  Callback for the libJPEG, executed when an unrecoverable error occured.

5.4.1 Detailed description

Wrapper calling the libJPEG Cube middleware to decode JPEG pictures. This module implements the functions to:

- decode a JPEG picture
- read picture geometry (heigh and width)
- display errors when decoding fails.

It uses the decoding part of the libJPEG Cube middleware, and the fs_api (to mimic filesystem functions).

5.4.2 Function documentation

jpeg_decode()

`uint32_t jpeg_decode (const char * jpeg,uint8_t(*)(uint8_t *, uint32_t) callback )`

Decode a jpeg formatted picture calling the Cube libJPEG middleware.

Table 41. **jpeg_decode() parameters**

jpeg	Pointer to the data array with jpeg format picture.
callback	Callback function to be called after a line of the picture has been decoded.

Returns
Number of errors

jpeg_decode_exit()

`void jpeg_decode_exit (j_common_ptr cinfo )`
Callback for the libJPEG, executed when an unrecoverable error occured.

Table 42. **jpeg_decode_exit() parameters**

cinfo	Pointer to the jpeg_common_struct with the current libJPEG state.

Returns
None

jpeg_GetBufferSize()

`uint32_t jpeg_GetBufferSize (uint8_t * jpeg)`
Gut not decompressed Jpeg buffer size.

Table 43. **jpeg_GetBufferSize() parameters**

jpeg	Pointer to the data array with jpeg format picture.

Returns
Size of buffer in bytes.

jpeg_getsize()

`void jpeg_getsize (const char * jpeg, uint32_t * Width, uint32_t * Height )`
Get the geometry of a JPEG picture.
Calls the Cube libJPEG middleware to read the jpeg header and extract the geometry info.

Table 44. **jpeg_getsize() parameters**

jpeg	Pointer to the data array with jpeg format picture.
Width	Pointer used to return the width of the JPEG picture.
Height	Pointer used to return the height of the JPEG picture.

Returns
None

5.5 FTM

This module provides functions to manage the protocol for the FTM demonstration.

Files

- `file mailbox.c`
  Mailbox functions to illustrate the FTM feature.

Functions

- void FTMManagement (void)
  Manage data exchange protocol with reader through mailbox.

5.5.1 Detailed description

This module provides functions to manage the protocol for the FTM demonstrations. The use cases demonstrated are:

- Data transfer
- Firmware upgrade
- Download pictures
- Upload pictures.

5.5.2 Function documentation

FTMManagement()

```
void FTMManagement (void )
```

Manage data exchange protocol with reader through mailbox.

Table 45. FTMManagement() parameters

None	No parameters.

Returns

None.

5.6 ST25DV-I2C mailbox functions

This module proposes common API for the ST25DV-I2C mailbox.

Files

- `file mailboxfunc.c`
  Common mailbox functions used for FTM protocol.

Data structures

- `struct MB_HEADER_T`
  Mailbox global header information structure definition.
- `struct MB_STOPWATCH_T`

Functions

- `int32_t InitMailBoxMode (void)`
  Initializes the mailbox mode, enables mailbox and disables mailbox watchdog.
- `int32_t DeInitMailBoxMode (void)`
  DeInitializes the mailbox mode, disables mailbox mode.
- `int32_t InitGPOforMailBoxMode (void)`
  Initializes the mailbox mode, enables mailbox and disables mailbox watchdog.
- `int32_t WriteMailBoxMsg (const uint8_t *const pData, const uint16_t NbBytes)`
  Writes message in mailbox.
- `int32_t ReadCompleteMailBoxMsg (uint8_t *const pData, uint16_t *const pLength)`
  Reads entire mailbox message from the tag.

- `int32_t ReadFragmentMailBoxMsg (uint8_t *const pData, const uint8_t Offset, const uint16_t NbBytes)`
  Reads part of mailbox message from the tag.
- `void MBDecodeHeader (const uint8_t *const pData, MB_HEADER_T *const mb_header)`
  Extracts global information from header in Fast transfer mode protocol.
- `void PrepareMBMsg (uint8_t *const pData, const MB_HEADER_T *const mb_header)`
  Prepares header message to send to mailbox.
- `bool SendMBData (MB_HEADER_T *const mb_header, const uint8_t nbretry)`
  Prepares and writes frame in mailbox.

Variables

- `uint8_t GPO_Activated`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.

5.6.1 Detailed description

This module proposes common api for the ST25DV-I2C mailbox. The module covers following functions:

- Mailbox init
- Read and write to mailbox
- Protocol header decode
- Protocol message preparation
- Protocol send message.

5.6.2 Data structure documentation

`struct MB_HEADER_T`

Mailbox global header information structure definition.

Table 46. struct MB_HEADER_T data fields

uint8_t	chaining	Defines if the frame is a simple frame (0) or chained frame (1)
uint16_t	chunknb	Informs for a specific frame the current chunk number of that frame
uint8_t	cmdresp	Defines current frame command, answer, acknowledge
uint8_t	error	Error code if an error is detected
uint8_t	fctcode	Function code used to define the action of the requester
uint8_t	framelength	Informs the data length of the current frame
uint16_t	framesize	Size of current frame to transfer
uint32_t	fulllength	Informs on the total length of data to transfer
uint8_t *	pData	Pointer to buffer data to transfer
uint16_t	totalchunk	Informs on the total number of chunk that will need to perform the transfer

struct MB_STOPWATCH_T

5.6.3 Function documentation

DeInitMailBoxMode()

`int32_t DeInitMailBoxMode (void )`

DeInitializes the mailbox mode, disables mailbox mode.

Table 47. DeInitMailBoxMode() parameters

None	No parameters.

DeInitMailBoxMode() returns
int32_t status.

InitGPOforMailBoxMode()

`int32_t InitGPOforMailBoxMode ( void )`
Initializes the mailbox mode, enables mailbox and disables mailbox watchdog.

Table 48. InitGPOforMailBoxMode() parameters

None	No parameters.

Returns
int32_t status.

InitMailBoxMode()

`int32_t InitMailBoxMode (void )`
Initializes the mailbox mode, enables mailbox and disables mailbox watchdog.

Table 49. InitMailBoxMode() parameters

None	No parameters.

Returns
int32_t status.

MBDecodeHeader()

`void MBDecodeHeader (const uint8_t *const pData,MB_HEADER_T *const mb_header )`
Extracts global information from header in FTM protocol.

Table 50. MBDecodeHeader() parameters

pData	Pointer to the mailbox frame.
mb_header	Pointer to structure for storing global header info.

Returns
None.

PrepareMBMsg()

`void PrepareMBMsg (uint8_t *const pData,const MB_HEADER_T *const mb_header )`
Prepares header message to send to mailbox.

Table 51. PrepareMBMsg() parameters

pData	Pointer the the mailbox frame.
mb_header	Pointer to the header information structure.

Returns

None.

ReadCompleteMailBoxMsg()

`int32_t ReadCompleteMailBoxMsg (uint8_t *const pData,uint16_t *const pLength )`

Reads entire mailbox message from the tag.

Table 52. ReadCompleteMailBoxMsg() parameters

pData	Pointer to the read data to store.
pLength	Number of bytes to read.

Returns

int32_t status.

ReadFragmentMailBoxMsg()

`int32_t ReadFragmentMailBoxMsg (uint8_t *const pData, const uint8_t Offset, const uint16_t NbBytes )`

Reads part of mailbox message from the tag.

Table 53. ReadFragmentMailBoxMsg() parameters

pData	Pointer to store data.
Offset	Offset in mailbox to start reading.
NbBytes	Number of bytes to read.

Returns

int32_t status.

SendMBData()

`bool SendMBData (MB_HEADER_T *const mb_header,const uint8_t nbretry )`

Prepares and writes frame in mailbox.

Table 54. SendMBData() parameters

mb_header	Pointer to structure containing frame header info.
nbretry	Number of attempts.

Table 55. SendMBData() return values

1	Message was written to mailbox.
0	Message was not written to mailbox.

WriteMailBoxMsg()

`int32_t WriteMailBoxMsg (const uint8_t *const pData,const uint16_t NbBytes )`

Writes message in mailbox.

Table 56. WriteMailBoxMsg() parameters

pData	Pointer to the data to write.

NbBytes	Number of bytes to write.

Returns

int32_t status.

5.7 ST25 discovery demonstration board

This is the main module for all the ST25 discovery demonstration boards.

Modules

- ST25DV-I2C common functions
  This module proposes a generic API used for the ST25DV-I2C.
- Flash memory API
  This module defines an API to access the internal Flash memory.
- Simple filesystem API
  Simple implementation of a filesystem API.
- Flash command
  This module implements high level functions to write firmware or data to Flash memory.
- LibJPEG decode wrapper
  Wrapper calling the libJPEG Cube middleware to decode JPEG pictures.
- Fast transfer mode demo
  This module provides functions to manage the protocol for the FTM demonstrations.
- ST25DV-I2C mailbox functions
  This module proposes common API for the ST25DV-I2C mailbox.
- ST25DV-I2C menu definition
  This module defines the structure and content of the ST25DV-I2C demonstration board menu.
- NDEF demonstration
  This module implements the NDEF demonstration functions.
- ST25DV-I2C features demonstration
  This module implements the functions to execute the demonstrations of the ST25DV-I2C specific features.
- ST25 discovery interrupt routines
  This module defines all the required interrupt routines for the ST25DV demonstration board.

Files

- `file main.c`
  Main program body.

Functions

- `int main (void)`
  Demo entry point.
- `void SplashScreen (void)`
  Display splash screen.
- `void MenuAbout (void)`
  Display "_about_" screen.

5.7.1 Detailed description

Demonstration boards are divided in three different submodules:

- FTM demonstration boards
- ST25DV-I2C features demonstration boards
- NDEF demonstration boards

5.8 ST25DV-I2C menu interface configuration

Interface file for the `menu_demo` middleware.

Functions

- `uint16_t Menu_GetDisplayWidth (void)`
  Interface function to retrieve the display width.
- `uint16_t Menu_GetDisplayHeight (void)`
  Interface function to retrieve the display height.
- `void Menu_SetStyle (ColorStyles_t style)`
  Interface function to set the display front and back colors.
- `uint32_t Menu_GetFontHeight (void)`
  Interface function to get the font height.
- `uint32_t Menu_GetFontWidth (void)`
  Interface function to get the font width.
- `void Menu_DisplayPicture (uint32_t PosX, uint32_t PosY, const char *pict)`
  Interface function to print a picture (bmp or jpeg) on the display.
- `void Menu_GetPictureDim (const char *pict, uint32_t *width, uint32_t *height)`
  Interface function to compute the dimensions of a picture.
- `void Menu_DisplayRectangle (uint32_t PosX, uint32_t PosY, uint32_t Height, uint32_t Width)`
  Interface function to print a rectangle on the display.
- `void Menu_DisplayString (uint32_t Line, const char *Str)`
  Interface function to print a string on the display.
- `void Menu_DisplayStringAt (uint32_t PosX, uint32_t PosY, const char *Str)`
  Interface function to print a text at a specific point on the display.
- `void Menu_DisplayChar (uint32_t Line, uint32_t Column, uint8_t Ascii)`
  Interface function to print a character on the display.
- `void Menu_DisplaySpecChar (uint32_t Line, uint32_t Column, const char *c)`
  Interface function to print a special character on the display.
- `void Menu_DisplayClear (void)`
  Interface function to clear the display.
- `void Menu_DisplayClearLine (uint16_t Line)`
  Interface function to clear a line on display.
- `uint8_t Menu_ReadPosition (Menu_Position_t *State)`
  Interface function to get the position of a touch on a touchscreen.
- `uint8_t Menu_ReadDirection (Menu_Direction_t *State)`
  Interface function to get the direction and push (select) of a joystick.
- `uint8_t Menu_ReadSelection (uint8_t *State)`
  Interface function to get the state of a simple button.
- `void Menu_Delay (uint32_t duration)`
  Interface function to add latence in the menu.

5.8.1 Detailed description

Interface file for the `menu_demo` middleware. This module implements functions for:

- Accessing the display
- Reading user inputs:
  - touch screen
  - joystick
  - simple button.

5.8.2 Function documentation

Menu_Delay()

`void Menu_Delay (uint32_t duration )`

Interface function to add latence in the menu.

Table 57. Menu_Delay() parameters

duration	Latence in ms.

Menu_DisplayChar()

`void Menu_DisplayChar (uint32_t Line,uint32_t Column,uint8_t Ascii )`

Interface function to print a character on the display.

Table 58. Menu_DisplayChar() parameters

Line	Indicates the line number where the string as to be printed.
Column	The position of the character on the line.
Ascii	Character to be written.

Menu_DisplayClearLine()

`void Menu_DisplayClearLine (uint16_t Line )`

Interface function to clear a line on display.

Table 59. Menu_DisplayClearLine() parameters

Line	Indicates the line number to clear.

Menu_DisplayPicture()

`void Menu_DisplayPicture (uint32_t PosX,uint32_t PosY,const char * pict )`

Interface function to print a picture (bmp or jpeg) on the display.

Table 60. Menu_DisplayPicture() parameters

PosX	Position of the top left corner of the picture on X-axis.
PosY	Position of the top left corner of the picture on Y-axis.
pict	Pointer to the picture adress in memory.

Menu_DisplayRectangle()

`void Menu_DisplayRectangle (uint32_t PosX,uint32_t PosY, uint32_t Height, uint32_t Width )`

Interface function to print a rectangle on the display.

Table 61. Menu_DisplayRectangle() parameters

PosX	Position of the top left corner of the rectangle on the X-axis.
PosY	Position of the top left corner of the rectangle on the Y-axis.
Height	Rectangle height.
Width	Rectangle width.

Menu_DisplaySpecChar()

`void Menu_DisplaySpecChar (uint32_t Line,uint32_t Column,const char * c )`

Interface function to print a special character on the display.

Table 62. Menu_DisplaySpecChar() parameters

Line	Indicates the line number where the string as to be printed.
Column	The position of the character on the line.
c	pointer to the character data.

Menu_DisplayString()

`void Menu_DisplayString (uint32_t Line,const char * Str )`

Interface function to print a string on the display.

Table 63. Menu_DisplayString() parameters

Line	Indicates the line number where the string as to be printed.
Str	String to be printed on the display.

Menu_DisplayStringAt()

`void Menu_DisplayStringAt (uint32_t PosX,uint32_t PosY,const char * Str )`

Interface function to print a text at a specific point on the display.

Table 64. Menu_DisplayStringAt() parameters

PosX	Position of the text on X-axis.
PosY	Position of the Text on Y-axis.
Str	String to be printed on the display.

Menu_GetDisplayHeight()

`uint16_t Menu_GetDisplayHeight ( void )`

Interface function to retrieve the display height.

Table 65. Menu_GetDisplayHeight() returns values

Display	height in pixels.

Menu_GetDisplayWidth()

`uint16_t Menu_GetDisplayWidth ( void )`

Interface function to retrieve the display width.

Table 66. Menu_GetDisplayWidth() returns values

Display	width in pixels.

Menu_GetFontHeight()

`uint32_t Menu_GetFontHeight (void )`

Interface function to get the FONT height.

Table 67. **Menu_GetFontHeight() returns values**

Font	height in pixels.

Menu_GetFontWidth()

```
uint32_t Menu_GetFontWidth (void )
```

Interface function to get the font width.

Table 68. **Menu_GetFontWidth() returns values**

Font	width in pixels.

Menu_GetPictureDim()

```
void Menu_GetPictureDim (const char * pict, uint32_t * width, uint32_t * height )
```

Interface function to compute the dimensions of a picture.

Table 69. **Menu_GetPictureDim() parameters**

pict	Pointer to the picture adress in memory.
width	Picture position on X-axis.
height	Picture position on Y-axis.

Menu_ReadDirection()

```
uint8_t Menu_ReadDirection (Menu_Direction_t * State )
```

Interface function to get the direction and push (select) of a joystick.

Table 70. **Menu_ReadDirection() parameters**

State	Pointer on a Menu_Direction_t structure, used to return the up,down,left,right direction and selection of the joystick.

Table 71. **Menu_ReadDirection() return values**

Return	null if the joystick is in neutral position, not-null otherwise.

Menu_ReadPosition()

```
uint8_t Menu_ReadPosition (Menu_Position_t * State )
```

Interface function to get the position of a touch on a touchscreen.

Table 72. **Menu_ReadPosition() parameters**

State	Pointer on a Menu_Position_t structure, used to return the X,Y coordinates of last detected touch.

Table 73. **Menu_ReadPosition() return values**

Return	not-null if a touch has been detected, null otherwise.

Menu_ReadSelection()

```
uint8_t Menu_ReadSelection (uint8_t * State )
```

Interface function to get the state of a simple button.

Table 74. Menu_ReadSelection() parameters

State	Pointer on a boolean, used to return the state of the button.

Table 75. Menu_ReadSelection() return values

Return	not/null if the button is pushed, not/null otherwise.

Menu_SetStyle()

`void Menu_SetStyle (ColorStyles_t style )`

Interface function to set the display front & back colors.

Table 76. Menu_SetStyle() parameters

style	One of the value defined in enum ColorStyles_t.

5.9 ST25DV-I2C menu definition

This module defines the structure and content of the ST25DV-I2C demonstration board menu.

Files

- `file Menu_definition.c`
  This file defines the content of the menu for the ST25DV-I2C demonstration board.

Functions

- `void Menu_Start (void)`
  Starts the main loop for the demonstration board menu.

5.9.1 Detailed description

This module defines the structure and content of the ST25DV-I2C demonstration board menu.

Menu structure is statically defined in the module, and complies with the expected structure of the `menu_demo` middleware. Call `Menu_Start()` to start the menu main loop.

5.10 NDEF demonstration board

This module implements the NDEF demonstration board functions.

Files

- `file ndef_demo.c`
  This file provides functions to execute NDEF demonstration board.

Functions

- `void NDEF_DEMO_Init_Tag (void)`
  This function initializes the NFC Tag to perform the NDEF demonstration board.
- `void NDEF_DEMO_Write_URI_URL (void)`
  This function writes an URL to the tag.
- `void NDEF_DEMO_Write_URI_Tel (void)`
  This function writes a phone number to the tag.

- `void NDEF_DEMO_Read_URI (void)`
  This function reads and URI from the tag and displays its content.
- `void NDEF_DEMO_Read_SMS (void)`
  This function reads a SMS from the tag and displays it.
- `void NDEF_DEMO_Write_SMS (void)`
  This function writes a SMS to the tag.
- `void NDEF_DEMO_Read_Email (void)`
  This function reads an email from the tag and displays its content.
- `void NDEF_DEMO_Write_Email (void)`
  This function writes an email to the tag.
- `void NDEF_DEMO_Read_Vcard (void)`
  This function reads a vCard record from the tag and display its content.
- `void NDEF_DEMO_Write_Vcard (void)`
  This function writes a vCard record to the tag.
- `void NDEF_DEMO_Write_Picture_Vcard (void)`
  This function writes a vCards with an embedded picture to the tag.
- `void NDEF_DEMO_Read_Geo (void)`
  This function reads a geolocation record from the tag and displays it content.
- `void NDEF_DEMO_Write_Geo (void)`
  This function writes a Geolocation record to the Tag.
- `void NDEF_DEMO_Read_MyAPP (void)`
  This function reads a MyApp record from the tag and starts the associated demo.
- `void NDEF_DEMO_Write_AAR (void)`
  This function writes an AAR record (selecting the ST NFC application) to the tag.
- `void NDEF_DEMO_MultiRecord_With_AAR (void)`
  This function adds an AAR record (selecting the ST NFC application) to an existing NDEF file on the tag.
- `void NDEF_DEMO_BLE_ChangeDeviceAddr (void)`
  This function change the BLE device address (to be used when the module is paired with a nearby smartphone).
- `void NDEF_DEMO_Write_BLE_OOB (void)`
  This function writes a Bluetooth Low Energy OOB record to the tag and starts the BLE module, waiting for a connec tion to occur.
- `void NDEF_DEMO_Read_Bluetooth_OOB (void)`
  This function reads a BLE OOB record from the tag and displays it content.
- `void NDEF_DEMO_Write_Wifi_OOB (void)`
  This function writes a WiFi OOB record to the Ttag and starts the WiFi module, waiting for a connection to occur.
- `void NDEF_DEMO_Read_Wifi_OOB (void)`
  This function reads a WiFi OOB record from the tag and displays its content.
- `void NDEF_DEMO_Write_empty_NDEF (void)`
  This function writes an empty NDEF message.
- `void NDEF_DEMO_Erase_CCFile (void)`
  This function writes 0xFF to the 4 first bytes in EEPROM.
- `void NDEF_DEMO_Clear_Eeprom (void)`
  This function writes 0xFF to the entire EEPROM.
- `void Hid_Profile_Application (void)`
  Starts the HID application, using touchscreen detection to send mouse input reports and user button as mouse button.

- `void NDEF_DEMO_Write_NoPicture_Vcard (void)`
  This function writes a small vCard record to the tag.

5.10.1 Detailed description

This module implements the NDEF demonstration board functions. It covers the following use cases:

- URI NDEF records: URL and phone
- SMS NDEF record
- Email NDEF record
- vCard NDEF record (with and without an embedded picture)
- Geolocation NDEF record
- MyApp custom NDEF record
- Multi NDEF record with AAR
- Bluetooth Low Energy OOB NDEF record
- WiFi OOB NDEF record

5.10.2 Function documentation

NDEF_DEMO_Init_Tag()

```
void NDEF_DEMO_Init_Tag (void )
```

This function initializes the NFC tag to perform the NDEF demonstration boards. The tag is configured to its default and a CC file is written.

5.11 ST25DV-I2C features demonstration

This module implements the functions to execute the demonstrations of the ST25DV-I2C specific features.

Files

- `file st25dv_features_demo.c`
  This file provides functions to execute ST25DV demonstrations.

Functions

- `void ST25DV_DEMO_EnergyHarvesting (void)`
  This functions runs the energy harvesting demonstration with digital potentiometer set to 240 Ohms.
- `void ST25DV_DEMO_GPO (void)`
  This function runs the GPO interrupts demonstrations.
- `void ST25DV_DEMO_RF_Off (void)`
  This function sets the ST25DV-I2C RF off.
- `void ST25DV_DEMO_RF_Sleep (void)`
  This function sets the ST25DV-I2C RF to sleep state.
- `void ST25DV_DEMO_Low_Power_Down (void)`
  This function sets the ST25DV-I2C in low power mode.
- `void ST25DV_DEMO_Memory_Mapping_Password (void)`
  This function configures the ST25DV-I2C memory mapping and sets a password protection.
- `void ST25DVxxKC_DEMO_CompareWriteEEPROM (void)`
  This function emulate an ST25DVxxK EEPROM write and compare it to the ST25DVxxKC write.

5.11.1 Detailed description

This module permits to execute the ST25DV-I2C demonstration boards specific features and covers the following use cases:

- Energy harvesting

- GPO interrupts from RF
- ST25DV-I2C states:
  - RF off
  - RF sleep
  - Low power down
- Memory map configuration and password protection.

5.11.2 Function documentation

ST25DV_DEMO_GPO()

```
void ST25DV_DEMO_GPO (void )
```

This function runs the GPO interrupts demonstrations.

This function displays the type of received GPO interrupts. It also displays the number of interrupt received.

ST25DV_DEMO_Memory_Mapping_Password()

```
void ST25DV_DEMO_Memory_Mapping_Password (void )
```

This function configures the ST25DV-I2C memory mapping and sets a password protection. Sets two memory areas:

1. ReadOnly with a vCard NDEF record.
2. NoRead, NoWrite with another vCard NDEF record.

5.12 HAL_MSP

HAL MSP module.

Modules

- `HAL_MSP_Private_Functions`

5.13 HAL_MSP_Private_Functions

Functions

- `void HAL_MspInit (void)`
  Initializes the global MSP.
- `void HAL_MspDeInit (void)`
  DeInitializes the global MSP.
- `void HAL_SPI_MspInit (SPI_HandleTypeDef *hspi)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for SPI.
- `void HAL_SPI_MspDeInit (SPI_HandleTypeDef *hspi)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_UART_MspInit (UART_HandleTypeDef *huart)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_UART_MspDeInit (UART_HandleTypeDef *huart)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_DMA_MspInit (void)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for DMA.
- `void HAL_DMA_MspDeInit (void)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for DMA.
- `void HAL_CRC_MspInit (CRC_HandleTypeDef *hcrc)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for CRC.

- `void HAL_CRC_MspDeInit (CRC_HandleTypeDef *hcrc)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for CRC.
- `void SystemClock_Config (void)`
  System clock configuration: the system Clock is configured as follow : `System Clock source = PLL (HSE) SYSCLK(Hz) = 80000000 HCLK(Hz) = 80000000 AHB Prescaler = 1 APB1 Prescaler = 1 APB2 Prescaler = 1 HSE Frequency(Hz) = 8000000 PLL_M = 1 PLL_N = 20 PLL_P = 7 PLL_Q = 4 PLL_R = 2 Flash Latency(WS) = 4.`
- `void MX_GPIO_Init (void)`

5.13.1 Function documentation

HAL_CRC_MspDeInit()

`void HAL_CRC_MspDeInit (CRC_HandleTypeDef * hcrc )`

DeInitializes the low level hardware: GPIO, CLOCK, NVIC for CRC.

Table 77. HAL_CRC_MspDeInit() parameters

hcrc	pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC module.

HAL_CRC_MspDeInit() return values

None

HAL_CRC_MspInit()

`void HAL_CRC_MspInit (CRC_HandleTypeDef * hcrc )`

Initializes the low level hardware: GPIO, CLOCK, NVIC for CRC.

Table 78. HAL_CRC_MspInit() parameters

hcrc	pointer to a CRC_HandleTypeDef structure that contains the configuration information for CRC module.

Return values

None

HAL_DMA_MspDeInit()

`void HAL_DMA_MspDeInit (void )`

DeInitializes the low level hardware: GPIO, CLOCK, NVIC for DMA.

HAL_DMA_MspInit()

`void HAL_DMA_MspInit (void )`

Initializes the low level hardware: GPIO, CLOCK, NVIC for DMA.

HAL_MspDeInit()

`void HAL_MspDeInit (void )`

DeInitializes the Global MSP.

HAL_MspInit()

`void HAL_MspInit (void )`

Initializes the Global MSP.

HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (SPI_HandleTypeDef * hspi )
```

DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.

Parameters

None

Return values

None

```
SPI3 GPIO Configuration PA15 ----> SPI3_NSS PC10 ----> SPI3_SCK PC11 ----> SPI3_MISO PC12 ----> SPI3_MOSI
```

HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (SPI_HandleTypeDef * hspi )
```

Initializes the low level hardware: GPIO, CLOCK, NVIC for SPI.

Parameters

None

Return values

None

SPI3 GPIO Configuration

```
PA15 ----> SPI3_NSS PC10 ----> SPI3_SCK PC11 ----> SPI3_MISO PC12 ----> SPI3_MOSI
```

HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (UART_HandleTypeDef * huart )
```

DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.

Parameters

None

Return values

None

USART3 GPIO Configuration

PB14 ------> USART3_RTS PD8 ------> USART3_TX PD9 ------> USART3_RX PD11 ------> USART3_CTS

HAL_UART_MspInit()

```
void HAL_UART_MspInit (UART_HandleTypeDef * huart )
```

Initializes the low level hardware: GPIO, CLOCK, NVIC for UART.

Parameters

None

Return values

None

USART3 GPIO Configuration

```
PB14 ----> USART3_RTS PD8 ----> USART3_TX PD9 ----> USART3_RX PD11 ----> USART3_CTS
```

MX_GPIO_Init()

```
void MX_GPIO_Init (void )
Configure pins as Analog Input Output EVENT_OUT EXTI PA10 ----> USB_OTG_FS_ID PA11 ----> USB_←, OTG_FS_DM PA12 ----> USB_OTG_FS_DP
```

SystemClock_Config()

```
void SystemClock_Config (void )
```

System clock configuration.

The system clock is configured as follow : System Clock source = PLL (HSE) SYSCLK(Hz) = 80000000 HCLK(Hz) = 80000000 AHB Prescaler = 1 APB1 Prescaler = 1 APB2 Prescaler = 1 HSE Frequency(Hz) = 8000000 PLL_M = 1 PLL_N = 20 PLL_P = 7 PLL_Q = 4 PLL_R = 2 Flash Latency(WS) = 4.

Parameters

None

Return values

None

5.14 ST25 discovery interrupt routines

This module defines all the required interrupt routines for the ST25DV demonstration board.

Modules

- Cortex®-M4 exceptions handlers
  This module defines handlers for the Cortex-M4 hardware exceptions.

Functions

- `void SysTick_Handler (void)`
  This function handles SysTick handler.
- `void TIMx_IRQHandler (void)`
  This function handles TIM interrupt request.
- `void TIMp_IRQHandler (void)`
  This function handles TIM interrupt request.
- `void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)`
  Period elapsed callback in non blocking mode This timer is used for calling back user registered functions with information.
- `void BNRG_SPI_EXTI_IRQHandler (void)`
  This function handles external line interrupt request for BlueNRG bluetooth module.
- `void USART3_IRQHandler (void)`
  This function handles USART3 interrupts for the WiFi module.
- `void NFCMEM_GPO_EXTIHandler (void)`
  This function handles external line 0 interrupt request.
- `void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)`
  This function handles callback from external line interrupt request.

Variables

- `volatile uint32_t ms_counter = 0`
  Current millisecond count.
- `volatile uint8_t GPO_Activated = 0`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.
- `TIM_HandleTypeDef TimHandle`
  Timer handle from st25_spwf_wifi.c.
- `TIM_HandleTypeDef PushTimHandle`
  Timer handle from st25_spwf_wifi.c.
- `UART_HandleTypeDef UartWiFiHandle`
  UART handle from WiFi_module.c.

5.14.1 Detailed description

This module defines all the required interrupt routines for the ST25DV demonstration board. Interrupts routines used in the ST25DV:

- System tick (ms timer)
- SPI3 for the BlueNRG bluetooth module
- USART3 for the Wifi module
- GPO interrrupt from ST25DV

Note: The hardware exceptions handler are defined in the same file but are documented in a sub-module to enforce readability.

5.14.2 Function documentation

HAL_GPIO_EXTI_Callback()

`void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin )`

This function handles callback from external line interrupt request.

Parameters

GPIO_Pin

HAL_TIM_PeriodElapsedCallback()

`void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)`

Period elapsed callback in non-blocking mode This timer is used for calling back user registered functions with information.

Table 79. HAL_TIM_PeriodElapsedCallback() parameters

htim	: TIM handle

SysTick_Handler()

`void SysTick_Handler (void )`

This function handles SysTick handler.

Parameters

None

Return values

None

TIMp_IRQHandler()

`void TIMp_IRQHandler (void )`

This function handles TIM interrupt request.

Parameters

None

Return values

None

TIMx_IRQHandler()

`void TIMx_IRQHandler (void )`

This function handles TIM interrupt request.

Parameters

None

Return values

None

5.15 Cortex-M4 exceptions handlers

This module defines handlers for the Cortex-M4 hardware exceptions.

Functions

- `void NMI_Handler (void)`
  This function handles NMI exception.
- `void HardFault_Handler (void)`
  This function handles hard fault exception.
- `void MemManage_Handler (void)`
  This function handles memory manage exception.
- `void BusFault_Handler (void)`
  This function handles bus fault exception.
- `void UsageFault_Handler (void)`
  This function handles usage fault exception.
- `void SVC_Handler (void)`
  This function handles SVCall exception.
- `void DebugMon_Handler (void)`
  This function handles debug monitor exception.
- `void PendSV_Handler (void)`
  This function handles PendSVC exception.

5.15.1 Detailed description

This module defines handlers for the Cortex-M4 hardware exceptions.

Hardware exceptions handlers are called when a specific situation occurs on the HW, such as unknown instruction, bus error.

5.15.2 Function documentation

BusFault_Handler()

```
void BusFault_Handler (void )
```

This function handles bus fault exception.

Parameters

None

Return values

None

DebugMon_Handler()

```
void DebugMon_Handler (void )
```

This function handles debug monitor exception.

Parameters

None

Return values

None

HardFault_Handler()

```
void HardFault_Handler (void )
```

This function handles hard fault exception.

Parameters

None

Return values
None

MemManage_Handler()

`void MemManage_Handler (void )`
This function handles memory manage exception.
Parameters
None
Return values
None

NMI_Handler()

`void NMI_Handler (void )`
This function handles NMI exception.
Parameters
None
Return values
None

PendSV_Handler()

`void PendSV_Handler (void )`
This function handles PendSVC exception.
Parameters
None
Return values
None

SVC_Handler()

`void SVC_Handler (void )`
This function handles SVCall exception.
Parameters
None
Return values
None

UsageFault_Handler()

`void UsageFault_Handler (void )`
This function handles usage fault exception.
Parameters
None
Return values
None

5.16 ST25 discovery NFCTAG board support package

This module provides high level functions to access the ST25DV-I2C NFC dynamic tag.

Files

- `file st25_discovery_nfctag.c`
  This file provides a set of functions needed to manage a NFC dual interface EEPROM memory.

Data structures

- `struct NFCTAG_DrvTypeDef`
  NFCTAG standard driver API structure definition.

Macros

- `#define NFCTAG_4K_SIZE ((uint32_t) 0x200)`
  Number of bytes for the NFCTAG 4 Kbits.
- `#define NFCTAG_16K_SIZE ((uint32_t) 0x800)`
  Number of bytes for the NFCTAG 16 Kbits.
- `#define NFCTAG_64K_SIZE ((uint32_t) 0x2000)`
  Number of bytes for the NFCTAG 64 Kbits.
- `#define NFCTAG_OK (0)`
  NFCTAG status enumerator definition.
- `#define BSP_NFCTAG_INSTANCE 0`
  NFCTAG instance.

Functions

- `int32_t BSP_NFCTAG_Init (const uint32_t Instance)`
  Initializes peripherals used by the I²C NFCTAG driver.
- `void BSP_NFCTAG_DeInit (const uint32_t Instance)`
  Deinitializes peripherals used by the I²C NFCTAG driver.
- `int32_t BSP_NFCTAG_isInitialized (const uint32_t Instance)`
  Check if the NFCTAG is initialized.
- `int32_t BSP_NFCTAG_ReadID (const uint32_t Instance, uint8_t *const wai_id)`
  Read the ID of the NFCTAG.
- `int32_t BSP_NFCTAG_IsDeviceReady (const uint32_t Instance, const uint32_t Trials)`
  Check if the NFCTAG is available.
- `int32_t BSP_NFCTAG_ConfigIT (const uint32_t Instance, const uint16_t ITConfig)`
  Configure NFCTAG interrupt.
- `int32_t BSP_NFCTAG_GetITStatus (const uint32_t Instance, uint16_t *const ITConfig)`
  Get NFCTAG interrupt configutration.
- `int32_t BSP_NFCTAG_ReadData (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Reads data in the NFCTAG at specific address.
- `int32_t BSP_NFCTAG_WriteData (const uint32_t Instance, const uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t Size)`
  Writes data in the NFCTAG at specific address.
- `int32_t BSP_NFCTAG_ReadRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Reads NFCTAG register.
- `int32_t BSP_NFCTAG_WriteRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Writes NFCTAG register.
- `uint32_t BSP_NFCTAG_GetByteSize (const uint32_t Instance)`
  Return the size of the NFCTAG.
- `int32_t BSP_NFCTAG_ReadICRev (const uint32_t Instance, uint8_t *const pICRev)`
  Reads the ST25DV-I2C IC revision.

- `int32_t BSP_NFCTAG_ReadITPulse (const uint32_t Instance, void *const pITtime)`
  Reads the ST25DV-I2C ITtime duration for the GPO pulses.
- `int32_t BSP_NFCTAG_WriteITPulse (const uint32_t Instance, const uint8_t ITtime)`
  Configures the ST25DV-I2C ITtime duration for the GPO pulse.
- `int32_t BSP_NFCTAG_ReadUID (const uint32_t Instance, void *const pUid)`
  Reads the ST25DV-I2C UID.
- `int32_t BSP_NFCTAG_ReadDSFID (const uint32_t Instance, uint8_t *const pDsfid)`
  Reads the ST25DV-I2C DSFID.
- `int32_t BSP_NFCTAG_ReadDsfidRFProtection (const uint32_t Instance, void *const pLockDsfid)`
  Reads the ST25DV-I2C DSFID RF lock state.
- `int32_t BSP_NFCTAG_ReadAFI (const uint32_t Instance, uint8_t *const pAfi)`
  Reads the ST25DV-I2C AFI.
- `int32_t BSP_NFCTAG_ReadAfiRFProtection (const uint32_t Instance, void *const pLockAfi)`
  Reads the AFI RF lock state.
- `int32_t BSP_NFCTAG_ReadI2CProtectZone (const uint32_t Instance, void *const pProtZone)`
  Reads the I2C Protected Area state.
- `int32_t BSP_NFCTAG_WriteI2CProtectZonex (const uint32_t Instance, const uint8_t Zone, const uint8_←, t ReadWriteProtection)`
  Sets the I²C write-protected state to an EEPROM Area.
- `int32_t BSP_NFCTAG_ReadLockCCFile (const uint32_t Instance, void *const pLockCCFile)`
  Reads the CCfile protection state.
- `int32_t BSP_NFCTAG_WriteLockCCFile (const uint32_t Instance, const uint8_t NbBlockCCFile, const uint8←,_t LockCCFile)`
  Locks the CCfile to prevent any RF write access.
- `int32_t BSP_NFCTAG_ReadLockCFG (const uint32_t Instance, void *const pLockCfg)`
  Reads the CFG registers protection.
- `int32_t BSP_NFCTAG_WriteLockCFG (const uint32_t Instance, const uint8_t LockCfg)`
  Lock/Unlock the CFG registers, to prevent any RF write access.
- `int32_t BSP_NFCTAG_PresentI2CPassword (const uint32_t Instance, const void *const PassWord)`
  Presents I²C password, to authorize the I²C writes to protected areas.
- `int32_t BSP_NFCTAG_WriteI2CPassword (const uint32_t Instance, const void *const PassWord)`
  Writes a new I²C password.
- `int32_t BSP_NFCTAG_ReadRFZxSS (const uint32_t Instance, const uint8_t Zone, void *const pRfprotZone)`
  Reads the RF zone security status (defining the allowed RF accesses).
- `int32_t BSP_NFCTAG_WriteRFZxSS (const uint32_t Instance, const uint8_t Zone, const void *const Rf←, ProtZone)`
  Writes the RF zone security status (defining the allowed RF accesses)
- `int32_t BSP_NFCTAG_ReadEndZonex (const uint32_t Instance, const uint8_t EndZone, uint8_t *pEndZ)`
  Reads the value of the end area address.
- `int32_t BSP_NFCTAG_WriteEndZonex (const uint32_t Instance, const uint8_t EndZone, const uint8_t EndZ)`
  Sets the end address of an area.

- `int32_t BSP_NFCTAG_InitEndZone (const uint32_t Instance)`
  Initializes the end address of the ST25DV-I2C areas with their default values (end of memory).
- `int32_t BSP_NFCTAG_CreateUserZone (const uint32_t Instance, uint16_t Zone1Length, uint16_t Zone2←, Length, uint16_t Zone3Length, uint16_t Zone4Length)`
  Creates user areas with defined lengths.
- `int32_t BSP_NFCTAG_ReadMemSize (const uint32_t Instance, void *const pSizeInfo)`
  Reads the ST25DV-I2C memory size.
- `int32_t BSP_NFCTAG_ReadEHMode (const uint32_t Instance, void *const pEH_mode)`
  Reads the energy harvesting mode.
- `int32_t BSP_NFCTAG_WriteEHMode (const uint32_t Instance, const uint8_t EH_mode)`
  Sets the energy harvesting mode.
- `int32_t BSP_NFCTAG_ReadRFMngt (const uint32_t Instance, void *const pRF_Mngt)`
  Reads the RF management configuration.
- `int32_t BSP_NFCTAG_WriteRFMngt (const uint32_t Instance, const uint8_t Rfmngt)`
  Sets the RF management configuration.
- `int32_t BSP_NFCTAG_GetRFDisable (const uint32_t Instance, void *const pRFDisable)`
  Reads the RF disable register information.
- `int32_t BSP_NFCTAG_SetRFDisable (const uint32_t Instance)`
  Sets the RF disable configuration.
- `int32_t BSP_NFCTAG_ResetRFDisable (const uint32_t Instance)`
  Resets the RF disable configuration.
- `int32_t BSP_NFCTAG_GetRFSleep (const uint32_t Instance, void *const pRFSleep)`
  Reads the RF sleep register information.
- `int32_t BSP_NFCTAG_SetRFSleep (const uint32_t Instance)`
  Sets the RF sleep configuration.
- `int32_t BSP_NFCTAG_ResetRFSleep (const uint32_t Instance)`
  Resets the RF sleep configuration.
- `int32_t BSP_NFCTAG_ReadMBMode (const uint32_t Instance, void *const pMB_mode)`
  Reads the mailbox mode.
- `int32_t BSP_NFCTAG_WriteMBMode (const uint32_t Instance, const uint8_t MB_mode)`
  Sets the mailbox mode.
- `int32_t BSP_NFCTAG_ReadMBWDG (const uint32_t Instance, uint8_t *const pWdgDelay)`
  Reads the mailbox watchdog duration coefficient.
- `int32_t BSP_NFCTAG_WriteMBWDG (const uint32_t Instance, const uint8_t WdgDelay)`
  Writes the mailbox watchdog coefficient delay.
- `int32_t BSP_NFCTAG_ReadMailboxData (const uint32_t Instance, uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t NbByte)`
  Reads N bytes of data from the mailbox, starting at the specified byte offset.
- `int32_t BSP_NFCTAG_WriteMailboxData (const uint32_t Instance, const uint8_t *const pData, const uint16_t NbByte)`
  Writes N bytes of data in the mailbox, starting from first mailbox address.
- `int32_t BSP_NFCTAG_ReadMailboxRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from the mailbox registers, starting at the specified I²C address.
- `int32_t BSP_NFCTAG_WriteMailboxRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified mailbox register.

- `int32_t BSP_NFCTAG_ReadI2CSecuritySession_Dyn (const uint32_t Instance, void *const pSession)`

  Reads the status of the security session open register.
- `int32_t BSP_NFCTAG_ReadITSTStatus_Dyn (const uint32_t Instance, uint8_t *const pITStatus)`

  Reads the IT status register from the ST25DV-I2C.
- `int32_t BSP_NFCTAG_ReadGPO_Dyn (const uint32_t Instance, uint8_t *GPOConfig)`

  Read value of dynamic GPO register configuration.
- `int32_t BSP_NFCTAG_GetGPO_en_Dyn (const uint32_t Instance, void *const pGPO_en)`

  Get dynamique GPO enable status.
- `int32_t BSP_NFCTAG_SetGPO_en_Dyn (const uint32_t Instance)`

  Set dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ResetGPO_en_Dyn (const uint32_t Instance)`

  Reset dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ReadEHCtrl_Dyn (const uint32_t Instance, void *const pEH_CTRL)`

  Read value of dynamic EH Ctrl register configuration.
- `int32_t BSP_NFCTAG_GetEHENMode_Dyn (const uint32_t Instance, void *const pEH_Val)`

  Reads the energy harvesting dynamic status.
- `int32_t BSP_NFCTAG_SetEHENMode_Dyn (const uint32_t Instance)`

  Dynamically sets the energy harvesting mode.
- `int32_t BSP_NFCTAG_ResetEHENMode_Dyn (const uint32_t Instance)`

  Dynamically unsets the energy harvesting mode.
- `int32_t BSP_NFCTAG_GetEHON_Dyn (const uint32_t Instance, void *const pEHON)`

  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t BSP_NFCTAG_GetRFField_Dyn (const uint32_t Instance, void *const pRF_Field)`

  Checks if RF field is present in front of the ST25DV-I2C.
- `int32_t BSP_NFCTAG_GetVCC_Dyn (const uint32_t Instance, void *const pVCC)`

  Check if V_{CC} is supplying the ST25DV-I2C.
- `int32_t BSP_NFCTAG_ReadRFMngt_Dyn (const uint32_t Instance, void *const pRF_Mngt)`

  Read value of dynamic RF management configuration.
- `int32_t BSP_NFCTAG_WriteRFMngt_Dyn (const uint32_t Instance, const uint8_t RF_Mngt)`

  Writes a value to the RF management dynamic register.
- `int32_t BSP_NFCTAG_GetRFDisable_Dyn (const uint32_t Instance, void *const pRFDisable)`

  Reads the RFDisable dynamic register information.
- `int32_t BSP_NFCTAG_SetRFDisable_Dyn (const uint32_t Instance)`

  Sets the RF disable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetRFDisable_Dyn (const uint32_t Instance)`

  Unsets the RF disable dynamic configuration.
- `int32_t BSP_NFCTAG_GetRFSleep_Dyn (const uint32_t Instance, void *const pRFSleep)`

  Reads the RF sleep dynamic register information.
- `int32_t BSP_NFCTAG_SetRFSleep_Dyn (const uint32_t Instance)`

  Sets the RF sleep dynamic configuration.

- `int32_t BSP_NFCTAG_ResetRFSleep_Dyn (const uint32_t Instance)`
  Unsets the RF sleep dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBCtrl_Dyn (const uint32_t Instance, void *const pCtrlStatus)`
  Reads the mailbox CTRL dynamic register.
- `int32_t BSP_NFCTAG_GetMBEN_Dyn (const uint32_t Instance, void *const pMBEN)`
  Reads the mailbox enable dynamic configuration.
- `int32_t BSP_NFCTAG_SetMBEN_Dyn (const uint32_t Instance)`
  Sets the mailbox enable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetMBEN_Dyn (const uint32_t Instance)`
  Unsets the mailbox enable dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBLength_Dyn (const uint32_t Instance, uint8_t *const pMBLength)`
  Reads the mailbox message length dynamic register.

5.16.1 Detailed description

This module provides high level functions to access the ST25DV-I2C NFC dynamic tag. These functions are designed to be called by the application or by a middleware.

5.16.2 Function documentation

BSP_NFCTAG_ConfigIT()

`int32_t BSP_NFCTAG_ConfigIT (const uint32_t Instance,const uint16_t ITConfig )`

Configure nfctag interrupt.

Table 80. BSP_NFCTAG_ConfigIT() parameters

ITConfig	: store interrupt to configure • 0x01 => RF BUSY • 0x02 => WIP

Table 81. BSP_NFCTAG_ConfigIT() return values

NFCTAG	enum status

BSP_NFCTAG_CreateUserZone()

`int32_t BSP_NFCTAG_CreateUserZone (const uint32_t Instance, uint16_t` *`Zone1Length`*`, uint16_t` *`Zone2Length`*`, uint16_t` *`Zone3Length`*`, uint16_t` *`Zone4Length`* `)`

Creates user areas with defined lengths.

Needs the I²C password presentation to be effective.

Table 82. BSP_NFCTAG_CreateUserZone() parameters

Zone1Length	Length of area1 in bytes (32 to 8192, 0x20 to 0x2000)
Zone2Length	Length of area2 in bytes (0 to 8128, 0x00 to 0x1FC0)
Zone3Length	Length of area3 in bytes (0 to 8064, 0x00 to 0x1F80)
Zone4Length	Length of area4 in bytes (0 to 8000, 0x00 to 0x1F40)

Returns

`int32_t enum status.`

BSP_NFCTAG_DeInit()

`void BSP_NFCTAG_DeInit (const uint32_t Instance )`

Deinitializes peripherals used by the I²C NFCTAG driver.

BSP_NFCTAG_GetByteSize()

`uint32_t BSP_NFCTAG_GetByteSize (const uint32_t Instance )`

Return the size of the nfctag.

Table 83. BSP_NFCTAG_GetByteSize() return values

Size	of the NFCtag in Bytes

BSP_NFCTAG_GetEHENMode_Dyn()

`int32_t BSP_NFCTAG_GetEHENMode_Dyn (const uint32_t Instance,void *const pEH_Val )`

Reads the energy harvesting dynamic status.

Table 84. BSP_NFCTAG_GetEHENMode_Dyn() parameters

pEH_Val	Pointer used to return the energy harvesting dynamic status.

Returns

`int32_t enum status.`

BSP_NFCTAG_GetEHON_Dyn()

`int32_t BSP_NFCTAG_GetEHON_Dyn (const uint32_t Instance,void *const pEHON )`

Reads the EH_ON status from the EH_CTRL_DYN register.

Table 85. BSP_NFCTAG_GetEHON_Dyn() parameters

pEHON	Pointer used to return the EHON status.

Returns

`int32_t enum status.`

BSP_NFCTAG_GetGPO_en_Dyn()

`int32_t BSP_NFCTAG_GetGPO_en_Dyn (const uint32_t Instance,void *const pGPO_en )`

Get dynamique GPO enable status.

Table 86. BSP_NFCTAG_GetGPO_en_Dyn() parameters

pGPO_en	Pointer of the GPO enable status to store

Table 87. BSP_NFCTAG_GetGPO_en_Dyn() return values

NFCTAG	enum status

BSP_NFCTAG_GetITStatus()

`int32_t BSP_NFCTAG_GetITStatus (const uint32_t Instance,uint16_t *const ITConfig )`

Get NFCTAG interrupt configutration.

Table 88. **BSP_NFCTAG_GetITStatus() parameters**

ITConfig	: store interrupt configuration • 0x01 => RF BUSY • 0x02 => WIP

Table 89. **BSP_NFCTAG_GetITStatus() return values**

NFCTAG	enum status

BSP_NFCTAG_GetMBEN_Dyn()

`int32_t BSP_NFCTAG_GetMBEN_Dyn (const uint32_t Instance,void *const pMBEN )`

Reads the mailbox enable dynamic configuration.

Returns

`int32_t enum status.`

BSP_NFCTAG_GetRFDisable()

`int32_t BSP_NFCTAG_GetRFDisable (const uint32_t Instance,void *const pRFDisable )`

Reads the RF disable register information.

Table 90. **BSP_NFCTAG_GetRFDisable() parameters**

pRFDisable	Pointer corresponding to the RF disable status.

Returns

`int32_t enum status.`

BSP_NFCTAG_GetRFDisable_Dyn()

`int32_t BSP_NFCTAG_GetRFDisable_Dyn (const uint32_t Instance,void *const pRFDisable )`

Reads the RF disable dynamic register information.

Table 91. **BSP_NFCTAG_GetRFDisable_Dyn() parameters**

pRFDisable	Pointer corresponding to the RF disable status.

Returns

int32_t enum status.

BSP_NFCTAG_GetRFField_Dyn()

`int32_t BSP_NFCTAG_GetRFField_Dyn (const uint32_t Instance,void *const pRF_Field )`

Checks if RF field is present in front of the ST25DV-I2C.

Table 92. **BSP_NFCTAG_GetRFField_Dyn() parameters**

pRF_Field	Pointer used to return the field presence.

Returns

int32_t enum status.

BSP_NFCTAG_GetRFSleep()

`int32_t BSP_NFCTAG_GetRFSleep (const uint32_t Instance,void *const pRFSleep )`

Reads the RF sleep register information.

Table 93. BSP_NFCTAG_GetRFSleep() parameters

pRFSleep	Pointer corresponding to the RF sleep status.

Returns
int32_t enum status.

BSP_NFCTAG_GetRFSleep_Dyn()

```
int32_t BSP_NFCTAG_GetRFSleep_Dyn (const uint32_t Instance,void *const pRFSleep )
```

Reads the RF sleep dynamic register information.

Table 94. BSP_NFCTAG_GetRFSleep_Dyn() parameters

pRFSleep	Pointer used to return the RF sleep state.

Returns
int32_t enum status.

BSP_NFCTAG_GetVCC_Dyn()

```
int32_t BSP_NFCTAG_GetVCC_Dyn (const uint32_t Instance,void *const pVCC )
```

Check if V_{CC} is supplying the ST25DV-I2C.

Table 95. BSP_NFCTAG_GetVCC_Dyn() parameters

pV$_{CC}$	Pointer of the V_{CC} status to store

Table 96. BSP_NFCTAG_GetVCC_Dyn() return values

NFCTAG	enum status

BSP_NFCTAG_Init()

```
int32_t BSP_NFCTAG_Init (const uint32_t Instance )
```

Initializes peripherals used by the I²C NFCTAG driver.

Returns
Error code.

BSP_NFCTAG_InitEndZone()

```
int32_t BSP_NFCTAG_InitEndZone (const uint32_t Instance )
```

Initializes the end address of the ST25DV-I2C areas with their default values (end of memory).

Needs the I²C password presentation to be effective. The ST25DV-I2C answers a NACK when setting the End←, Zone2 and EndZone3 to same value than repectively EndZone1 and EndZone2. These NACKs are ok.

Returns
int32_t enum status.

BSP_NFCTAG_IsDeviceReady()

```
int32_t BSP_NFCTAG_IsDeviceReady (const uint32_t Instance,const uint32_t Trials )
```

Check if the nfctag is available.

Table 97. **BSP_NFCTAG_IsDeviceReady() parameters**

Trials	: Number of trials

Table 98. **BSP_NFCTAG_IsDeviceReady() return values**

NFCTAG	enum status

BSP_NFCTAG_isInitialized()

`int32_t BSP_NFCTAG_isInitialized (const uint32_t Instance )`

Check if the NFCTAG is initialized.

Table 99. **BSP_NFCTAG_isInitialized() return values**

0	if the NFCTAG is not initialized
1	if the NFCTAG is already initialized

BSP_NFCTAG_PresentI2CPassword()

`int32_t BSP_NFCTAG_PresentI2CPassword (const uint32_t Instance,const void *const PassWord )`

Presents I²C password, to authorize the I²C writes to protected areas.

Table 100. **BSP_NFCTAG_PresentI2CPassword() parameters**

PassWord	Password value on 32 bits

Returns

int32_t enum status.

BSP_NFCTAG_ReadAFI()

`int32_t BSP_NFCTAG_ReadAFI (const uint32_t Instance,uint8_t *const pAfi)`

Reads the ST25DV-I2C AFI.

Table 101. **BSP_NFCTAG_ReadAFI() parameters**

pAfi	Pointer used to return the ST25DV-I2C AFI value.

Returns

int32_t enum status.

BSP_NFCTAG_ReadAfiRFProtection()

`int32_t BSP_NFCTAG_ReadAfiRFProtection (const uint32_t Instance,void *const pLockAfi )`

Reads the AFI RF Lock state.

Table 102. **BSP_NFCTAG_ReadAfiRFProtection() parameters**

pLockAfi	Pointer used to return the ASFID lock state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadData()

`int32_t BSP_NFCTAG_ReadData (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size )`

Reads data in the nfctag at specific address.

Table 103. BSP_NFCTAG_ReadData() parameters

pData	: pointer to store read data
TarAddr	: I2C data memory address to read
Size	: Size in bytes of the value to be read

Table 104. BSP_NFCTAG_ReadData() return values

NFCTAG	enum status

BSP_NFCTAG_ReadDSFID()

`int32_t BSP_NFCTAG_ReadDSFID (const uint32_t Instance,uint8_t *const pDsfid )`

Reads the ST25DV-I2C DSFID.

Table 105. BSP_NFCTAG_ReadDSFID() parameters

pDsfid	Pointer used to return the ST25DV-I2C DSFID value.

Returns

int32_t enum status.

BSP_NFCTAG_ReadDsfidRFProtection()

`int32_t BSP_NFCTAG_ReadDsfidRFProtection (const uint32_t Instance,void *const pLockDsfid )`

Reads the ST25DV-I2C DSFID RF Lock state.

Table 106. BSP_NFCTAG_ReadDsfidRFProtection() parameters

pLockDsfid	Pointer used to return the DSFID lock state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadEHCtrl_Dyn()

`int32_t BSP_NFCTAG_ReadEHCtrl_Dyn (const uint32_t Instance,void *const pEH_CTRL )`

Read value of dynamic EH Ctrl register configuration.

Table 107. BSP_NFCTAG_ReadEHCtrl_Dyn() parameters

pEH_CTRL	: pointer of the dynamic EH Ctrl configuration to store

Table 108. BSP_NFCTAG_ReadEHCtrl_Dyn() return values

NFCTAG	enum status

BSP_NFCTAG_ReadEHMode()

`int32_t BSP_NFCTAG_ReadEHMode (const uint32_t Instance,void *const pEH_mode )`

Reads the energy harvesting mode.

Table 109. BSP_NFCTAG_ReadEHMode() parameters

pEH_mode	Pointer corresponding to the Energy Harvesting state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadEndZonex()

`int32_t BSP_NFCTAG_ReadEndZonex (const uint32_t Instance,const uint8_t EndZone, uint8_t * pEndZ ) Reads the value of the end area address.`

Table 110. BSP_NFCTAG_ReadEndZonex() parameters

EndZone	value corresponding to an area end address.
pEndZ	Pointer used to return the end address of the area.

Returns

int32_t enum status.

BSP_NFCTAG_ReadGPO_Dyn()

`int32_t BSP_NFCTAG_ReadGPO_Dyn (const uint32_t Instance,uint8_t * GPOConfig )`

Read value of dynamic GPO register configuration.

Table 111. BSP_NFCTAG_ReadGPO_Dyn() parameters

pGPO	pointer of the dynamic GPO configuration to store.

Table 112. BSP_NFCTAG_ReadGPO_Dyn() return values

NFCTAG	enum status

BSP_NFCTAG_ReadI2CProtectZone()

`int32_t BSP_NFCTAG_ReadI2CProtectZone (const uint32_t Instance,void *const pProtZone )`

Reads the I²C protected area state.

Table 113. BSP_NFCTAG_ReadI2CProtectZone() parameters

pProtZone	Pointer used to return the protected area state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadI2CSecuritySession_Dyn()

`int32_t BSP_NFCTAG_ReadI2CSecuritySession_Dyn ( const uint32_t Instance, void *const pSession )`

Reads the status of the security session open register.

Table 114. BSP_NFCTAG_ReadI2CSecuritySession_Dyn() parameters

pSession	Pointer used to return the session status.

Returns
int32_t enum status.

BSP_NFCTAG_ReadICRev()

`int32_t BSP_NFCTAG_ReadICRev (const uint32_t Instance,uint8_t *const pICRev )`
Reads the ST25DV-I2C IC revision.

Table 115. BSP_NFCTAG_ReadICRev() parameters

pICRev	Pointer on the uint8_t used to return the ST25DV-I2C IC revision number.

Returns
int32_t enum status.

BSP_NFCTAG_ReadID()

`int32_t BSP_NFCTAG_ReadID (const uint32_t Instance,uint8_t *const wai_id )`
Read the ID of the NFCTAG.

Table 116. BSP_NFCTAG_ReadID() parameters

wai←,_id	: the pointer where the who_am_i of the device is stored

Table 117. BSP_NFCTAG_ReadID() return values

NFCTAG	enum status

BSP_NFCTAG_ReadITPulse()

`int32_t BSP_NFCTAG_ReadITPulse (const uint32_t Instance,void *const pITtime )`
Reads the ST25DV-I2C ITtime duration for the GPO pulses.

Table 118. BSP_NFCTAG_ReadITPulse() parameters

pITtime	Pointer to return the GPO pulse duration coefficient = 302,06 us - ITtime ∗ 512 / fc.

Returns
int32_t enum status.

BSP_NFCTAG_ReadITSTStatus_Dyn()

`int32_t BSP_NFCTAG_ReadITSTStatus_Dyn (const uint32_t Instance, uint8_t *const pITStatus )`
Reads the IT status register from the ST25DV-I2C.

Table 119. **BSP_NFCTAG_ReadITSTStatus_Dyn() parameters**

pITStatus	Pointer on uint8_t, used to return the IT status, such as: • RFUSERSTATE = 0x01 • RFBUSY = 0x02 • RFINTERRUPT = 0x04 • FIELDFALLING = 0x08 • FIELDRISING = 0x10 • RFPUTMSG = 0x20 • RFGETMSG = 0x40 • RFWRITE = 0x80

Returns

int32_t enum status.

BSP_NFCTAG_ReadLockCCFile()

`int32_t BSP_NFCTAG_ReadLockCCFile (const uint32_t Instance,void *const pLockCCFile )`

Reads the CCfile protection state.

Table 120. **BSP_NFCTAG_ReadLockCCFile() parameters**

pLockCCFile	Pointer corresponding to the lock state of the CCFile.

Returns

int32_t enum status.

BSP_NFCTAG_ReadLockCFG()

`int32_t BSP_NFCTAG_ReadLockCFG (const uint32_t Instance,void *const pLockCfg )`

Reads the Cfg registers protection.

Table 121. **BSP_NFCTAG_ReadLockCFG() parameters**

pLockCfg	Pointer corresponding to the Cfg registers lock state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMailboxData()

`int32_t BSP_NFCTAG_ReadMailboxData (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr,const uint16_t NbByte )`

Reads N bytes of data from the mailbox, starting at the specified byte offset.

Table 122. **BSP_NFCTAG_ReadMailboxData() parameters**

pData	Pointer on the buffer used to return the read data.
Offset	Offset in the Mailbox memory, byte number to start the read.
NbByte	Number of bytes to be read.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMailboxRegister()

`int32_t BSP_NFCTAG_ReadMailboxRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )`

Reads N bytes from the mailbox registers, starting at the specified I²C address.

Table 123. BSP_NFCTAG_ReadMailboxRegister() parameters

pData	Pointer on the buffer used to return the data.
TarAddr	I2C memory address to be read.
NbByte	Number of bytes to be read.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMBCtrl_Dyn()

`int32_t BSP_NFCTAG_ReadMBCtrl_Dyn (const uint32_t Instance,void *const pCtrlStatus )`

Reads the mailbox ctrl dynamic register.

Table 124. BSP_NFCTAG_ReadMBCtrl_Dyn() parameters

pCtrlStatus	Pointer structure used to return the dynamic Mailbox ctrl information.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMBLength_Dyn()

`int32_t BSP_NFCTAG_ReadMBLength_Dyn (const uint32_t Instance,uint8_t *const pMBLength )`

Reads the mailbox message length dynamic register.

Table 125. BSP_NFCTAG_ReadMBLength_Dyn() parameters

pMBLength	Pointer on a uint8_t used to return the Mailbox message length.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMBMode()

`int32_t BSP_NFCTAG_ReadMBMode (const uint32_t Instance, void *const pMB_mode )`

Reads the mailbox mode.

Table 126. BSP_NFCTAG_ReadMBMode() parameters

pMB_mode	Pointer used to return the Mailbox mode.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMBWDG()

`int32_t BSP_NFCTAG_ReadMBWDG (const uint32_t Instance, uint8_t *const pWdgDelay )`

Reads the mailbox watchdog duration coefficient.

Table 127. BSP_NFCTAG_ReadMBWDG() parameters

pWdgDelay	Pointer on a uint8_t used to return the watchdog duration coefficient.

Returns

int32_t enum status.

BSP_NFCTAG_ReadMemSize()

`int32_t BSP_NFCTAG_ReadMemSize (const uint32_t Instance, void *const pSizeInfo )`

Reads the ST25DV-I2C memory size.

Table 128. BSP_NFCTAG_ReadMemSize() parameters

pSizeInfo	Pointer used to return the Memory size information.

Returns

int32_t enum status.

BSP_NFCTAG_ReadRegister()

`int32_t BSP_NFCTAG_ReadRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size )`

Reads NFCTAG register.

Table 129. BSP_NFCTAG_ReadRegister() parameters

pData	: pointer to store read data
TarAddr	: I²C register address to read
Size	: Size in bytes of the value to be read

Table 130. BSP_NFCTAG_ReadRegister() return values

NFCTAG	enum status

BSP_NFCTAG_ReadRFMngt()

`int32_t BSP_NFCTAG_ReadRFMngt (const uint32_t Instance, void *const pRF_Mngt )`

Reads the RF management configuration.

Table 131. BSP_NFCTAG_ReadRFMngt() parameters

pRF_Mngt	Pointer used to return the RF Management configuration.

Returns

int32_t enum status.

BSP_NFCTAG_ReadRFMngt_Dyn()

`int32_t BSP_NFCTAG_ReadRFMngt_Dyn (const uint32_t Instance, void *const pRF_Mngt )`

Read value of dynamic RF management configuration.

Table 132. BSP_NFCTAG_ReadRFMngt_Dyn() parameters

pRF_Mngt	pointer of the dynamic RF Management configuration to store

Table 133. BSP_NFCTAG_ReadRFMngt_Dyn() return values

NFCTAG	enum status

BSP_NFCTAG_ReadRFZxSS()

`int32_t BSP_NFCTAG_ReadRFZxSS (const uint32_t Instance,const uint8_t Zone,void *const pRfprotZone )`

Reads the RF zone security status (defining the allowed RF accesses).

Table 134. BSP_NFCTAG_ReadRFZxSS() parameters

Zone	value coresponding to the protected area.
pRfprotZone	Pointer corresponding to the area protection state.

Returns

int32_t enum status.

BSP_NFCTAG_ReadUID()

`int32_t BSP_NFCTAG_ReadUID (const uint32_t Instance,void *const pUid )`

Reads the ST25DV-I2C UID.

Table 135. BSP_NFCTAG_ReadUID() parameters

pUid	Pointer used to return the ST25DV-I2C UID value.

Returns

int32_t enum status.

BSP_NFCTAG_ResetEHENMode_Dyn()

`int32_t BSP_NFCTAG_ResetEHENMode_Dyn (const uint32_t Instance )`

Dynamically unsets the energy harvesting mode.

Returns

int32_t enum status.

BSP_NFCTAG_ResetGPO_en_Dyn()

`int32_t BSP_NFCTAG_ResetGPO_en_Dyn (const uint32_t Instance )`

Reset dynamique GPO enable configuration.

Table 136. BSP_NFCTAG_ResetGPO_en_Dyn() return values

NFCTAG	enum status.

BSP_NFCTAG_ResetMBEN_Dyn()

`int32_t BSP_NFCTAG_ResetMBEN_Dyn (const uint32_t Instance )`

Unsets the mailbox enable dynamic configuration.

Returns

int32_t enum status.

BSP_NFCTAG_ResetRFDisable()

`int32_t BSP_NFCTAG_ResetRFDisable (const uint32_t Instance )`

Resets the RF disable configuration.

Needs the I²C password presentation to be effective.

Returns

int32_t enum status.

BSP_NFCTAG_ResetRFDisable_Dyn()

`int32_t BSP_NFCTAG_ResetRFDisable_Dyn (const uint32_t Instance )`

Unsets the RF disable dynamic configuration.

Returns

int32_t enum status.

BSP_NFCTAG_ResetRFSleep()

`int32_t BSP_NFCTAG_ResetRFSleep (const uint32_t Instance )`

Resets the RF Sleep configuration.

Needs the I²C password presentation to be effective.

Returns

int32_t enum status.

BSP_NFCTAG_ResetRFSleep_Dyn()

`int32_t BSP_NFCTAG_ResetRFSleep_Dyn (const uint32_t Instance )`

Unsets the RF Sleep dynamic configuration.

Returns

int32_t enum status.

BSP_NFCTAG_SetEHENMode_Dyn()

`int32_t BSP_NFCTAG_SetEHENMode_Dyn (const uint32_t Instance )`

Dynamically sets the Energy Harvesting mode.

Returns

int32_t enum status.

BSP_NFCTAG_SetGPO_en_Dyn()

`int32_t BSP_NFCTAG_SetGPO_en_Dyn (const uint32_t Instance )`

Set dynamique GPO enable configuration.

Table 137. BSP_NFCTAG_SetGPO_en_Dyn() return values

NFCTAG	enum status.

BSP_NFCTAG_SetMBEN_Dyn()

`int32_t BSP_NFCTAG_SetMBEN_Dyn (const uint32_t Instance )`

Sets the mailbox enable dynamic configuration.

Returns
int32_t enum status.

BSP_NFCTAG_SetRFDisable()

`int32_t BSP_NFCTAG_SetRFDisable (const uint32_t Instance )`
Sets the RF disable configuration.
Needs the I²C password presentation to be effective.
Returns
int32_t enum status.

BSP_NFCTAG_SetRFDisable_Dyn()

`int32_t BSP_NFCTAG_SetRFDisable_Dyn (const uint32_t Instance )`
Sets the RF Disable dynamic configuration.
Returns
int32_t enum status.

BSP_NFCTAG_SetRFSleep()

`int32_t BSP_NFCTAG_SetRFSleep (const uint32_t Instance )`
Sets the RF Sleep configuration.
Needs the I²C Password presentation to be effective.
Returns
int32_t enum status.

BSP_NFCTAG_SetRFSleep_Dyn()

`int32_t BSP_NFCTAG_SetRFSleep_Dyn (const uint32_t Instance )`
Sets the RF Sleep dynamic configuration.
Returns
int32_t enum status.

BSP_NFCTAG_WriteData()

`int32_t BSP_NFCTAG_WriteData (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size )`
Writes data in the nfctag at specific address.

Table 138. BSP_NFCTAG_WriteData() parameters

pData	: pointer to the data to write
TarAddr	: I2C data memory address to write
Size	: Size in bytes of the value to be written

Table 139. BSP_NFCTAG_WriteData() return values

NFCTAG	enum status.

BSP_NFCTAG_WriteEHMode()

`int32_t BSP_NFCTAG_WriteEHMode (const uint32_t Instance,const uint8_t EH_mode )`
Sets the Energy harvesting mode.
Needs the I²C Password presentation to be effective.

Table 140. **BSP_NFCTAG_WriteEHMode() parameters**

EH_mode	value for the Energy harvesting mode to be set.

Returns

int32_t enum status.

BSP_NFCTAG_WriteEndZonex()

`int32_t BSP_NFCTAG_WriteEndZonex (const uint32_t Instance, const uint8_t EndZone, const uint8_t EndZ )`

Sets the end address of an area.

Needs the I2C Password presentation to be effective.

Note: The ST25DV-I2C answers a NACK when setting the EndZone2 and EndZone3 to same value than repectively EndZone1 and EndZone2.

These NACKs are ok.

Table 141. **BSP_NFCTAG_WriteEndZonex() parameters**

EndZone	value corresponding to an area.
EndZ	End zone value to be written.

Returns

int32_t enum status.

BSP_NFCTAG_WriteI2CPassword()

`int32_t BSP_NFCTAG_WriteI2CPassword (const uint32_t Instance,const void *const PassWord )`

Writes a new I²C password.

Needs the I²C Password presentation to be effective.

Table 142. **BSP_NFCTAG_WriteI2CPassword() parameters**

PassWord	New I²C PassWord value on 32bits.

Returns

int32_t enum status.

BSP_NFCTAG_WriteI2CProtectZonex()

`int32_t BSP_NFCTAG_WriteI2CProtectZonex (const uint32_t Instance, const uint8_t Zone, const uint8_t ReadWriteProtection )`

Sets the I²C write-protected state to an EEPROM Area. Needs the I²C Password presentation to be effective.

Table 143. **BSP_NFCTAG_WriteI2CProtectZonex() parameters**

Zone	value coresponding to the area to protect.
ReadWriteProtection	value corresponding to the protection to be set.

Returns

int32_t enum status.

BSP_NFCTAG_WriteITPulse()

int32_t BSP_NFCTAG_WriteITPulse (const uint32_t *Instance*,const uint8_t *ITtime*)

Configures the ST25DV-I2C ITtime duration for the GPO pulse. Needs the I2C Password presentation to be effective.

Table 144. BSP_NFCTAG_WriteITPulse() parameters

ITtime	Coefficient for the Pulse duration to be written (Pulse duration = 302,06 us - ITtime ∗ 512 / fc)

Table 145. BSP_NFCTAG_WriteITPulse() return values

int32←,_t	enum status.

BSP_NFCTAG_WriteLockCCFile()

`int32_t BSP_NFCTAG_WriteLockCCFile (const uint32_t Instance, const uint8_t NbBlockCCFile, const uint8_t LockCCFile )`

Locks the CCfile to prevent any RF write access. Needs the I²C Password presentation to be effective.

Table 146. BSP_NFCTAG_WriteLockCCFile() parameters

NbBlockCCFile	value corresponding to the number of blocks to be locked.
LockCCFile	value corresponding to the lock state to apply on the CCFile.

Returns

int32_t enum status.

BSP_NFCTAG_WriteLockCFG()

`int32_t BSP_NFCTAG_WriteLockCFG (const uint32_t Instance,const uint8_t LockCfg )`

Lock/Unlock the Cfg registers, to prevent any RF write access. Needs the I²C Password presentation to be effective.

Table 147. BSP_NFCTAG_WriteLockCFG() parameters

LockCfg	value corresponding to the lock state to be written.

Returns

int32_t enum status.

BSP_NFCTAG_WriteMailboxData()

`int32_t BSP_NFCTAG_WriteMailboxData (const uint32_t Instance, const uint8_t ∗const pData, const uint16_t NbByte )`

Writes N bytes of data in the mailbox, starting from first mailbox address.

Table 148. BSP_NFCTAG_WriteMailboxData() parameters

pData	Pointer to the buffer containing the data to be written.
NbByte	Number of bytes to be written.

Returns

int32_t enum status.

BSP_NFCTAG_WriteMailboxRegister()

int32_t BSP_NFCTAG_WriteMailboxRegister (const uint32_t *Instance*, const uint8_t ∗const *pData*, const uint16_t *TarAddr*, const uint16_t *NbByte*)

Writes N bytes to the specified mailbox register.

Table 149. BSP_NFCTAG_WriteMailboxRegister() parameters

pData	Pointer on the data to be written.
TarAddr	I²C register address to be written.
NbByte	Number of bytes to be written.

Returns

int32_t enum status.

BSP_NFCTAG_WriteMBMode()

`int32_t BSP_NFCTAG_WriteMBMode (const uint32_t Instance, const uint8_t MB_mode )`

Sets the Mailbox mode.

Needs the I²C Password presentation to be effective.

Table 150. BSP_NFCTAG_WriteMBMode() parameters

MB_mode	value corresponding to the Mailbox mode to be set.

Returns

int32_t enum status.

BSP_NFCTAG_WriteMBWDG()

`int32_t BSP_NFCTAG_WriteMBWDG (const uint32_t Instance, const uint8_t WdgDelay )`

Writes the mailbox watchdog coefficient delay. Needs the I²C Password presentation to be effective.

Table 151. BSP_NFCTAG_WriteMBWDG() parameters

WdgDelay	Watchdog duration coefficient to be written (Watch dog duration = MB_WDG∗30 ms +/- 6%).

Returns

int32_t enum status.

BSP_NFCTAG_WriteRegister()

`int32_t BSP_NFCTAG_WriteRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr,const uint16_t Size )`

Writes NFCTAG Register.

Table 152. BSP_NFCTAG_WriteRegister() parameters

pData	: pointer to the data to write
TarAddr	: I2C register address to write
Size	: Size in bytes of the value to be written

Table 153. **BSP_NFCTAG_WriteRegister() return values**

NFCTAG	enum status

BSP_NFCTAG_WriteRFMngt()

int32_t BSP_NFCTAG_WriteRFMngt (const uint32_t *Instance*,const uint8_t *Rfmngt*)
Sets the RF management configuration.
Needs the I²C Password presentation to be effective.

Table 154. **BSP_NFCTAG_WriteRFMngt() parameters**

Rfmngt	Value of the RF Management configuration to be written.

Returns
int32_t enum status.

BSP_NFCTAG_WriteRFMngt_Dyn()

`int32_t BSP_NFCTAG_WriteRFMngt_Dyn (const uint32_t Instance, const uint8_t RF_Mngt )`
Writes a value to the RF management dynamic register.

Table 155. **BSP_NFCTAG_WriteRFMngt_Dyn() parameters**

RF_Mngt	Value to be written to the RF Management dynamic register.

Returns
int32_t enum status.

BSP_NFCTAG_WriteRFZxSS()

`int32_t BSP_NFCTAG_WriteRFZxSS (const uint32_t Instance, const uint8_t Zone, const void *const RfProtZone )`
Writes the RF Zone Security Status (defining the allowed RF accesses).
Needs the I²C Password presentation to be effective.

Table 156. **BSP_NFCTAG_WriteRFZxSS() parameters**

Zone	value corresponding to the area on which to set the RF protection.
RfProtZone	Pointer defining the protection to be set on the area.

Returns
int32_t enum status.

6 ST25DVxxKC driver

This module implements the functions to drive the ST25DVxxKC NFC dynamic tag.

Macros

- `#define ST25DVXXKC_MAX_INSTANCE 1`

  This component driver only supports 1 instance of the component.

Functions

- `int32_t ST25DVxxKC_Init (ST25DVxxKC_Object_t *const pObj)`

  ST25DVxxKC nfctag Initialization.
- `int32_t ST25DVxxKC_ReadID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRef)`

  Reads the ST25DVxxKC ID.
- `int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRev)`

  Reads the ST25DVxxKC IC Revision.
- `int32_t ST25DVxxKC_IsDeviceReady (const ST25DVxxKC_Object_t *const pObj, const uint32_t Trials)`

  Checks the ST25DVxxKC availability.
- `int32_t ST25DVxxKC_GetGPOStatus (const ST25DVxxKC_Object_t *const pObj, uint16_t *const p←, GPOStatus)`

  Reads the ST25DVxxKC GPO configuration.
- `int32_t ST25DVxxKC_ConfigureGPO (const ST25DVxxKC_Object_t *const pObj, const uint16_t ITConf)`

  Configures the ST25DVxxKC GPO.
- `int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_PULSE_DURATION_E *const pITtime)`

  Reads the ST25DVxxKC ITtime duration for the GPO pulses.
- `int32_t ST25DVxxKC_WriteITPulse (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PULSE_DURATION_E ITtime)`

  Configures the ST25DVxxKC ITtime duration for the GPO pulse.
- `int32_t ST25DVxxKC_ReadData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes of Data, starting from the specified I2C address.
- `int32_t ST25DVxxKC_WriteData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes of Data starting from the specified I2C Address.
- `int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes from Registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes to the specified register.
- `int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t *const p←, Uid)`

  Reads the ST25DVxxKC UID.
- `int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid)`

  Reads the ST25DVxxKC DSFID.

- `int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid)`

  Reads the ST25DVxxKC DSFID RF Lock state.
- `int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi)`

  Reads the ST25DVxxKC AFI.
- `int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi)`

  Reads the AFI RF Lock state.
- `int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone)`

  Reads the I2C Protected Area state.
- `int32_t ST25DVxxKC_WriteI2CProtectZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ Zone, const ST25DVxxKC_PROTECTION_CONF_E ReadWriteProtection)`

  Sets the I2C write-protected state to an EEPROM Area.
- `int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile)`

  Reads the CCile protection state.
- `int32_t ST25DVxxKC_WriteLockCCFile (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile, const ST25DVxxKC_LOCK_STATUS_E LockCCFile)`

  Locks the CCile to prevent any RF write access.
- `int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockCfg)`

  Reads the Cfg registers protection.
- `int32_t ST25DVxxKC_WriteLockCFG (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_LOCK_STATUS_E LockCfg)`

  Lock/Unlock the Cfg registers, to prevent any RF write access.
- `int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Presents I2C password, to authorize the I2C writes to protected areas.
- `int32_t ST25DVxxKC_WriteI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Writes a new I2C password.
- `int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, ST25DVxxKC_RF_PROT_ZONE_t *const pRfprotZone)`

  Reads the RF Zone Security Status (defining the allowed RF accesses).
- `int32_t ST25DVxxKC_WriteRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, const ST25DVxxKC_RF_PROT_ZONE_t RfProtZone)`

  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, uint8_t *pEndZ)`

  Reads the value of the an area end address.
- `int32_t ST25DVxxKC_WriteEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, const uint8_t EndZ)`

  Sets the end address of an area.
- `int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj)`

  Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).

- `int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, const uint16_t Zone1←, Length, const uint16_t Zone2Length, const uint16_t Zone3Length, const uint16_t Zone4Length)`

  Creates user areas with defined lengths.
- `int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo)`

  Reads the ST25DVxxKC Memory Size.
- `int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode)`

  Reads the Energy harvesting mode.
- `int32_t ST25DVxxKC_WriteEHMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EH_MODE_STATUS_E EH_mode)`

  Sets the Energy harvesting mode.
- `int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`

  Reads the RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt (const ST25DVxxKC_Object_t *const pObj, const uint8_t Rfmngt)`

  Sets the RF Management configuration.
- `int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`

  Reads the RFDisable register information.
- `int32_t ST25DVxxKC_SetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Disable configuration.
- `int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Disable configuration.
- `int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`

  Reads the RFSleep register information.
- `int32_t ST25DVxxKC_SetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMB_mode)`

  Reads the Mailbox mode.
- `int32_t ST25DVxxKC_WriteMBMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EN_STATUS_E MB_mode)`

  Sets the Mailbox mode.
- `int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pWdgDelay)`

  Reads the Mailbox watchdog duration coefficient.
- `int32_t ST25DVxxKC_WriteMBWDG (const ST25DVxxKC_Object_t *const pObj, const uint8_t WdgDelay)`

  Writes the Mailbox watchdog coefficient delay.
- `int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t Offset, const uint16_t NbByte)`

  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- `int32_t ST25DVxxKC_WriteMailboxData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t NbByte)`

  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.

- `int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, Data, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteMailboxRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes to the specified mailbox register.
- `int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession)`

  Reads the status of the security session open register.
- `int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, ITStatus)`

  Reads the IT status register from the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *GPOConfig)`

  Read value of dynamic GPO register configuration.
- `int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pGPO_en)`

  Get dynamique GPO enable status.
- `int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Set dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Reset dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)`

  Read value of dynamic EH Ctrl register configuration.
- `int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEH_Val)`

  Reads the Energy Harvesting dynamic status.
- `int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Dynamically sets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Dynamically unsets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEHON)`

  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_FIELD_STATUS_E *const pRF_Field)`

  Checks if RF Field is present in front of the ST25DVxxKC.
- `int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC)`

  Check if VCC is supplying the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`

  Read value of dynamic RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, const uint8_t RF_←, Mngt)`

  Writes a value to the RF Management dynamic register.
- `int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`

  Reads the RFDisable dynamic register information.

- `int32_t ST25DVxxKC_SetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t ST25DVxxKC_SetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MB_CTRL_DYN_STATUS_*const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, MBLength)`
  Reads the Mailbox message length dynamic register.
- `int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_IO_t *const pIO)`
  Register Component Bus IO operations.
- `int32_t ST25DVxxKC_ReadReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, uint8_t *const Data, const uint16_t len)`
  Read register from component.
- `int32_t ST25DVxxKC_WriteReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, const uint8_t *const Data, const uint16_t len)`
  Write register to component.
- `int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read IC Ref register.
- `int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA1 register.
- `int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA1 register.
- `int32_t ST25DVxxKC_GetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA2 register.
- `int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA2 register.
- `int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA3 register.

- `int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ENDA3 register.
- `int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read DSFID register.
- `int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read AFI register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MEM_SIZE_MSB register.
- `int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read BLK_SIZE register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MEM_SIZE_LSB register.
- `int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ICREV register.
- `int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read UID register.
- `int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CPASSWD register.
- `int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CPASSWD register.
- `int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKDSFI register.
- `int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKAFI register.
- `int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_MODE_RW register.
- `int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_MODE_RW register.
- `int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MBLEN_DYN_MBLEN register.

- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_STRESERVED register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_CURRENTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_DEVICECODE register.
- `int32_t ST25DVxxKC_SetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write I2CCFG_DEVICECODE register.
- `int32_t ST25DVxxKC_GetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_E0 register.
- `int32_t ST25DVxxKC_SetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CCFG_E0 register.
- `int32_t ST25DVxxKC_GetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_RFSWITCHOFF register.
- `int32_t ST25DVxxKC_SetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write I2CCFG_RFSWITCHOFF register.
- `int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ENABLE register.

- `int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ALL register.
- `int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ALL register.
- `int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_I2CWRITEENABLE register.

- `int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_RFOFF register.
- `int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_RFOFF register.
- `int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ITTIME register.
- `int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ITTIME register.
- `int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ALL register.
- `int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ALL register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITTIME_DELAY register.
- `int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ITTIME_DELAY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_FIELDFALLING register.

- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_FIELDRISING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFGETMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFWRITE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_ALL register.
- `int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_MODE register.
- `int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write EH_MODE register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_FIELD_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_VCC_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_RFSLEEP register.

- `int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write RF_MNGT_DYN_RFDIS register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_RFOFF register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write RF_MNGT_DYN_RFOFF register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_PWDCTRL register.

- `int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_ALL register.

- `int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_ALL register.
- `int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCFG_B0 register.

- `int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCFG_B0 register.
- `int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2C_SSO_DYN_I2CSSO register.

Variables

- `ST25DVxxKC_Drv_t St25Dvxxkc_Drv`
  Standard NFC tag driver API for the ST25DVxxKC.

6.1 Detailed description

This module implements the functions to drive the ST25DVxxKC NFC dynamic tag and to access the ST25DVxxKC registers.

As recommended by the STM32 Cube methodology, this driver provides a standard structure to expose the NFC tag standard API.

It also provides an extended API through its extended driver structure.

To be usable on any MCU, this driver calls several IOBus functions. The IOBus functions are implemented outside this driver, and are in charge of accessing the MCU peripherals used for the communication with the tag.

6.2 Function documentation

ST25DVxxKC_ConfigureGPO()

`int32_t ST25DVxxKC_ConfigureGPO (const ST25DVxxKC_Object_t *const pObj,const uint16_t ITConf)`

Configures the ST25DVxxKC GPO.

Needs the I²C password presentation to be effective.

Table 157. ST25DVxxKC_ConfigureGPO() parameters

in	*pObj*	pointer to the device structure object.
in	*ITConf*	Provides the GPO configuration to apply: • RFUSERSTATE = 0x01 • RFBUSY = 0x02 • RFINTERRUPT = 0x04 • FIELDFALLING = 0x08 • FIELDRISING = 0x10 • RFPUTMSG = 0x20 • RFGETMSG = 0x40 • RFWRITE = 0x80

Table 158. ST25DVxxKC_ConfigureGPO() return values

int32←,_t	enum status.

ST25DVxxKC_CreateUserZone()

`int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, const uint16_t Zone1Length, const uint16_t Zone2Length, const uint16_t Zone3Length, const uint16_t Zone4Length )`

Creates user areas with defined lengths.

Needs the I2C Password presentation to be effective.

Table 159. ST25DVxxKC_CreateUserZone() parameters

in	*pObj*	pointer to the device structure object.
in	*Zone1Length*	Length of area1 in bytes (32 to 8192, 0x20 to 0x2000)
in	*Zone2Length*	Length of area2 in bytes (0 to 8128, 0x00 to 0x1FC0)
in	*Zone3Length*	Length of area3 in bytes (0 to 8064, 0x00 to 0x1F80)
in	*Zone4Length*	Length of area4 in bytes (0 to 8000, 0x00 to 0x1F40)

Returns

`int32_t enum status.`

ST25DVxxKC_GetAFI()

`int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read AFI register.

Table 160. ST25DVxxKC_GetAFI() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetBLK_SIZE()

`int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read BLK_SIZE register.

Table 161. ST25DVxxKC_GetBLK_SIZE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetDSFID()

`int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read DSFID register.

Table 162. ST25DVxxKC_GetDSFID() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_ALL()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_CTRL_DYN_ALL register.

Table 163. ST25DVxxKC_GetEH_CTRL_DYN_ALL() parameters

in	*ctx*	structure containing context driver	
out	*value*	data pointer to store register content	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_EH_EN()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_CTRL_DYN_EH_EN register.

Table 164. ST25DVxxKC_GetEH_CTRL_DYN_EH_EN() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_EH_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_CTRL_DYN_EH_ON register.

Table 165. ST25DVxxKC_GetEH_CTRL_DYN_EH_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise.

ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_CTRL_DYN_FIELD_ON register.

Table 166. ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read EH_CTRL_DYN_VCC_ON register.

Table 167. ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_MODE()

```
int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read EH_MODE register.

Table 168. ST25DVxxKC_GetEH_MODE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEHENMode_Dyn()

```
int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const
pObj,ST25DVxxKC_EN_STATUS_E *const pEH_Val )
```

Reads the energy harvesting dynamic status.

Table 169. ST25DVxxKC_GetEHENMode_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pEH_Val*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the Energy Harvesting dynamic status.

Returns

int32_t enum status.

ST25DVxxKC_GetEHON_Dyn()

```
int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EN_STATUS_E *const pEHON ) Reads the EH_ON status from the EH_CTRL_DYN
register.
```

Table 170. ST25DVxxKC_GetEHON_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pEHON*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the EHON status.

Returns

int32_t enum status.

ST25DVxxKC_GetENDA1()

`int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read ENDA1 register.

Table 171. ST25DVxxKC_GetENDA1() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetENDA2()

`int32_t ST25DVxxKC_GetENDA2 ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read ENDA2 register.

Table 172. ST25DVxxKC_GetENDA2() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetENDA3()

`int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read ENDA3 register.

Table 173. ST25DVxxKC_GetENDA3() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_ALL()

`int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read GPO1_ALL register.

Table 174. ST25DVxxKC_GetGPO1_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_ENABLE()

`int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read GPO1_ENABLE register.

Table 175. ST25DVxxKC_GetGPO1_ENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_FIELDCHANGE()

`int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read GPO1_FIELDCHANGE register.

Table 176. ST25DVxxKC_GetGPO1_FIELDCHANGE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFACTIVITY()

`int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read GPO1_RFACTIVITY register.

Table 177. ST25DVxxKC_GetGPO1_RFACTIVITY() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFGETMSG()

`int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read GPO1_RFGETMSG register.

Table 178. ST25DVxxKC_GetGPO1_RFGETMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFINTERRUPT()

```
int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFINTERRUPT register.

Table 179. ST25DVxxKC_GetGPO1_RFINTERRUPT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFPUTMSG()

```
int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFPUTMSG register.

Table 180. ST25DVxxKC_GetGPO1_RFPUTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFUSERSTATE()

```
int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read GPO1_RFUSERSTATE register.

Table 181. ST25DVxxKC_GetGPO1_RFUSERSTATE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFWRITE()

```
int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFWRITE register.

Table 182. ST25DVxxKC_GetGPO1_RFWRITE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_ALL()

`int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read GPO2_ALL register.

Table 183. ST25DVxxKC_GetGPO2_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_I2CWRITEENABLE()

`int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read GPO2_I2CWRITEENABLE register.

Table 184. ST25DVxxKC_GetGPO2_I2CWRITEENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_ITTIME()

`int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read GPO2_ITTIME register.

Table 185. ST25DVxxKC_GetGPO2_ITTIME() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_RFOFF()

`int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read GPO2_RFOFF register.

Table 186. ST25DVxxKC_GetGPO2_RFOFF() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_DYN_ALL()

```
int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read GPO_DYN_ALL register.

Table 187. ST25DVxxKC_GetGPO_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_DYN_ENABLE()

```
int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t
*const value )
```

Read GPO_DYN_ENABLE register.

Table 188. ST25DVxxKC_GetGPO_DYN_ENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_en_Dyn()

```
int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const
pObj,ST25DVxxKC_EN_STATUS_E *const pGPO_en )
```

Get dynamique GPO enable status.

Table 189. ST25DVxxKC_GetGPO_en_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pGPO_en*	ST25DVxxKC_EN_STATUS pointer of the GPO enable status to store.

Table 190. ST25DVxxKC_GetGPO_en_Dyn() return values

NFCTAG	enum status

ST25DVxxKC_GetGPOStatus()

```
int32_t ST25DVxxKC_GetGPOStatus (const ST25DVxxKC_Object_t *const pObj, uint16_t
*const pGPOStatus )
```

Reads the ST25DVxxKC GPO configuration.

Table 191. ST25DVxxKC_GetGPOStatus() parameters

in	*pObj*	pointer to the device structure object.
out	*pGPOStatus*	Pointer on a uint16_t used to return the current GPO consiguration, as:

		• RFUSERSTATE = 0x0001 • RFBUSY = 0x0002 • RFINTERRUPT = 0x0004 • FIELDFALLING = 0x0008 • FIELDRISING = 0x0010 • RFPUTMSG = 0x0020 • RFGETMSG = 0x0040 • RFWRITE = 0x0080 • I2CWRITEENABLE = 0x0100 • RFOFF = 0x0200

Table 192. ST25DVxxKC_GetGPOStatus() return values

int32←,_t	enum status

ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO()

`int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value ) Read I2C_SSO_DYN_I2CSSO register.`

Table 193. ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CCFG_DEVICECODE()

`int32_t ST25DVxxKC_GetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )`

Read I2CCFG_DEVICECODE register.

Table 194. ST25DVxxKC_GetI2CCFG_DEVICECODE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CCFG_E0()

`int32_t ST25DVxxKC_GetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read I2CCFG_E0 register.

Table 195. ST25DVxxKC_GetI2CCFG_E0() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CCFG_RFSWITCHOFF()

```
int32_t ST25DVxxKC_GetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read I2CCFG_RFSWITCHOFF register.

Table 196. ST25DVxxKC_GetI2CCFG_RFSWITCHOFF() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CPASSWD()

```
int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read I2CPASSWD register.

Table 197. ST25DVxxKC_GetI2CPASSWD() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_ALL()

```
int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read I2CSS_ALL register.

Table 198. ST25DVxxKC_GetI2CSS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ1()

```
int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read I2CSS_PZ1 register.

Table 199. ST25DVxxKC_GetI2CSS_PZ1() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ2()

```
int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read I2CSS_PZ2 register.

Table 200. ST25DVxxKC_GetI2CSS_PZ2() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ3()

```
int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_PZ3 register.

Table 201. ST25DVxxKC_GetI2CSS_PZ3() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ4()

```
int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_PZ4 register.

Table 202. ST25DVxxKC_GetI2CSS_PZ4() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetICREF()

```
int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read IC Ref register.

Table 203. ST25DVxxKC_GetICREF() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetICREV()

```
int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read ICREV register.

Table 204. ST25DVxxKC_GetICREV() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_ALL()

```
int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_ALL register.

Table 205. ST25DVxxKC_GetITSTS_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_FIELDFALLING()

```
int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_FIELDFALLING register.

Table 206. ST25DVxxKC_GetITSTS_DYN_FIELDFALLING() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_FIELDRISING()

```
int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_FIELDRISING register.

Table 207. ST25DVxxKC_GetITSTS_DYN_FIELDRISING() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFACTIVITY()

int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t ∗const *ctx,* uint8_t ∗const *value*) Read ITSTS_DYN_RFACTIVITY register.

Table 208. ST25DVxxKC_GetITSTS_DYN_RFACTIVITY() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFGETMSG()

int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t ∗const *ctx,* uint8_t ∗const *value*) Read ITSTS_DYN_RFGETMSG register.

Table 209. ST25DVxxKC_GetITSTS_DYN_RFGETMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read ITSTS_DYN_RFINTERRUPT register.

Table 210. ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFPUTMSG()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read ITSTS_DYN_RFPUTMSG register.

Table 211. ST25DVxxKC_GetITSTS_DYN_RFPUTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read ITSTS_DYN_RFUSERSTATE register.

Table 212. ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFWRITE()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read ITSTS_DYN_RFWRITE register.

Table 213. ST25DVxxKC_GetITSTS_DYN_RFWRITE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITTIME_DELAY()

```
int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read ITTIME_DELAY register.

Table 214. ST25DVxxKC_GetITTIME_DELAY() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKAFI()

```
int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read LOCKAFI register.

Table 215. ST25DVxxKC_GetLOCKAFI() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_ALL()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_ALL register.

Table 216. ST25DVxxKC_GetLOCKCCFILE_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_BLCK0()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_BLCK0 register.

Table 217. ST25DVxxKC_GetLOCKCCFILE_BLCK0() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_BLCK1()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_BLCK1 register.

Table 218. ST25DVxxKC_GetLOCKCCFILE_BLCK1() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCFG_B0()

```
int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCFG_B0 register.

Table 219. ST25DVxxKC_GetLOCKCFG_B0() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKDSFID()

```
int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKDSFI register.

Table 220. ST25DVxxKC_GetLOCKDSFID() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_ALL()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read MB_CTRL_DYN_ALL register.

Table 221. ST25DVxxKC_GetMB_CTRL_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read MB_CTRL_DYN_CURRENTMSG register.

Table 222. ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read MB_CTRL_DYN_HOSTMISSMSG register.

Table 223. ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG()

`int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value ) Read MB_CTRL_DYN_HOSTPUTMSG register.`

Table 224. ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_MBEN()

`int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read MB_CTRL_DYN_MBEN register.

Table 225. ST25DVxxKC_GetMB_CTRL_DYN_MBEN() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG()

`int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read MB_CTRL_DYN_RFMISSMSG register.

Table 226. ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG()

`int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read MB_CTRL_DYN_RFPUTMSG register.

Table 227. ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read MB_CTRL_DYN_STRESERVED register.

Table 228. ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_MODE_RW()

```
int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read MB_MODE_RW register.

Table 229. ST25DVxxKC_GetMB_MODE_RW() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_WDG_DELAY()

```
int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read MB_WDG_DELAY register.

Table 230. ST25DVxxKC_GetMB_WDG_DELAY() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMBEN_Dyn()

```
int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const
pObj,ST25DVxxKC_EN_STATUS_E *const pMBEN )
```

Reads the Mailbox Enable dynamic configuration.

Table 231. ST25DVxxKC_GetMBEN_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pMBEN*	pointer on ST25DVxxKC_EN_STATUS_E values used to return the Mailbox enable state.

Returns

int32_t enum status.

ST25DVxxKC_GetMBLEN_DYN_MBLEN()

`int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read MBLEN_DYN_MBLEN register.

Table 232. ST25DVxxKC_GetMBLEN_DYN_MBLEN() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMEM_SIZE_LSB()

`int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read MEM_SIZE_LSB register.

Table 233. ST25DVxxKC_GetMEM_SIZE_LSB() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMEM_SIZE_MSB()

`int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read MEM_SIZE_MSB register.

Table 234. ST25DVxxKC_GetMEM_SIZE_MSB() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_ALL()

`int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value)`

Read RF_MNGT_ALL register.

Table 235. ST25DVxxKC_GetRF_MNGT_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_ALL()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RF_MNGT_DYN_ALL register.

Table 236. ST25DVxxKC_GetRF_MNGT_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_RFDIS()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value ) Read RF_MNGT_DYN_RFDIS register.
```

Table 237. ST25DVxxKC_GetRF_MNGT_DYN_RFDIS() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_RFOFF()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value ) Read RF_MNGT_DYN_RFOFF register.
```

Table 238. ST25DVxxKC_GetRF_MNGT_DYN_RFOFF() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RF_MNGT_DYN_RFSLEEP register.

Table 239. ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_RFDIS()

```
int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RF_MNGT_RFDIS register.

Table 240. ST25DVxxKC_GetRF_MNGT_RFDIS() rarameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_RFSLEEP()

```
int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RF_MNGT_RFSLEEP register.

Table 241. ST25DVxxKC_GetRF_MNGT_RFSLEEP() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_ALL()

```
int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA1SS_ALL register.

Table 242. ST25DVxxKC_GetRFA1SS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA1SS_PWDCTRL register.

Table 243. ST25DVxxKC_GetRFA1SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA1SS_RWPROT register.

Table 244. ST25DVxxKC_GetRFA1SS_RWPROT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_ALL()

```
int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA2SS_ALL register.

Table 245. ST25DVxxKC_GetRFA2SS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA2SS_PWDCTRL register.

Table 246. ST25DVxxKC_GetRFA2SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA2SS_RWPROT register.

Table 247. ST25DVxxKC_GetRFA2SS_RWPROT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_ALL()

`int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read RFA3SS_ALL register.

Table 248. ST25DVxxKC_GetRFA3SS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_PWDCTRL()

`int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read RFA3SS_PWDCTRL register.

Table 249. ST25DVxxKC_GetRFA3SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_RWPROT()

`int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read RFA3SS_RWPROT register.

Table 250. ST25DVxxKC_GetRFA3SS_RWPROT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_ALL()

`int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

Read RFA4SS_ALL register.

Table 251. ST25DVxxKC_GetRFA4SS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA4SS_PWDCTRL register.

Table 252. ST25DVxxKC_GetRFA4SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA4SS_RWPROT register.

Table 253. ST25DVxxKC_GetRFA4SS_RWPROT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFDisable()

```
int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable )
```

Reads the RFDisable register information.

Table 254. ST25DVxxKC_GetRFDisable() parameters

in	*pObj*	pointer to the device structure object.
out	*pRFDisable*	Pointer on a ST25DVxxKC_EN_STATUS value corresponding to the RF Disable status.

Returns

int32_t enum status.

ST25DVxxKC_GetRFDisable_Dyn()

```
int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)
```

Reads the RFDisable dynamic register information.

Table 255. ST25DVxxKC_GetRFDisable_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pRFDisable*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the RF Disable state.

Returns

int32_t enum status.

ST25DVxxKC_GetRFField_Dyn()

`int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_FIELD_STATUS_E *const pRF_Field)`

Checks if RF Field is present in front of the ST25DVxxKC.

Table 256. ST25DVxxKC_GetRFField_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pRF_Field*	Pointer on a ST25DVxxKC_FIELD_STATUS value used to return the field presence.

Returns

int32_t enum status.

ST25DVxxKC_GetRFSleep()

`int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`

Reads the RFSleep register information.

Table 257. ST25DVxxKC_GetRFSleep() parameters

in	*pObj*	pointer to the device structure object.
out	*pRFSleep*	Pointer on a ST25DVxxKC_EN_STATUS value corresponding to the RF Sleep status.

Returns

int32_t enum status.

ST25DVxxKC_GetRFSleep_Dyn()

`int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`

Reads the RFSleep dynamic register information.

Table 258. ST25DVxxKC_GetRFSleep_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pRFSleep*	Pointer on a ST25DVxxKC_EN_STATUS values used to return the RF Sleep state.

Returns

int32_t enum status.

ST25DVxxKC_GetUID()

`int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )`

Read UID register.

Table 259. ST25DVxxKC_GetUID() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetVCC_Dyn()

`int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC)`

Check if VCC is supplying the ST25DVxxKC.

Table 260. ST25DVxxKC_GetVCC_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pVCC*	ST25DVxxKC_VCC_STATUS pointer of the VCC status to store.

Table 261. ST25DVxxKC_GetVCC_Dyn() return values

NFCTAG	enum status.

ST25DVxxKC_Init()

`int32_t ST25DVxxKC_Init (ST25DVxxKC_Object_t *const pObj)`

ST25DVxxKC nfctag Initialization.

Table 262. ST25DVxxKC_Init() parameters

in,out	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_InitEndZone()

`int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj)`

Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).

Needs the I2C Password presentation to be effective.. The ST25DVxxKC answers a NACK when setting the End←, Zone2 and EndZone3 to same value than repectively EndZone1 & EndZone2. These NACKs are ok.

Table 263. ST25DVxxKC_InitEndZone() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_IsDeviceReady()

`int32_t ST25DVxxKC_IsDeviceReady (const ST25DVxxKC_Object_t *const pObj, const uint32_t Trials)`

Checks the ST25DVxxKC availability.

The ST25DVxxKC I2C is NACKed when a RF communication is on-going. This function determines if the ST25←, DVxxKC is ready to answer an I2C request.

Table 264. ST25DVxxKC_IsDeviceReady() parameters

in	*pObj*	pointer to the device structure object.
in	*Trials*	Max number of tentative.

Table 265. **ST25DVxxKC_IsDeviceReady() return values**

int32←,_t	enum status.

ST25DVxxKC_PresentI2CPassword()

```
int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)
```

Presents I2C password, to authorize the I2C writes to protected areas.

Table 266. **ST25DVxxKC_PresentI2CPassword() parameters**

in	*pObj*	pointer to the device structure object.
in	*PassWord*	Password value on 32bits

Returns

int32_t enum status.

ST25DVxxKC_ReadAFI()

```
int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi )
```

Reads the ST25DVxxKC AFI.

Table 267. **ST25DVxxKC_ReadAFI() parameters**

in	*pObj*	pointer to the device structure object.
out	*pAfi*	Pointer used to return the ST25DVxxKC AFI value.

Returns

int32_t enum status.

ST25DVxxKC_ReadAfiRFProtection()

```
int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi )
```

Reads the AFI RF Lock state.

Table 268. **ST25DVxxKC_ReadAfiRFProtection() parameters**

in	*pObj*	pointer to the device structure object.
out	*pLockAfi*	Pointer on a ST25DVxxKC_LOCK_STATUS used to return the ASFID lock state.

Returns

int32_t enum status.

ST25DVxxKC_ReadData()

```
int32_t ST25DVxxKC_ReadData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )
```

Reads N bytes of Data, starting from the specified I²C address.

Table 269. **ST25DVxxKC_ReadData() parameters**

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer used to return the read data.

in	*TarAddr*	I2C data memory address to read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadDSFID()

```
int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid)
```

Reads the ST25DVxxKC DSFID.

Table 270. ST25DVxxKC_ReadDSFID() parameters

in	*pObj*	pointer to the device structure object.
out	*pDsfid*	Pointer used to return the ST25DVxxKC DSFID value.

Returns

int32_t enum status.

ST25DVxxKC_ReadDsfidRFProtection()

```
int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid)
```

Reads the ST25DVxxKC DSFID RF Lock state.

Table 271. ST25DVxxKC_ReadDsfidRFProtection() parameters

in	*pObj*	pointer to the device structure object.
out	*pLockDsfid*	Pointer on a ST25DVxxKC_LOCK_STATUS used to return the DSFID lock state.

Returns

int32_t enum status.

ST25DVxxKC_ReadEHCtrl_Dyn()

```
int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)
```

Read value of dynamic EH Ctrl register configuration.

Table 272. ST25DVxxKC_ReadEHCtrl_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pEH_CTRL*	: ST25DVxxKC_EH_CTRL pointer of the dynamic EH Ctrl configuration to store.

Table 273. ST25DVxxKC_ReadEHCtrl_Dyn() return values

NFCTAG	enum status

ST25DVxxKC_ReadEHMode()

```
int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode)
```

Reads the energy harvesting mode.

Table 274. **ST25DVxxKC_ReadEHMode() parameters**

in	*pObj*	pointer to the device structure object.
out	*pEH_mode*	Pointer on a ST25DVxxKC_EH_MODE_STATUS value corresponding to the Energy Harvesting state.

Returns

int32_t enum status.

ST25DVxxKC_ReadEndZonex()

```
int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, uint8_t * pEndZ)
```

Reads the value of the an area end address.

Table 275. **ST25DVxxKC_ReadEndZonex() parameters**

in	*pObj*	pointer to the device structure object.
in	*EndZone*	ST25DVxxKC_END_ZONE value corresponding to an area end address.
out	*pEndZ*	Pointer used to return the end address of the area.

Returns

int32_t enum status.

ST25DVxxKC_ReadGPO_Dyn()

```
int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t * GPOConfig )
```

Read value of dynamic GPO register configuration.

Table 276. **ST25DVxxKC_ReadGPO_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pGPO*	ST25DVxxKC_GPO pointer of the dynamic GPO configuration to store.

Table 277. **ST25DVxxKC_ReadGPO_Dyn() return values**

NFCTAG	enum status

ST25DVxxKC_ReadI2CProtectZone()

```
int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone )
```

Reads the I²C Protected Area state.

Table 278. **ST25DVxxKC_ReadI2CProtectZone() parameters**

in	*pObj*	pointer to the device structure object.
out	*pProtZone*	Pointer on a ST25DVxxKC_I2C_PROT_ZONE structure used to return the Protected Area state.

Returns

int32_t enum status.

ST25DVxxKC_ReadI2CSecuritySession_Dyn()

`int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn ( const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession)`

Reads the status of the security session open register.

Table 279. ST25DVxxKC_ReadI2CSecuritySession_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pSession*	Pointer on a ST25DVxxKC_I2CSSO_STATUS value used to return the session status.

Returns

int32_t enum status.

ST25DVxxKC_ReadICRev()

`int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRev)`

Reads the ST25DVxxKC IC revision.

Table 280. ST25DVxxKC_ReadICRev() parameters

in	*pObj*	pointer to the device structure object.
out	*pICRev*	Pointer on the uint8_t used to return the ST25DVxxKC IC Revision number.

Returns

int32_t enum status.

ST25DVxxKC_ReadID()

`int32_t ST25DVxxKC_ReadID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRef)`

Reads the ST25DVxxKC ID.

Table 281. ST25DVxxKC_ReadID() parameters

in	*pObj*	pointer to the device structure object.
out	*pICRef*	Pointer on an uint8_t used to return the ST25DVxxKC ID.

Returns

int32_t enum status.

ST25DVxxKC_ReadITPulse()

`int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_PULSE_DURATION_E *const pITtime)`

Reads the ST25DVxxKC ITtime duration for the GPO pulses.

Table 282. ST25DVxxKC_ReadITPulse() parameters

in	*pObj*	pointer to the device structure object.
out	*pITtime*	Pointer used to return the coefficient for the GPO Pulse duration (Pulse duration = 302,06 us - ITtime * 512 / fc).

Returns

int32_t enum status.

ST25DVxxKC_ReadITSTStatus_Dyn()

`int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pITStatus) Reads the IT status register from the ST25DVxxKC.`

Table 283. ST25DVxxKC_ReadITSTStatus_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pITStatus*	Pointer on uint8_t, used to return the IT status, such as: • RFUSERSTATE = 0x01 • RFBUSY = 0x02 • RFINTERRUPT = 0x04 • FIELDFALLING = 0x08 • FIELDRISING = 0x10 • RFPUTMSG = 0x20 • RFGETMSG = 0x40 • RFWRITE = 0x80

Returns

int32_t enum status.

ST25DVxxKC_ReadLockCCFile()

`int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile )`

Reads the CCile protection state.

Table 284. ST25DVxxKC_ReadLockCCFile() parameters

in	*pObj*	pointer to the device structure object.
out	*pLockCCFile*	Pointer on a ST25DVxxKC_LOCK_CCFILE value corresponding to the lock state of the CCFile.

Returns

int32_t enum status.

ST25DVxxKC_ReadLockCFG()

`int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockCfg)`

Reads the Cfg registers protection.

Table 285. ST25DVxxKC_ReadLockCFG() parameters

in	*pObj*	pointer to the device structure object.
out	*pLockCfg*	Pointer on a ST25DVxxKC_LOCK_STATUS value corresponding to the Cfg registers lock state.

Returns

int32_t enum status.

ST25DVxxKC_ReadMailboxData()

`int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t Offset, const uint16_t NbByte)`

Reads N bytes of data from the Mailbox, starting at the specified byte offset.

Table 286. **ST25DVxxKC_ReadMailboxData() parameters**

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer on the buffer used to return the read data.
in	*Offset*	Offset in the Mailbox memory, byte number to start the read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadMailboxRegister()

```
int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj,
uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)
```

Reads N bytes from the mailbox registers, starting at the specified I²C address.

Table 287. **ST25DVxxKC_ReadMailboxRegister() parameters**

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer on the buffer used to return the data.
in	*TarAddr*	I2C memory address to be read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBCtrl_Dyn()

```
int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const
pObj,ST25DVxxKC_MB_CTRL_DYN_STATUS_t *const pCtrlStatus)
```

Reads the Mailbox ctrl dynamic register.

Table 288. **ST25DVxxKC_ReadMBCtrl_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pCtrlStatus*	Pointer on a ST25DVxxKC_MB_CTRL_DYN_STATUS structure used to return the dynamic Mailbox ctrl information.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBLength_Dyn()

```
int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t
*const pMBLength)
```

Reads the Mailbox message length dynamic register.

Table 289. **ST25DVxxKC_ReadMBLength_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pMBLength*	Pointer on a uint8_t used to return the Mailbox message length.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBMode()

`int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj,ST25DVxxKC_EN_STATUS_E *const pMB_mode)`

Reads the Mailbox mode.

Table 290. ST25DVxxKC_ReadMBMode() parameters

in	*pObj*	pointer to the device structure object.
out	*pMB_mode*	Pointer on a ST25DVxxKC_EH_MODE_STATUS value used to return the Mailbox mode.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBWDG()

`int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj,uint8_t *const pWdgDelay)`

Reads the Mailbox watchdog duration coefficient.

Table 291. ST25DVxxKC_ReadMBWDG() parameters

in	*pObj*	pointer to the device structure object.
out	*pWdgDelay*	Pointer on a uint8_t used to return the watchdog duration coefficient.

Returns

int32_t enum status.

ST25DVxxKC_ReadMemSize()

`int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo)`

Reads the ST25DVxxKC Memory Size.

Table 292. ST25DVxxKC_ReadMemSize() parameters

in	*pObj*	pointer to the device structure object.
out	*pSizeInfo*	Pointer on a ST25DVxxKC_MEM_SIZE structure used to return the Memory size information.

Returns

int32_t enum status.

ST25DVxxKC_ReadReg()

`int32_t ST25DVxxKC_ReadReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, uint8_t *const Data, const uint16_t len)`

Read register from component.

Table 293. ST25DVxxKC_ReadReg() parameters

in	*ctx*	structure containing context driver
in	*Reg*	register to read
out	*Data*	pointer to store register content
out	*len*	length of data

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_ReadRegister()

```
int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t
*const pData, const uint16_t TarAddr, const uint16_t NbByte)
```

Reads N bytes from Registers, starting at the specified I2C address.

Table 294. ST25DVxxKC_ReadRegister() parameters

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer used to return the read data.
in	*TarAddr*	I2C memory address to be read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadRFMngt()

```
int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)
```

Reads the RF Management configuration.

Table 295. ST25DVxxKC_ReadRFMngt() parameters

in	*pObj*	pointer to the device structure object.
out	*pRF_Mngt*	Pointer on a ST25DVxxKC_RF_MNGT structure used to return the RF Management configuration.

Returns

int32_t enum status.

ST25DVxxKC_ReadRFMngt_Dyn()

```
int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)
```

Read value of dynamic RF Management configuration.

Table 296. ST25DVxxKC_ReadRFMngt_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pRF_Mngt*	ST25DVxxKC_RF_MNGT pointer of the dynamic RF Management configuration to store

Table 297. ST25DVxxKC_ReadRFMngt_Dyn() return values

NFCTAG	enum status

ST25DVxxKC_ReadRFZxSS()

```
int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_PROTECTION_ZONE_E Zone, ST25DVxxKC_RF_PROT_ZONE_t *const pRfprotZone )
```

Reads the RF Zone Security Status (defining the allowed RF accesses).

Table 298. **ST25DVxxKC_ReadRFZxSS() parameters**

in	*pObj*	pointer to the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value coresponding to the protected area.
out	*pRfprotZone*	Pointer on a ST25DVxxKC_RF_PROT_ZONE value corresponding to the area protection state.

Returns

int32_t enum status.

ST25DVxxKC_ReadUID()

```
int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t *const pUid)
```

Reads the ST25DVxxKC UID.

Table 299. **ST25DVxxKC_ReadUID() parameters**

in	*pObj*	pointer to the device structure object.
out	*pUid*	Pointer used to return the ST25DVxxKC UID value.

Returns

int32_t enum status.

ST25DVxxKC_RegisterBusIO()

```
int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_IO_t *const pIO)
```

Register Component Bus IO operations.

Table 300. **ST25DVxxKC_RegisterBusIO() parameters**

out	*pObj*	pointer to the device structure object.
in	*pIO*	pointer to the IO APIs structure.

Returns

int32_t enum status.

ST25DVxxKC_ResetEHENMode_Dyn()

```
int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)
```

Dynamically unsets the energy harvesting mode.

Table 301. **ST25DVxxKC_ResetEHENMode_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetGPO_en_Dyn()

```
int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)
```

Reset dynamique GPO enable configuration.

Table 302. **ST25DVxxKC_ResetGPO_en_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Table 303. **ST25DVxxKC_ResetGPO_en_Dyn() return values**

NFCTAG	enum status.

ST25DVxxKC_ResetMBEN_Dyn()

```
int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)
```

Unsets the Mailbox Enable dynamic configuration.

Table 304. **ST25DVxxKC_ResetMBEN_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFDisable()

```
int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj)
```

Resets the RF Disable configuration.

Needs the I²C Password presentation to be effective.

Table 305. **ST25DVxxKC_ResetRFDisable() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFDisable_Dyn()

```
int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj )
```

Unsets the RF Disable dynamic configuration.

Table 306. **ST25DVxxKC_ResetRFDisable_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFSleep()

```
int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj)
```

Resets the RF Sleep configuration.

Needs the I²C Password presentation to be effective.

Table 307. **ST25DVxxKC_ResetRFSleep() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFSleep_Dyn()

`int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`

Unsets the RF Sleep dynamic configuration.

Table 308. ST25DVxxKC_ResetRFSleep_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetEH_CTRL_DYN_EH_EN()

`int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write EH_CTRL_DYN_EH_EN register.

Table 309. ST25DVxxKC_SetEH_CTRL_DYN_EH_EN() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetEH_MODE()

`int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write EH_MODE register.

Table 310. ST25DVxxKC_SetEH_MODE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetEHENMode_Dyn()

`int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Dynamically sets the Energy Harvesting mode.

Table 311. ST25DVxxKC_SetEHENMode_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetENDA1()

```
int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write ENDA1 register.

Table 312. ST25DVxxKC_SetENDA1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetENDA2()

```
int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write ENDA2 register.

Table 313. ST25DVxxKC_SetENDA2() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetENDA3()

```
int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write ENDA3 register.

Table 314. ST25DVxxKC_SetENDA3() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_ALL()

```
int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_ALL register.

Table 315. ST25DVxxKC_SetGPO1_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_ENABLE()

```
int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_ENABLE register.

Table 316. ST25DVxxKC_SetGPO1_ENABLE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_FIELDCHANGE()

```
int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_FIELDCHANGE register.

Table 317. ST25DVxxKC_SetGPO1_FIELDCHANGE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFACTIVITY()

```
int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFACTIVITY register.

Table 318. ST25DVxxKC_SetGPO1_RFACTIVITY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFGETMSG()

```
int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFGETMSG register.

Table 319. ST25DVxxKC_SetGPO1_RFGETMSG() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFINTERRUPT()

```
int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFINTERRUPT register.

Table 320. ST25DVxxKC_SetGPO1_RFINTERRUPT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFPUTMSG()

```
int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFPUTMSG register.

Table 321. ST25DVxxKC_SetGPO1_RFPUTMSG() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFUSERSTATE()

```
int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFUSERSTATE register.

Table 322. ST25DVxxKC_SetGPO1_RFUSERSTATE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFWRITE()

```
int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write GPO1_RFWRITE register.

Table 323. ST25DVxxKC_SetGPO1_RFWRITE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_ALL()

```
int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write GPO2_ALL register.

Table 324. ST25DVxxKC_SetGPO2_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_I2CWRITEENABLE()

```
int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write GPO2_I2CWRITEENABLE register.

Table 325. ST25DVxxKC_SetGPO2_I2CWRITEENABLE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_ITTIME()

```
int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO2_ITTIME register.

Table 326. ST25DVxxKC_SetGPO2_ITTIME() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_RFOFF()

```
int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO2_RFOFF register.

Table 327. ST25DVxxKC_SetGPO2_RFOFF() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_DYN_ALL()

`int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write GPO_DYN_ALL register.

Table 328. ST25DVxxKC_SetGPO_DYN_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_DYN_ENABLE()

`int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write GPO_DYN_ENABLE register.

Table 329. ST25DVxxKC_SetGPO_DYN_ENABLE() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_en_Dyn()

`int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj ) Set dynamique GPO enable configuration.`

Table 330. ST25DVxxKC_SetGPO_en_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Table 331. ST25DVxxKC_SetGPO_en_Dyn() return values

NFCTAG	enum status.

ST25DVxxKC_SetI2CCFG_DEVICECODE()

`int32_t ST25DVxxKC_SetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )`

Write I2CCFG_DEVICECODE register.

Table 332. ST25DVxxKC_SetI2CCFG_DEVICECODE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CCFG_E0()

```
int32_t ST25DVxxKC_SetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CCFG_E0 register.

Table 333. ST25DVxxKC_SetI2CCFG_E0() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CCFG_RFSWITCHOFF()

```
int32_t ST25DVxxKC_SetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CCFG_RFSWITCHOFF register.

Table 334. ST25DVxxKC_SetI2CCFG_RFSWITCHOFF() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CPASSWD()

```
int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CPASSWD register.

Table 335. ST25DVxxKC_SetI2CPASSWD() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_ALL()

```
int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write I2CSS_ALL register.

Table 336. ST25DVxxKC_SetI2CSS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ1()

```
int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ1 register.

Table 337. ST25DVxxKC_SetI2CSS_PZ1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ2()

```
int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ2 register.

Table 338. ST25DVxxKC_SetI2CSS_PZ2() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ3()

```
int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ3 register.

Table 339. ST25DVxxKC_SetI2CSS_PZ3() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ4()

```
int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx,const uint8_t *const value )
```

Write I2CSS_PZ4 register.

Table 340. ST25DVxxKC_SetI2CSS_PZ4() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetITTIME_DELAY()

```
int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write ITTIME_DELAY register.

Table 341. ST25DVxxKC_SetITTIME_DELAY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_ALL()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write LOCKCCFILE_ALL register.

Table 342. ST25DVxxKC_SetLOCKCCFILE_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_BLCK0()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write LOCKCCFILE_BLCK0 register.

Table 343. ST25DVxxKC_SetLOCKCCFILE_BLCK0() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_BLCK1()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write LOCKCCFILE_BLCK1 register.

Table 344. ST25DVxxKC_SetLOCKCCFILE_BLCK1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCFG_B0()

```
int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write LOCKCFG_B0 register.

Table 345. ST25DVxxKC_SetLOCKCFG_B0() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_CTRL_DYN_MBEN()

```
int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_CTRL_DYN_MBEN register.

Table 346. ST25DVxxKC_SetMB_CTRL_DYN_MBEN() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_MODE_RW()

```
int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_MODE_RW register.

Table 347. ST25DVxxKC_SetMB_MODE_RW() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_WDG_DELAY()

```
int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_WDG_DELAY register.

Table 348. ST25DVxxKC_SetMB_WDG_DELAY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMBEN_Dyn()

`int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Sets the Mailbox Enable dynamic configuration.

Table 349. ST25DVxxKC_SetMBEN_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetRF_MNGT_ALL()

`int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write RF_MNGT_ALL register.

Table 350. ST25DVxxKC_SetRF_MNGT_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_ALL()

`int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write RF_MNGT_DYN_ALL register.

Table 351. ST25DVxxKC_SetRF_MNGT_DYN_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_RFDIS()

`int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write RF_MNGT_DYN_RFDIS register.

Table 352. ST25DVxxKC_SetRF_MNGT_DYN_RFDIS() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_RFOFF()

```
int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_DYN_RFOFF register.

Table 353. ST25DVxxKC_SetRF_MNGT_DYN_RFOFF() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP()

```
int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP ( const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_DYN_RFSLEEP register.

Table 354. ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_RFDIS()

```
int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_RFDIS register.

Table 355. ST25DVxxKC_SetRF_MNGT_RFDIS() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_RFSLEEP()

```
int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_RFSLEEP register.

Table 356. ST25DVxxKC_SetRF_MNGT_RFSLEEP() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_ALL()

```
int32_t ST25DVxxKC_SetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_ALL register.

Table 357. ST25DVxxKC_SetRFA1SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_PWDCTRL register.

Table 358. ST25DVxxKC_SetRFA1SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_RWPROT register.

Table 359. ST25DVxxKC_SetRFA1SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_ALL()

```
int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_ALL register.

Table 360. ST25DVxxKC_SetRFA2SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_PWDCTRL register.

Table 361. ST25DVxxKC_SetRFA2SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_RWPROT register.

Table 362. ST25DVxxKC_SetRFA2SS_RWPROT() parameters

in	*ctx*	structure containing context driver	
in	*value*	data pointer to write to register	

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_ALL()

```
int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA3SS_ALL register.

Table 363. ST25DVxxKC_SetRFA3SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA3SS_PWDCTRL register.

Table 364. ST25DVxxKC_SetRFA3SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write RFA3SS_RWPROT register.

Table 365. ST25DVxxKC_SetRFA3SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_ALL()

```
int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t
*const value )
```

Write RFA4SS_ALL register.

Table 366. ST25DVxxKC_SetRFA4SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write RFA4SS_PWDCTRL register.

Table 367. ST25DVxxKC_SetRFA4SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const
uint8_t *const value )
```

Write RFA4SS_RWPROT register.

Table 368. ST25DVxxKC_SetRFA4SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFDisable()

`int32_t ST25DVxxKC_SetRFDisable (const ST25DVxxKC_Object_t *const pObj )`

Sets the RF Disable configuration.

Needs the I2C Password presentation to be effective.

Table 369. ST25DVxxKC_SetRFDisable() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetRFDisable_Dyn()

`int32_t ST25DVxxKC_SetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Sets the RF Disable dynamic configuration.

Table 370. ST25DVxxKC_SetRFDisable_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetRFSleep()

`int32_t ST25DVxxKC_SetRFSleep (const ST25DVxxKC_Object_t *const pObj )`

Sets the RF Sleep configuration.

Needs the I²C Password presentation to be effective.

Table 371. ST25DVxxKC_SetRFSleep() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetRFSleep_Dyn()

`int32_t ST25DVxxKC_SetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Sets the RF Sleep dynamic configuration.

Table 372. ST25DVxxKC_SetRFSleep_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_WriteData()

`int32_t ST25DVxxKC_WriteData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )`

Writes N bytes of Data starting from the specified I2C Address.

Table 373. **ST25DVxxKC_WriteData() parameters**

in	*pObj*	pointer to the device structure object.
in	*pData*	Pointer on the data to be written.
in	*TarAddr*	I2C data memory address to be written.
in	*NbByte*	Number of bytes to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteEHMode()

```
int32_t ST25DVxxKC_WriteEHMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EH_MODE_STATUS_E EH_mode )
```

Sets the Energy harvesting mode.

Needs the I²C Password presentation to be effective.

Table 374. **ST25DVxxKC_WriteEHMode() parameters**

in	*pObj*	pointer to the device structure object.
in	*EH_mode*	ST25DVxxKC_EH_MODE_STATUS value for the Energy harvesting mode to be set.

Returns

int32_t enum status.

ST25DVxxKC_WriteEndZonex()

```
int32_t ST25DVxxKC_WriteEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, const uint8_t EndZ )
```

Sets the end address of an area.

Needs the I²C Password presentation to be effective.

Note: The ST25DVxxKC answers a NACK when setting the EndZone2 & EndZone3 to same value than repectively EndZone1 & EndZone2.

These NACKs are ok.

Table 375. **ST25DVxxKC_WriteEndZonex() parameters**

in	*pObj*	pointer to the device structure object.
in	*EndZone*	ST25DVxxKC_END_ZONE value corresponding to an area.
in	*EndZ*	End zone value to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteI2CPassword()

```
int32_t ST25DVxxKC_WriteI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord )
```

Writes a new I²C password.

Needs the I²C Password presentation to be effective.

Table 376. **ST25DVxxKC_WriteI2CPassword() parameters**

in	*pObj*	pointer to the device structure object.

in	*PassWord*	New I2C PassWord value on 32bits.

Returns

int32_t enum status.

ST25DVxxKC_WriteI2CProtectZonex()

```
int32_t ST25DVxxKC_WriteI2CProtectZonex (const ST25DVxxKC_Object_t *const pObj,
const ST25DVxxKC_PROTECTION_ZONE_E Zone, const ST25DVxxKC_PROTECTION_CONF_E
ReadWriteProtection )
```

Sets the I²C write-protected state to an EEPROM Area. Needs the I2C Password presentation to be effective.

Table 377. ST25DVxxKC_WriteI2CProtectZonex() parameters

in	*pObj*	pointer to the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value coresponding to the area to protect.
in	*ReadWriteProtection*	ST25DVxxKC_PROTECTION_CONF value corresponding to the protection to be set.

Returns

int32_t enum status.

ST25DVxxKC_WriteITPulse()

```
int32_t ST25DVxxKC_WriteITPulse (const ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_PULSE_DURATION_E ITtime )
```

Configures the ST25DVxxKC ITtime duration for the GPO pulse.

Needs the I²C Password presentation to be effective.

Table 378. ST25DVxxKC_WriteITPulse() parameters

in	*pObj*	pointer to the device structure object.
in	*ITtime*	Coefficient for the Pulse duration to be written (Pulse duration = 302,06 us - ITtime ∗ 512 / fc)

Table 379. ST25DVxxKC_WriteITPulse() return values

int32←,_t	enum status.

ST25DVxxKC_WriteLockCCFile()

```
int32_t ST25DVxxKC_WriteLockCCFile (const ST25DVxxKC_Object_t *const pObj,
const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile, const ST25DVxxKC_LOCK_STATUS_E
LockCCFile )
```

Locks the CCile to prevent any RF write access. Needs the I²C Password presentation to be effective.

Table 380. ST25DVxxKC_WriteLockCCFile() parameters

in	*pObj*	pointer to the device structure object.
in	*NbBlockCCFile*	ST25DVxxKC_CCFILE_BLOCK value corresponding to the number of blocks to be locked.
in	*LockCCFile*	ST25DVxxKC_LOCK_CCFILE value corresponding to the lock state to apply on the CCFile.

Returns

int32_t enum status.

ST25DVxxKC_WriteLockCFG()

`int32_t ST25DVxxKC_WriteLockCFG (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_LOCK_STATUS_E LockCfg ) Lock/Unlock the Cfg registers, to prevent any RF write access.`

Needs the I²C Password presentation to be effective.

Table 381. ST25DVxxKC_WriteLockCFG() parameters

in	*pObj*	pointer to the device structure object.
in	*LockCfg*	ST25DVxxKC_LOCK_STATUS value corresponding to the lock state to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteMailboxData()

`int32_t ST25DVxxKC_WriteMailboxData ( const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t NbByte )`

Writes N bytes of data in the Mailbox, starting from first Mailbox Address.

Table 382. ST25DVxxKC_WriteMailboxData() parameters

in	*pObj*	pointer to the device structure object.
in	*pData*	Pointer to the buffer containing the data to be written.
in	*NbByte*	Number of bytes to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteMailboxRegister()

`int32_t ST25DVxxKC_WriteMailboxRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )`

Writes N bytes to the specified mailbox register.

Table 383. ST25DVxxKC_WriteMailboxRegister() parameters

in	*pObj*	pointer to the device structure object.
in	*pData*	Pointer on the data to be written.
in	*TarAddr*	I2C register address to be written.
in	*NbByte*	Number of bytes to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteMBMode()

`int32_t ST25DVxxKC_WriteMBMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EN_STATUS_E MB_mode )`

Sets the Mailbox mode.

Needs the I²C Password presentation to be effective.

Table 384. **ST25DVxxKC_WriteMBMode() parameters**

in	*pObj*	pointer to the device structure object.
out	*MB_mode*	ST25DVxxKC_EN_STATUS value corresponding to the Mailbox mode to be set.

Returns

int32_t enum status.

ST25DVxxKC_WriteMBWDG()

```
int32_t ST25DVxxKC_WriteMBWDG (const ST25DVxxKC_Object_t *const pObj, const uint8_t WdgDelay )
```

Writes the Mailbox watchdog coefficient delay.

Needs the I²C Password presentation to be effective.

Table 385. **ST25DVxxKC_WriteMBWDG() parameters**

in	*pObj*	pointer to the device structure object.
in	*WdgDelay*	Watchdog duration coefficient to be written (Watch dog duration = MB_WDG∗30 ms +/- 6%).

Returns

int32_t enum status.

ST25DVxxKC_WriteReg()

```
int32_t ST25DVxxKC_WriteReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, const uint8_t *const Data, const uint16_t len )
```

Write register to component.

Table 386. **ST25DVxxKC_WriteReg() parameters**

in	*ctx*	structure containing context driver
in	*Reg*	register to write
in	*Data*	data pointer to write to register
out	*len*	length of data

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_WriteRegister()

```
int32_t ST25DVxxKC_WriteRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )
```

Writes N bytes to the specified register.

Needs the I²C Password presentation to be effective.

Table 387. **ST25DVxxKC_WriteRegister() parameters**

in	*pObj*	pointer to the device structure object.	
in	*pData*	Pointer on the data to be written.	
Geneiranted	**by** *Ta* **D** *r* **o** *A* **xy** *d* **g** *d* **en** *r*	I2C register address to written.	
in	*NbByte*	Number of bytes to be written.	

Returns

int32_t enum status.

ST25DVxxKC_WriteRFMngt()

```
int32_t ST25DVxxKC_WriteRFMngt (const ST25DVxxKC_Object_t *const pObj, const
uint8_t Rfmngt )
```

Sets the RF Management configuration.

Needs the I²C Password presentation to be effective.

Table 388. ST25DVxxKC_WriteRFMngt() parameters

in	*pObj*	pointer to the device structure object.
in	*Rfmngt*	Value of the RF Management configuration to be written.

Returns

int32_t enum status.

ST25DVxxKC_WriteRFMngt_Dyn()

```
int32_t ST25DVxxKC_WriteRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, const
uint8_t RF_Mngt )
```

Writes a value to the RF Management dynamic register.

Table 389. ST25DVxxKC_WriteRFMngt_Dyn() parameters

in	*pObj*	pointer to the device structure object.
in	*RF_Mngt*	Value to be written to the RF Management dynamic register.

Returns

int32_t enum status.

ST25DVxxKC_WriteRFZxSS()

```
int32_t ST25DVxxKC_WriteRFZxSS (const ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_PROTECTION_ZONE_E Zone,const ST25DVxxKC_RF_PROT_ZONE_t RfProtZone )
```

Writes the RF Zone Security Status (defining the allowed RF accesses).

Needs the I²C Password presentation to be effective.

Table 390. ST25DVxxKC_WriteRFZxSS() parameters

in	*pObj*	pointer to the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value corresponding to the area on which to set the RF protection.
in	*RfProtZone*	Pointer on a ST25DVxxKC_RF_PROT_ZONE value defininf the protection to be set on the area.

Returns

int32_t enum status.

6.3 Variable documentation

St25Dvxxkc_Drv

```
ST25DVxxKC_Drv_t St25Dvxxkc_Drv
```

Standard NFC tag driver API for the ST25DVxxKC.

Provides a generic way to access the ST25DVxxKC implementation of the NFC tag standard driver functions.

7 Middlewares

Modules

- NUCLEO_WIFI_API
  Wi-Fi_interface API.

7.1 NUCLEO_WIFI_API

Wi-Fi_interface API.

Modules

- NUCLEO_WIFI_API_Private_Macros
- NUCLEO_WIFI_API_Private_Variables

Detailed description

Wi-Fi_interface API.

7.1.1 NUCLEO_WIFI_API_Private_Variables

Data structures

- struct wifi_security
- struct wifi_scan
- struct wifi_config

Functions

- `void ind_wifi_ap_client_joined (uint8_t *client_mac_address)`
  This function is the callback for the wifi middleware, called when a client has joined.

7.1.2 Function documentation

ind_wifi_ap_client_joined()

`void ind_wifi_ap_client_joined (uint8_t * mac)`

This function is the callback for the Wifi Middleware, called when a client has joined.

Table 391. ind_wifi_ap_client_joined() parameters

mac	: string used to return the MAC address of connected device.

8 STM32L4xx_HAL_Driver

Modules

- HAL_MSP
  HAL MSP module.

8.1 Board support package

Container module for the BSP modules.

Modules

- ST25 Discovery NFCTAG board support package
  This module provides high-level functions to access the ST25DV-I2C NFC dynamic tag.
- ST25DVxxKC driver
  This module implements the functions to drive the ST25DVxxKC NFC dynamic tag.
- ST25DVXXKC

8.1.1 Detailed description

Container module for the BSP modules.

The board support package software (BSP) is defined by the Cube methodology as the abstraction layer for the board specific features.

It implements all the functions required to access:

- the components on the board.
- the MCU peripherals requiring a board specific configuration.

8.2 ST25DVXXKC

Data structures

- struct ST25DVxxKC_EH_CTRL_t
  ST25DVxxKC EH Ctrl structure definition.
- struct ST25DVxxKC_GPO_t
  ST25DVxxKC GPO structure definition.
- struct ST25DVxxKC_RF_MNGT_t
  ST25DVxxKC RF Management structure definition.
- struct ST25DVxxKC_RF_PROT_ZONE_t
  ST25DVxxKC RF Area protection structure definition.
- struct ST25DVxxKC_I2C_PROT_ZONE_t
  ST25DVxxKC I2C Area protection structure definition.
- struct ST25DVxxKC_MB_CTRL_DYN_STATUS_t
  ST25DVxxKC MB_CTRL_DYN register structure definition.
- struct ST25DVxxKC_LOCK_CCFILE_t
  ST25DVxxKC Lock CCFile structure definition.
- struct ST25DVxxKC_MEM_SIZE_t
  ST25DVxxKC Memory size structure definition.
- struct ST25DVxxKC_UID_t
  ST25DVxxKC UID information structure definition.
- struct ST25DVxxKC_PASSWD_t
  ST25DVxxKC Password structure definition.

- struct ST25DVxxKC_IO_t
  ST25DVxxKC IO API structure definition.
- struct ST25DVxxKC_Object_t
  ST25DVxxKC device structure definition.
- struct ST25DVxxKC_Ctx_t
  ST25DVxxKC context structure.

Macros

- `#define I_AM_ST25DV04KC 0x50`
  ST25DVxxKC 4Kbits ICref.
- `#define I_AM_ST25DV64KC 0x51`
  ST25DVxxKC 16/64Kbits ICref.
- `#define ST25DVXXKC_AM_I_OPEN_DRAIN(x) (((x) == 0x50) || ((x) == 0x51))`
  Check ST25DVxxKC Open Drain version.
- `#define ST25DVXXKC_AM_I_CMOS (x) (((x) == 0x52) || ((x) == 0x53))`
  Check ST25DVxxKC CMOS version.
- `#define ST25DVXXKC_ADDR_DATA_I2C 0xA6`
  I²C address to be used for ST25DVxxKC Data accesses.
- `#define ST25DVXXKC_ADDR_SYST_I2C 0xAE`
  I²C address to be used for ST25DVxxKC System accesses.
- `#define ST25DVXXKC_WRITE_TIMEOUT 320`
  I²C Time out (ms), min value : (Max write bytes) / (Internal page write) ∗ tw (256/4)∗5.
- `#define ST25DVXXKC_MAX_WRITE_BYTE 256`
  Size of the ST25DVxxKC write buffer.
- `#define ST25DVXXKC_MAX_MAILBOX_LENGTH 256`
  Size of the ST25DVxxKC Mailbox memory
- `#define ST25DVXXKC_IS_DYNAMIC_REGISTER 0x2000`
  Offset of ST25DVxxKC dynamic registers
- `#define ST25DVXXKC_GPO1_REG 0x0000`
  ST25DVxxKC GPO1 register address
- `#define ST25DVXXKC_GPO2_REG 0x0001`
  ST25DVxxKC GPO2 register address.
- `#define ST25DVXXKC_EH_MODE_REG 0x0002`
  ST25DVxxKC Energy Harvesting register address.
- `#define ST25DVXXKC_RF_MNGT_REG 0x0003`
  ST25DVxxKC RF management register address.
- `#define ST25DVXXKC_RFA1SS_REG 0x0004`
  ST25DVxxKC Area 1 security register address.
- `#define ST25DVXXKC_ENDA1_REG 0x0005`
  ST25DVxxKC Area 1 end address register address.
- `#define ST25DVXXKC_RFA2SS_REG 0x0006`
  ST25DVxxKC Area 2 security register address.
- `#define ST25DVXXKC_ENDA2_REG 0x0007`
  ST25DVxxKC Area 2 end address register address.
- `#define ST25DVXXKC_RFA3SS_REG 0x0008`
  ST25DVxxKC Area 3 security register address.
- `#define ST25DVXXKC_ENDA3_REG 0x0009`
  ST25DVxxKC Area 3 end address register address.

- `#define ST25DVXXKC_RFA4SS_REG 0x000A`
  ST25DVxxKC Area 4 security register address.
- `#define ST25DVXXKC_I2CSS_REG 0x000B`
  ST25DVxxKC I2C security register address.
- `#define ST25DVXXKC_LOCKCCFILE_REG 0x000C`
  ST25DVxxKC Capability Container lock register address.
- `#define ST25DVXXKC_MB_REG 0x000D`
  ST25DVxxKC Mailbox mode register address.
- `#define ST25DVXXKC_I2CCFG_REG 0x000E`
  ST25DVxxKC I2C configuration register address.
- `#define ST25DVXXKC_LOCKCFG_REG 0x000F`
  ST25DVxxKC Configuration lock register address.
- `#define ST25DVXXKC_LOCKDSFID_REG 0x0010`
  ST25DVxxKC DSFID lock register address.
- `#define ST25DVXXKC_LOCKAFI_REG 0x0011`
  ST25DVxxKC AFI lock register address.
- `#define ST25DVXXKC_DSFID_REG0x0012`
  ST25DVxxKC DSFID register address.
- `#define ST25DVXXKC_AFI_REG 0x0013`
  ST25DVxxKC AFI register address.
- `#define ST25DVXXKC_MEM_SIZE_LSB_REG 0x0014`
  ST25DVxxKC Memory size register address.
- `#define ST25DVXXKC_MEM_SIZE_MSB_REG 0x0015`
  ST25DVxxKC Memory size register address.
- `#define ST25DVXXKC_BLK_SIZE_REG 0x0016`
  ST25DVxxKC Block size register address.
- `#define ST25DVXXKC_ICREF_REG 0x0017`
  ST25DVxxKC ICref register address.
- `#define ST25DVXXKC_UID_REG 0x0018`
  ST25DVxxKC UID register address.
- `#define ST25DVXXKC_ICREV_REG 0x0020`
  ST25DVxxKC IC revision register address.
- `#define ST25DVXXKC_I2CPASSWD_REG 0x0900`
  ST25DVxxKC I2C password register address.
- `#define ST25DVXXKC_GPO_DYN_REG 0x2000`
  ST25DVxxKC GPO dynamic register address.
- `#define ST25DVXXKC_EH_CTRL_DYN_REG 0x2002`
  ST25DVxxKC Energy Harvesting control dynamic register address.
- `#define ST25DVXXKC_RF_MNGT_DYN_REG 0x2003`
  ST25DVxxKC RF management dynamic register address.
- `#define ST25DVXXKC_I2C_SSO_DYN_REG 0x2004`
  ST25DVxxKC I2C secure session opened dynamic register address.
- `#define ST25DVXXKC_ITSTS_DYN_REG 0x2005`
  ST25DVxxKC Interrupt status dynamic register address.
- `#define ST25DVXXKC_MB_CTRL_DYN_REG 0x2006`
  ST25DVxxKC Mailbox control dynamic register address.
- `#define ST25DVXXKC_MBLEN_DYN_REG 0x2007`
  ST25DVxxKC Mailbox message length dynamic register address.

- `#define ST25DVXXKC_MAILBOX_RAM_REG 0x2008`
  ST25DVxxKC Mailbox buffer address.
- `#define NFCTAG_OK (0)`
  NFCTAG status enumerator definition.

Typedefs

- `typedef int32_t(* ST25DVxxKC_WriteReg_Func) (const void *const, const uint16_t, const uint8_t *const, const uint16_t)`
  ST25DVxxKC Write register function typedef definition.
- `typedef int32_t(* ST25DVxxKC_ReadReg_Func) (const void *const, const uint16_t, uint8_t *const, const uint16_t)`
  ST25DVxxKC Read register function typedef definition.

Enumerations

- enum ST25DVxxKC_EN_STATUS_E
  ST25DVxxKC Enable Disable enumerator definition.
- enum ST25DVxxKC_EH_MODE_STATUS_E
  ST25DVxxKC Energy Harvesting mode enumerator definition.
- enum ST25DVxxKC_FIELD_STATUS_E
  ST25DVxxKC FIELD status enumerator definition.
- enum ST25DVxxKC_VCC_STATUS_E
  ST25DVxxKC VCC status enumerator definition.
- enum ST25DVxxKC_PROTECTION_CONF_E
  ST25DVxxKC protection status enumerator definition.
- enum ST25DVxxKC_PROTECTION_ZONE_E
  ST25DVxxKC area protection enumerator definition.
- enum ST25DVxxKC_PASSWD_PROT_STATUS_E
  ST25DVxxKC password protection status enumerator definition.
- enum ST25DVxxKC_LOCK_STATUS_E
  ST25DVxxKC lock status enumerator definition.
- enum ST25DVxxKC_CCFILE_BLOCK_E
  ST25DVxxKC Number of Blocks for the CCFile enumerator definition.
- enum ST25DVxxKC_I2CSSO_STATUS_E
  ST25DVxxKC session status enumerator definition.
- enum ST25DVxxKC_END_ZONE_E
  ST25DVxxKC area end address enumerator definition.
- enum ST25DVxxKC_PULSE_DURATION_E
  ST25DVxxKC IT pulse duration enumerator definition.
- enum ST25DVxxKC_CURRENT_MSG_E
  ST25DVxxKC Mailbox Current Message enumerator definition.

Functions

- `int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from Registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified register.

- `int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_IO_t *const pIO)`

  Register Component Bus IO operations.
- `int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo)`

  Reads the ST25DVxxKC Memory Size.
- `int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRev)`

  Reads the ST25DVxxKC IC Revision.
- `int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_PULSE_DURATION_E *const pITtime)`

  Reads the ST25DVxxKC ITtime duration for the GPO pulses.
- `int32_t ST25DVxxKC_WriteITPulse (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PULSE_DURATION_E ITtime)`

  Configures the ST25DVxxKC ITtime duration for the GPO pulse.
- `int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t *const p←, Uid)`

  Reads the ST25DVxxKC UID.
- `int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid)`

  Reads the ST25DVxxKC DSFID.
- `int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid)`

  Reads the ST25DVxxKC DSFID RF Lock state.
- `int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi)`

  Reads the ST25DVxxKC AFI.
- `int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi)`

  Reads the AFI RF Lock state.
- `int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone)`

  Reads the I²C Protected Area state.
- `int32_t ST25DVxxKC_WriteI2CProtectZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ Zone, const ST25DVxxKC_PROTECTION_CONF_E ReadWriteProtection)`

  Sets the I²C write-protected state to an EEPROM Area.
- `int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile)`

  Reads the CCile protection state.
- `int32_t ST25DVxxKC_WriteLockCCFile (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile, const ST25DVxxKC_LOCK_STATUS_E LockCCFile)`

  Locks the CCfile to prevent any RF write access.
- `int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockCfg)`

  Reads the Cfg registers protection.
- `int32_t ST25DVxxKC_WriteLockCFG (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_LOCK_STATUS_E LockCfg)`

  Lock/Unlock the Cfg registers, to prevent any RF write access.

- `int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Presents I²C password, to authorize the I²C writes to protected areas.
- `int32_t ST25DVxxKC_WriteI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Writes a new I²C password.
- `int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, ST25DVxxKC_RF_PROT_ZONE_t *const pRfprotZone)`

  Reads the RF Zone Security Status (defining the allowed RF accesses).
- `int32_t ST25DVxxKC_WriteRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, const ST25DVxxKC_RF_PROT_ZONE_t RfProtZone)`

  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, uint8_t *pEndZ)`

  Reads the value of the an area end address.
- `int32_t ST25DVxxKC_WriteEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, const uint8_t EndZ)`

  Sets the end address of an area.
- `int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj)`

  Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).
- `int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, uint16_t Zone1Length, uint16_t Zone2Length, uint16_t Zone3Length, uint16_t Zone4Length)`

  Creates user areas with defined lengths.
- `int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode)`

  Reads the Energy harvesting mode.
- `int32_t ST25DVxxKC_WriteEHMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EH_MODE_STATUS_E EH_mode)`

  Sets the Energy harvesting mode.
- `int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`

  Reads the RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt (const ST25DVxxKC_Object_t *const pObj, const uint8_t Rfmngt)`

  Sets the RF Management configuration.
- `int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`

  Reads the RFDisable register information.
- `int32_t ST25DVxxKC_SetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Disable configuration.
- `int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Disable configuration.
- `int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`

  Reads the RFSleep register information.
- `int32_t ST25DVxxKC_SetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Sleep configuration.

- `int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMB_mode)`
  Reads the Mailbox mode.
- `int32_t ST25DVxxKC_WriteMBMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EN_STATUS_E MB_mode)`
  Sets the Mailbox mode.
- `int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pWdgDelay)`
  Reads the Mailbox watchdog duration coefficient.
- `int32_t ST25DVxxKC_WriteMBWDG (const ST25DVxxKC_Object_t *const pObj, const uint8_t WdgDelay)`
  Writes the Mailbox watchdog coefficient delay.
- `int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- `int32_t ST25DVxxKC_WriteMailboxData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t NbByte)`
  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.
- `int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, Data, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteMailboxRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified mailbox register.
- `int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession)`
  Reads the status of the security session open register.
- `int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, ITStatus)`
  Reads the IT status register from the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *GPOConfig)`
  Read value of dynamic GPO register configuration.
- `int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pGPO_en)`
  Get dynamique GPO enable status.
- `int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Set dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Reset dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)`
  Read value of dynamic EH Ctrl register configuration.
- `int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEH_Val)`
  Reads the Energy Harvesting dynamic status.
- `int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Dynamically sets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Dynamically unsets the Energy Harvesting mode.

- `int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEHON)`
  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_FIELD_STATUS_E *const pRF_Field)`
  Checks if RF Field is present in front of the ST25DVxxKC.
- `int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC)`
  Check if V_{CC} is supplying the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`
  Read value of dynamic RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, const uint8_t RF_←, Mngt)`
  Writes a value to the RF Management dynamic register.
- `int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`
  Reads the RFDisable dynamic register information.
- `int32_t ST25DVxxKC_SetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t ST25DVxxKC_SetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MB_CTRL_DYN_STATUS_*const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, MBLength)`
  Reads the Mailbox message length dynamic register.
- `int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read IC Ref register.
- `int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA1 register.
- `int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA1 register.

- `int32_t ST25DVxxKC_GetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA2 register.
- `int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA2 register.
- `int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA3 register.
- `int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA3 register.
- `int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read DSFID register.
- `int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read AFI register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read MEM_SIZE_MSB register.
- `int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read BLK_SIZE register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read MEM_SIZE_LSB register.
- `int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ICREV register.
- `int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read UID register.
- `int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CPASSWD register.
- `int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CPASSWD register.
- `int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKDSFI register.
- `int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKAFI register.
- `int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read MB_MODE_RW register.
- `int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write MB_MODE_RW register.

- `int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MBLEN_DYN_MBLEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_STRESERVED register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_CURRENTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFINTERRUPT register.

- `int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`
  Write GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO1_ALL register.
- `int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO1_ALL register.
- `int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO2_RFOFF register.
- `int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO2_RFOFF register.

- `int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO2_ITTIME register.
- `int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO2_ITTIME register.
- `int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO2_ALL register.
- `int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO2_ALL register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITTIME_DELAY register.
- `int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ITTIME_DELAY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`
  Read ITSTS_DYN_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`
  Read ITSTS_DYN_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`
  Read ITSTS_DYN_FIELDFALLING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_FIELDRISING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_RFGETMSG register.

- `int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_RFWRITE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ITSTS_DYN_ALL register.
- `int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_MODE register.
- `int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write EH_MODE register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`
  Write EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_CTRL_DYN_EH_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_CTRL_DYN_FIELD_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_CTRL_DYN_VCC_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read EH_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RF_MNGT_RFSLEEP register.
- `int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_DYN_RFDIS register.

- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`
  Write RF_MNGT_DYN_RFDIS register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA1SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA1SS_ALL(const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA1SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA2SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA2SS_ALL register.

- `int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA3SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA3SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read RFA4SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write RFA4SS_ALL register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CSS_PZ3 register.

- `int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CSS_ALL register.
- `int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write I2CSS_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKCFG_B0 register.
- `int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCFG_B0 register.
- `int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2C_SSO_DYN_I2CSSO register.

Variables

- ST25DVxxKC_Drv_t St25Dvxxkc_Drv
  Standard NFC tag driver API for the ST25DVxxKC.
  struct ST25DVxxKC_EH_CTRL_t
  ST25DVxxKC EH Ctrl structure definition.
  struct ST25DVxxKC_GPO_t
  ST25DVxxKC GPO structure definition.
  struct ST25DVxxKC_RF_MNGT_t
  ST25DVxxKC RF Management structure definition.
  struct ST25DVxxKC_RF_PROT_ZONE_t
  ST25DVxxKC RF Area protection structure definition.
  struct ST25DVxxKC_I2C_PROT_ZONE_t
  ST25DVxxKC I2C Area protection structure definition.
  struct ST25DVxxKC_MB_CTRL_DYN_STATUS_t
  ST25DVxxKC MB_CTRL_DYN register structure definition.
  struct ST25DVxxKC_LOCK_CCFILE_t
  ST25DVxxKC Lock CCFile structure definition.
  struct ST25DVxxKC_MEM_SIZE_t
  ST25DVxxKC Memory size structure definition.
  struct ST25DVxxKC_UID_t
  ST25DVxxKC UID information structure definition.
  struct ST25DVxxKC_PASSWD_t
  ST25DVxxKC Password structure definition.
  struct ST25DVxxKC_IO_t
  ST25DVxxKC IO API structure definition.
  struct ST25DVxxKC_Object_t
  ST25DVxxKC device structure definition.
  struct ST25DVxxKC_Ctx_t
  ST25DVxxKC context structure.

8.2.1 Function documentation

ST25DVxxKC_CreateUserZone()

```
int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, const uint16_t Zone1Length, const uint16_t Zone2Length, const uint16_t Zone3Length, const uint16_t Zone4Length)
```

Creates user areas with defined lengths.

Needs the I²C Password presentation to be effective.

Table 392. ST25DVxxKC_CreateUserZone() parameters

in	*pObj*	pointer to the device structure object.
in	*Zone1Length*	Length of area1 in bytes (32 to 8192, 0x20 to 0x2000)
in	*Zone2Length*	Length of area2 in bytes (0 to 8128, 0x00 to 0x1FC0)
in	*Zone3Length*	Length of area3 in bytes (0 to 8064, 0x00 to 0x1F80)
in	*Zone4Length*	Length of area4 in bytes (0 to 8000, 0x00 to 0x1F40)

Returns

int32_t enum status.

ST25DVxxKC_GetAFI()

```
int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read AFI register.

Table 393. ST25DVxxKC_GetAFI() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetBLK_SIZE()

```
int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read BLK_SIZE register.

Table 394. ST25DVxxKC_GetBLK_SIZE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetDSFID()

```
int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read DSFID register.

Table 395. ST25DVxxKC_GetDSFID() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_ALL()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read EH_CTRL_DYN_ALL register.

Table 396. ST25DVxxKC_GetEH_CTRL_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_EH_EN()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read EH_CTRL_DYN_EH_EN register.

Table 397. ST25DVxxKC_GetEH_CTRL_DYN_EH_EN() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_EH_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value) Read EH_CTRL_DYN_EH_ON register.
```

Table 398. ST25DVxxKC_GetEH_CTRL_DYN_EH_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON ( const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_CTRL_DYN_FIELD_ON register.

Table 399. ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON()

```
int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value ) Read EH_CTRL_DYN_VCC_ON register.
```

Table 400. ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEH_MODE()

```
int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read EH_MODE register.

Table 401. ST25DVxxKC_GetEH_MODE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetEHENMode_Dyn()

```
int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEH_Val )
```

Reads the Energy Harvesting dynamic status.

Table 402. ST25DVxxKC_GetEHENMode_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pEH_Val*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the Energy Harvesting
		dynamic status. **Generated by Doxygen**

Returns

int32_t enum status.

ST25DVxxKC_GetEHON_Dyn()

```
int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEHON ) Reads the EH_ON status from the EH_CTRL_DYN register.
```

Table 403. ST25DVxxKC_GetEHON_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pEHON*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the EHON status.

Returns

int32_t enum status.

ST25DVxxKC_GetENDA1()

```
int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ENDA1 register.

Table 404. ST25DVxxKC_GetENDA1() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetENDA2()

```
int32_t ST25DVxxKC_GetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ENDA2 register.

Table 405. ST25DVxxKC_GetENDA2() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetENDA3()

```
int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read ENDA3 register.

Table 406. ST25DVxxKC_GetENDA3() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_ALL()

```
int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_ALL register.

Table 407. ST25DVxxKC_GetGPO1_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_ENABLE()

```
int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read GPO1_ENABLE register.

Table 408. ST25DVxxKC_GetGPO1_ENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_FIELDCHANGE()

```
int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_FIELDCHANGE register.

Table 409. ST25DVxxKC_GetGPO1_FIELDCHANGE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFACTIVITY()

```
int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFACTIVITY register.

Table 410. ST25DVxxKC_GetGPO1_RFACTIVITY() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFGETMSG()

```
int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFGETMSG register.

Table 411. ST25DVxxKC_GetGPO1_RFGETMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFINTERRUPT()

```
int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFINTERRUPT register.

Table 412. ST25DVxxKC_GetGPO1_RFINTERRUPT() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFPUTMSG()

```
int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFPUTMSG register.

Table 413. ST25DVxxKC_GetGPO1_RFPUTMSG() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFUSERSTATE()

```
int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFUSERSTATE register.

Table 414. ST25DVxxKC_GetGPO1_RFUSERSTATE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO1_RFWRITE()

```
int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO1_RFWRITE register.

Table 415. ST25DVxxKC_GetGPO1_RFWRITE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_ALL()

```
int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read GPO2_ALL register.

Table 416. ST25DVxxKC_GetGPO2_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_I2CWRITEENABLE()

```
int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value)
```

Read GPO2_I2CWRITEENABLE register.

Table 417. ST25DVxxKC_GetGPO2_I2CWRITEENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_ITTIME()

```
int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read GPO2_ITTIME register.

Table 418. ST25DVxxKC_GetGPO2_ITTIME() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO2_RFOFF()

```
int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read GPO2_RFOFF register.

Table 419. ST25DVxxKC_GetGPO2_RFOFF() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_DYN_ALL()

```
int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read GPO_DYN_ALL register.

Table 420. ST25DVxxKC_GetGPO_DYN_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_DYN_ENABLE()

```
int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read GPO_DYN_ENABLE register.

Table 421. ST25DVxxKC_GetGPO_DYN_ENABLE() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetGPO_en_Dyn()

```
int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pGPO_en )
```

Get dynamique GPO enable status.

Table 422. ST25DVxxKC_GetGPO_en_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pGPO_en*	ST25DVxxKC_EN_STATUS pointer of the GPO enable status to store.

Table 423. ST25DVxxKC_GetGPO_en_Dyn() return values

NFCTAG	enum status

ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO()

```
int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2C_SSO_DYN_I2CSSO register.

Table 424. ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CPASSWD()

```
int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CPASSWD register.

Table 425. ST25DVxxKC_GetI2CPASSWD() parameters

in	*ctx*	structure containing context driver

out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_ALL()

```
int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_ALL register.

Table 426. ST25DVxxKC_GetI2CSS_ALL() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ1()

```
int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_PZ1 register.

Table 427. ST25DVxxKC_GetI2CSS_PZ1() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ2()

```
int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_PZ2 register.

Table 428. ST25DVxxKC_GetI2CSS_PZ2() parameters

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ3()

```
int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read I2CSS_PZ3 register.

Table 429. **ST25DVxxKC_GetI2CSS_PZ3() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetI2CSS_PZ4()

```
int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read I2CSS_PZ4 register.

Table 430. **ST25DVxxKC_GetI2CSS_PZ4() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetICREF()

```
int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read IC Ref register.

Table 431. **ST25DVxxKC_GetICREF() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetICREV()

```
int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ICREV register.

Table 432. **ST25DVxxKC_GetICREV() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_ALL()

```
int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_ALL register.

Table 433. **ST25DVxxKC_GetITSTS_DYN_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_FIELDFALLING()

```
int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value)
```

Read ITSTS_DYN_FIELDFALLING register.

Table 434. **ST25DVxxKC_GetITSTS_DYN_FIELDFALLING() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_FIELDRISING()

```
int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const
ctx,uint8_t *const value)
```

Read ITSTS_DYN_FIELDRISING register.

Table 435. **ST25DVxxKC_GetITSTS_DYN_FIELDRISING() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFACTIVITY()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read ITSTS_DYN_RFACTIVITY register.

Table 436. **ST25DVxxKC_GetITSTS_DYN_RFACTIVITY() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFGETMSG()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read ITSTS_DYN_RFGETMSG register.

Table 437. **ST25DVxxKC_GetITSTS_DYN_RFGETMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx,uint8_t *const value )
```

Read ITSTS_DYN_RFINTERRUPT register.

Table 438. **ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFPUTMSG()

int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t ∗const *ctx,* uint8_t ∗const *value*)

Read ITSTS_DYN_RFPUTMSG register.

Table 439. **ST25DVxxKC_GetITSTS_DYN_RFPUTMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_RFUSERSTATE register.

Table 440. **ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITSTS_DYN_RFWRITE()

```
int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITSTS_DYN_RFWRITE register.

Table 441. **ST25DVxxKC_GetITSTS_DYN_RFWRITE() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetITTIME_DELAY()

```
int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read ITTIME_DELAY register.

Table 442. **ST25DVxxKC_GetITTIME_DELAY() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKAFI()

```
int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKAFI register.

Table 443. **ST25DVxxKC_GetLOCKAFI() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_ALL()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_ALL register.

Table 444. **ST25DVxxKC_GetLOCKCCFILE_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_BLCK0()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_BLCK0 register.

Table 445. **ST25DVxxKC_GetLOCKCCFILE_BLCK0() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCCFILE_BLCK1()

```
int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCCFILE_BLCK1 register.

Table 446. **ST25DVxxKC_GetLOCKCCFILE_BLCK1() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKCFG_B0()

```
int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKCFG_B0 register.

Table 447. **ST25DVxxKC_GetLOCKCFG_B0() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetLOCKDSFID()

```
int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read LOCKDSFI register.

Table 448. **ST25DVxxKC_GetLOCKDSFID() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_ALL()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read MB_CTRL_DYN_ALL register.

Table 449. **ST25DVxxKC_GetMB_CTRL_DYN_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value)
```

Read MB_CTRL_DYN_CURRENTMSG register.

Table 450. **ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG()

int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t ∗const *ctx,* uint8_t ∗const *value*)

Read MB_CTRL_DYN_HOSTMISSMSG register.

Table 451. **ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const
ctx,uint8_t *const value )
```

Read MB_CTRL_DYN_HOSTPUTMSG register.

Table 452. **ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_MBEN()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read MB_CTRL_DYN_MBEN register.

Table 453. **ST25DVxxKC_GetMB_CTRL_DYN_MBEN() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG ( const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value)
```

Read MB_CTRL_DYN_RFMISSMSG register.

Table 454. **ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG ( const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read MB_CTRL_DYN_RFPUTMSG register.

Table 455. **ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED()

```
int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read MB_CTRL_DYN_STRESERVED register.

Table 456. **ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_MODE_RW()

```
int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read MB_MODE_RW register.

Table 457. **ST25DVxxKC_GetMB_MODE_RW() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMB_WDG_DELAY()

```
int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx,uint8_t
*const value )
```

Read MB_WDG_DELAY register.

Table 458. **ST25DVxxKC_GetMB_WDG_DELAY() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMBEN_Dyn()

```
int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const
pObj,ST25DVxxKC_EN_STATUS_E *const pMBEN )
```

Reads the Mailbox Enable dynamic configuration.

Table 459. **ST25DVxxKC_GetMBEN_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pMBEN*	pointer on ST25DVxxKC_EN_STATUS_E values used to return the Mailbox enable state.

Returns

int32_t enum status.

ST25DVxxKC_GetMBLEN_DYN_MBLEN()

```
int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read MBLEN_DYN_MBLEN register.

Table 460. **ST25DVxxKC_GetMBLEN_DYN_MBLEN() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMEM_SIZE_LSB()

```
int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value)
```

Read MEM_SIZE_LSB register.

Table 461. **ST25DVxxKC_GetMEM_SIZE_LSB() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetMEM_SIZE_MSB()

```
int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read MEM_SIZE_MSB register.

Table 462. **ST25DVxxKC_GetMEM_SIZE_MSB() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_ALL()

```
int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read RF_MNGT_ALL register.

Table 463. **ST25DVxxKC_GetRF_MNGT_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_ALL()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx,uint8_t
*const value )
```

Read RF_MNGT_DYN_ALL register.

Table 464. **ST25DVxxKC_GetRF_MNGT_DYN_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_RFDIS()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read RF_MNGT_DYN_RFDIS register.

Table 465. **ST25DVxxKC_GetRF_MNGT_DYN_RFDIS() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP()

```
int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx,
uint8_t *const value )
```

Read RF_MNGT_DYN_RFSLEEP register.

Table 466. **ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_RFDIS()

```
int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read RF_MNGT_RFDIS register.

Table 467. **ST25DVxxKC_GetRF_MNGT_RFDIS() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRF_MNGT_RFSLEEP()

```
int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t
*const value )
```

Read RF_MNGT_RFSLEEP register.

Table 468. **ST25DVxxKC_GetRF_MNGT_RFSLEEP() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_ALL()

```
int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const
value )
```

Read RFA1SS_ALL register.

Table 469. **ST25DVxxKC_GetRFA1SS_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA1SS_PWDCTRL register.

Table 470. **ST25DVxxKC_GetRFA1SS_PWDCTRL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA1SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA1SS_RWPROT register.

Table 471. **ST25DVxxKC_GetRFA1SS_RWPROT() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_ALL()

```
int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA2SS_ALL register.

Table 472. **ST25DVxxKC_GetRFA2SS_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA2SS_PWDCTRL register.

Table 473. **ST25DVxxKC_GetRFA2SS_PWDCTRL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA2SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA2SS_RWPROT register.

Table 474. **ST25DVxxKC_GetRFA2SS_RWPROT() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_ALL()

```
int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA3SS_ALL register.

Table 475. **ST25DVxxKC_GetRFA3SS_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA3SS_PWDCTRL register.

Table 476. **ST25DVxxKC_GetRFA3SS_PWDCTRL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA3SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA3SS_RWPROT register.

Table 477. **ST25DVxxKC_GetRFA3SS_RWPROT() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_ALL()

```
int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA4SS_ALL register.

Table 478. **ST25DVxxKC_GetRFA4SS_ALL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_PWDCTRL()

```
int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
```

Read RFA4SS_PWDCTRL register.

Table 479. **ST25DVxxKC_GetRFA4SS_PWDCTRL() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFA4SS_RWPROT()

```
int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read RFA4SS_RWPROT register.

Table 480. **ST25DVxxKC_GetRFA4SS_RWPROT() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetRFDisable()

```
int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable )
```

Reads the RFDisable register information.

Table 481. **ST25DVxxKC_GetRFDisable() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRFDisable*	Pointer on a ST25DVxxKC_EN_STATUS value corresponding to the RF Disable status.

Returns

int32_t enum status.

ST25DVxxKC_GetRFDisable_Dyn()

```
int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EN_STATUS_E *const pRFDisable )
```

Reads the RFDisable dynamic register information.

Table 482. **ST25DVxxKC_GetRFDisable_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRFDisable*	Pointer on a ST25DVxxKC_EN_STATUS value used to return the RF Disable state.

Returns

int32_t enum status.

ST25DVxxKC_GetRFField_Dyn()

```
int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_FIELD_STATUS_E *const pRF_Field )
```

Checks if RF Field is present in front of the ST25DVxxKC.

Table 483. **ST25DVxxKC_GetRFField_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRF_Field*	Pointer on a ST25DVxxKC_FIELD_STATUS value used to return the field presence.

Returns

int32_t enum status.

ST25DVxxKC_GetRFSleep()

```
int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EN_STATUS_E *const pRFSleep )
```

Reads the RFSleep register information.

Table 484. **ST25DVxxKC_GetRFSleep() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRFSleep*	Pointer on a ST25DVxxKC_EN_STATUS value corresponding to the RF Sleep status.

Returns

int32_t enum status.

ST25DVxxKC_GetRFSleep_Dyn()

```
int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EN_STATUS_E *const pRFSleep )
```

Reads the RFSleep dynamic register information.

Table 485. **ST25DVxxKC_GetRFSleep_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRFSleep*	Pointer on a ST25DVxxKC_EN_STATUS values used to return the RF Sleep state.

Returns

int32_t enum status.

ST25DVxxKC_GetUID()

```
int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value )
```

Read UID register.

Table 486. **ST25DVxxKC_GetUID() parameters**

in	*ctx*	structure containing context driver
out	*value*	data pointer to store register content

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_GetVCC_Dyn()

```
int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC )
```

Check if VCC is supplying the ST25DVxxKC.

Table 487. **ST25DVxxKC_GetVCC_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pVCC*	ST25DVxxKC_VCC_STATUS pointer of the VCC status to store.

Table 488. **ST25DVxxKC_GetVCC_Dyn() return values**

NFCTAG	enum status.

ST25DVxxKC_InitEndZone()

```
int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj )
```

Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).

Needs the I²C Password presentation to be effective.. The ST25DVxxKC answers a NACK when setting the End←, Zone2 & EndZone3 to same value than repectively EndZone1 & EndZone2. These NACKs are ok.

Table 489. **ST25DVxxKC_InitEndZone() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_PresentI2CPassword()

```
int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord )
```

Presents I²C password, to authorize the I2C writes to protected areas.

Table 490. **ST25DVxxKC_PresentI2CPassword() parameters**

in	*pObj*	pointer to the device structure object.
in	*PassWord*	Password value on 32bits

Returns
int32_t enum status.

ST25DVxxKC_ReadAFI()

`int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi )`
Reads the ST25DVxxKC AFI.

Table 491. **ST25DVxxKC_ReadAFI() parameters**

in	*pObj*	pointer to the device structure object.
out	*pAfi*	Pointer used to return the ST25DVxxKC AFI value.

Returns
int32_t enum status.

ST25DVxxKC_ReadAfiRFProtection()

`int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi )`
Reads the AFI RF Lock state.

Table 492. **ST25DVxxKC_ReadAfiRFProtection() parameters**

in	*pObj*	pointer to the device structure object.
out	*pLockAfi*	Pointer on a ST25DVxxKC_LOCK_STATUS used to return the ASFID lock state.

Returns
int32_t enum status.

ST25DVxxKC_ReadDSFID()

`int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid )` Reads the ST25DVxxKC DSFID.

Table 493. **ST25DVxxKC_ReadDSFID() parameters**

in	*pObj*	pointer to the device structure object.
out	*pDsfid*	Pointer used to return the ST25DVxxKC DSFID value.

Returns
int32_t enum status.

ST25DVxxKC_ReadDsfidRFProtection()

`int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid )`
Reads the ST25DVxxKC DSFID RF Lock state.

Table 494. **ST25DVxxKC_ReadDsfidRFProtection() parameters**

in	*pObj*	pointer to the device structure object.
out	*pLockDsfid*	Pointer on a ST25DVxxKC_LOCK_STATUS used to return the DSFID lock state.

Returns

int32_t enum status.

ST25DVxxKC_ReadEHCtrl_Dyn()

```
int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)
```

Read value of dynamic EH Ctrl register configuration.

Table 495. **ST25DVxxKC_ReadEHCtrl_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pEH_CTRL*	: ST25DVxxKC_EH_CTRL pointer of the dynamic EH Ctrl configuration to store.

Table 496. **ST25DVxxKC_ReadEHCtrl_Dyn() return values**

NFCTAG	enum status

ST25DVxxKC_ReadEHMode()

```
int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode )
```

Reads the Energy harvesting mode.

Table 497. **ST25DVxxKC_ReadEHMode() parameters**

in	*pObj*	pointer to the device structure object.
out	*pEH_mode*	Pointer on a ST25DVxxKC_EH_MODE_STATUS value corresponding to the Energy Harvesting state.

Returns

int32_t enum status.

ST25DVxxKC_ReadEndZonex()

```
int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_END_ZONE_E EndZone, uint8_t * pEndZ )
```

Reads the value of the an area end address.

Table 498. **ST25DVxxKC_ReadEndZonex() parameters**

in	*pObj*	pointer to the device structure object.
in	*EndZone*	ST25DVxxKC_END_ZONE value corresponding to an area end address.
out	*pEndZ*	Pointer used to return the end address of the area.

Returns

int32_t enum status.

ST25DVxxKC_ReadGPO_Dyn()

```
int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *
GPOConfig )
```

Read value of dynamic GPO register configuration.

Table 499. ST25DVxxKC_ReadGPO_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pGPO*	ST25DVxxKC_GPO pointer of the dynamic GPO configuration to store.

Table 500. ST25DVxxKC_ReadGPO_Dyn() return values

NFCTAG	enum status.

ST25DVxxKC_ReadI2CProtectZone()

```
int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone )
```

Reads the I²C Protected Area state.

Table 501. ST25DVxxKC_ReadI2CProtectZone() parameters

in	*pObj*	pointer to the device structure object.
out	*pProtZone*	Pointer on a ST25DVxxKC_I2C_PROT_ZONE structure used to return the Protected Area state.

Returns

int32_t enum status.

ST25DVxxKC_ReadI2CSecuritySession_Dyn()

```
int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn ( const ST25DVxxKC_Object_t *const
pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession )
```

Reads the status of the security session open register.

Table 502. ST25DVxxKC_ReadI2CSecuritySession_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pSession*	Pointer on a ST25DVxxKC_I2CSSO_STATUS value used to return the session status.

Returns

int32_t enum status.

ST25DVxxKC_ReadICRev()

```
int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const
pICRev )
```

Reads the ST25DVxxKC IC Revision.

Table 503. ST25DVxxKC_ReadICRev() parameters

in	*pObj*	pointer to the device structure object.
out	*pICRev*	Pointer on the uint8_t used to return the ST25DVxxKC IC Revision number.

Returns

int32_t enum status.

ST25DVxxKC_ReadITPulse()

```
int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_PULSE_DURATION_E *const pITtime )
```

Reads the ST25DVxxKC ITtime duration for the GPO pulses.

Table 504. ST25DVxxKC_ReadITPulse() parameters

in	*pObj*	pointer to the device structure object.
out	*pITtime*	Pointer used to return the coefficient for the GPO Pulse duration (Pulse duration = 302,06 us - ITtime ∗ 512 / fc).

Returns

int32_t enum status.

ST25DVxxKC_ReadITSTStatus_Dyn()

```
int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const
pObj,uint8_t *const pITStatus )
```

Reads the IT status register from the ST25DVxxKC.

Table 505. ST25DVxxKC_ReadITSTStatus_Dyn() parameters

in	*pObj*	pointer to the device structure object.
out	*pITStatus*	Pointer on uint8_t, used to return the IT status, such as: • RFUSERSTATE = 0x01 • RFBUSY = 0x02 • RFINTERRUPT = 0x04 • FIELDFALLING = 0x08 • FIELDRISING = 0x10 • RFPUTMSG = 0x20 • RFGETMSG = 0x40 • RFWRITE = 0x80

Returns

int32_t enum status.

ST25DVxxKC_ReadLockCCFile()

```
int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile)
```

Reads the CCile protection state.

Table 506. ST25DVxxKC_ReadLockCCFile() parameters

in	*pObj*	pointer to the device structure object.
out	*pLockCCFile*	Pointer on a ST25DVxxKC_LOCK_CCFILE value corresponding to the lock state of the CCFile.

Returns

int32_t enum status.

ST25DVxxKC_ReadLockCFG()

```
int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_LOCK_STATUS_E *const pLockCfg )
```

Reads the Cfg registers protection.

Table 507. ST25DVxxKC_ReadLockCFG() parameters

in	*pObj*	pointer to the device structure object.
out	*pLockCfg*	Pointer on a ST25DVxxKC_LOCK_STATUS value corresponding to the Cfg registers lock state.

Returns

int32_t enum status.

ST25DVxxKC_ReadMailboxData()

```
int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t
*const pData, const uint16_t Offset, const uint16_t NbByte )
```

Reads N bytes of data from the Mailbox, starting at the specified byte offset.

Table 508. ST25DVxxKC_ReadMailboxData() parameters

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer on the buffer used to return the read data.
in	*Offset*	Offset in the Mailbox memory, byte number to start the read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadMailboxRegister()

```
int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj,
uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte )
```

Reads N bytes from the mailbox registers, starting at the specified I²C address.

Table 509. ST25DVxxKC_ReadMailboxRegister() parameters

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer on the buffer used to return the data.
in	*TarAddr*	I²C memory address to be read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBCtrl_Dyn()

```
int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_MB_CTRL_DYN_STATUS_t *const pCtrlStatus )
```

Reads the Mailbox ctrl dynamic register.

Table 510. **ST25DVxxKC_ReadMBCtrl_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pCtrlStatus*	Pointer on a ST25DVxxKC_MB_CTRL_DYN_STATUS structure used to return the dynamic Mailbox ctrl information.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBLength_Dyn()

```
int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pMBLength )
```

Reads the Mailbox message length dynamic register.

Table 511. **ST25DVxxKC_ReadMBLength_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pMBLength*	Pointer on a uint8_t used to return the Mailbox message length.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBMode()

```
int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMB_mode )
```

Reads the Mailbox mode.

Table 512. **ST25DVxxKC_ReadMBMode() parameters**

in	*pObj*	pointer to the device structure object.
out	*pMB_mode*	Pointer on a ST25DVxxKC_EH_MODE_STATUS value used to return the Mailbox mode.

Returns

int32_t enum status.

ST25DVxxKC_ReadMBWDG()

```
int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pWdgDelay )
```

Reads the Mailbox watchdog duration coefficient.

Table 513. **ST25DVxxKC_ReadMBWDG() parameters**

in	*pObj*	pointer to the device structure object.
out	*pWdgDelay*	Pointer on a uint8_t used to return the watchdog duration coefficient.

Returns

int32_t enum status.

ST25DVxxKC_ReadMemSize()

```
int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo )
```

Reads the ST25DVxxKC Memory Size.

Table 514. **ST25DVxxKC_ReadMemSize() parameters**

in	*pObj*	pointer to the device structure object.
out	*pSizeInfo*	Pointer on a ST25DVxxKC_MEM_SIZE structure used to return the Memory size information.

Returns

int32_t enum status.

ST25DVxxKC_ReadRegister()

```
int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t
*const pData, const uint16_t TarAddr, const uint16_t NbByte )
```

Reads N bytes from Registers, starting at the specified I²C address.

Table 515. **ST25DVxxKC_ReadRegister() parameters**

in	*pObj*	pointer to the device structure object.
out	*pData*	Pointer used to return the read data.
in	*TarAddr*	I²C memory address to be read.
in	*NbByte*	Number of bytes to be read.

Returns

int32_t enum status.

ST25DVxxKC_ReadRFMngt()

```
int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_RF_MNGT_t *const pRF_Mngt )
```

Reads the RF Management configuration.

Table 516. **ST25DVxxKC_ReadRFMngt() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRF_Mngt*	Pointer on a ST25DVxxKC_RF_MNGT structure used to return the RF Management configuration.

Returns

int32_t enum status.

ST25DVxxKC_ReadRFMngt_Dyn()

```
int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj,
ST25DVxxKC_RF_MNGT_t *const pRF_Mngt )
```

Read value of dynamic RF Management configuration.

Table 517. **ST25DVxxKC_ReadRFMngt_Dyn() parameters**

in	*pObj*	pointer to the device structure object.
out	*pRF_Mngt*	ST25DVxxKC_RF_MNGT pointer of the dynamic RF Management configuration to store

Table 518. **ST25DVxxKC_ReadRFMngt_Dyn() return values**

NFCTAG	enum status

ST25DVxxKC_ReadRFZxSS()

```
int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_PROTECTION_ZONE_E Zone,
```

ST25DVxxKC_RF_PROT_ZONE_t ∗const *pRfprotZone*) Reads the RF Zone Security Status (defining the allowed RF accesses).

Table 519. ST25DVxxKC_ReadRFZxSS() parameters

in	*pObj*	pointer to the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value coresponding to the protected area.
out	*pRfprotZone*	Pointer on a ST25DVxxKC_RF_PROT_ZONE value corresponding to the area protection state.

Returns

int32_t enum status.

ST25DVxxKC_ReadUID()

```
int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t
*const pUid )
```

Reads the ST25DVxxKC UID.

Table 520. ST25DVxxKC_ReadUID() parameters

in	*pObj*	pointer to the device structure object.
out	*pUid*	Pointer used to return the ST25DVxxKC UID value.

Returns

int32_t enum status.

ST25DVxxKC_RegisterBusIO()

```
int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const
ST25DVxxKC_IO_t *const pIO )
```

Register Component Bus IO operations.

Table 521. ST25DVxxKC_RegisterBusIO() parameters

out	*pObj*	pointer to the device structure object.
in	*pIO*	pointer to the IO APIs structure.

Returns

int32_t enum status.

ST25DVxxKC_ResetEHENMode_Dyn()

```
int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj )
Dynamically unsets the Energy Harvesting mode.
```

Table 522. ST25DVxxKC_ResetEHENMode_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetGPO_en_Dyn()

`int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Reset dynamique GPO enable configuration.

Table 523. ST25DVxxKC_ResetGPO_en_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Table 524. ST25DVxxKC_ResetGPO_en_Dyn() return values

NFCTAG	enum status.

ST25DVxxKC_ResetMBEN_Dyn()

`int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Unsets the Mailbox Enable dynamic configuration.

Table 525. ST25DVxxKC_ResetMBEN_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFDisable()

`int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj )`

Resets the RF Disable configuration.

Needs the I²C Password presentation to be effective.

Table 526. ST25DVxxKC_ResetRFDisable() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFDisable_Dyn()

`int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Unsets the RF Disable dynamic configuration.

Table 527. ST25DVxxKC_ResetRFDisable_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFSleep()

`int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj )`

Resets the RF Sleep configuration.

Needs the I²C Password presentation to be effective.

Table 528. **ST25DVxxKC_ResetRFSleep() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_ResetRFSleep_Dyn()

`int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Unsets the RF Sleep dynamic configuration.

Table 529. **ST25DVxxKC_ResetRFSleep_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetEH_CTRL_DYN_EH_EN()

`int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write EH_CTRL_DYN_EH_EN register.

Table 530. **ST25DVxxKC_SetEH_CTRL_DYN_EH_EN() parameters**

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetEH_MODE()

`int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )`

Write EH_MODE register.

Table 531. **ST25DVxxKC_SetEH_MODE() parameters**

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetEHENMode_Dyn()

`int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj )`

Dynamically sets the Energy Harvesting mode.

Table 532. **ST25DVxxKC_SetEHENMode_Dyn() parameters**

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetENDA1()

```
int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write ENDA1 register.

Table 533. ST25DVxxKC_SetENDA1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetENDA2()

```
int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write ENDA2 register.

Table 534. ST25DVxxKC_SetENDA2() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetENDA3()

```
int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write ENDA3 register.

Table 535. ST25DVxxKC_SetENDA3() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_ALL()

```
int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_ALL register.

Table 536. ST25DVxxKC_SetGPO1_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_ENABLE()

```
int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_ENABLE register.

Table 537. ST25DVxxKC_SetGPO1_ENABLE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_FIELDCHANGE()

```
int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_FIELDCHANGE register.

Table 538. ST25DVxxKC_SetGPO1_FIELDCHANGE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFACTIVITY()

```
int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFACTIVITY register.

Table 539. ST25DVxxKC_SetGPO1_RFACTIVITY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFGETMSG()

```
int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFGETMSG register.

Table 540. ST25DVxxKC_SetGPO1_RFGETMSG() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFINTERRUPT()

```
int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFINTERRUPT register.

Table 541. ST25DVxxKC_SetGPO1_RFINTERRUPT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFPUTMSG()

int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const *ctx,* const uint8_t *const *value*)

Write GPO1_RFPUTMSG register.

Table 542. ST25DVxxKC_SetGPO1_RFPUTMSG() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFUSERSTATE()

```
int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFUSERSTATE register.

Table 543. ST25DVxxKC_SetGPO1_RFUSERSTATE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO1_RFWRITE()

```
int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO1_RFWRITE register.

Table 544. ST25DVxxKC_SetGPO1_RFWRITE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_ALL()

```
int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO2_ALL register.

Table 545. ST25DVxxKC_SetGPO2_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_I2CWRITEENABLE()

```
int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO2_I2CWRITEENABLE register.

Table 546. ST25DVxxKC_SetGPO2_I2CWRITEENABLE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_ITTIME()

```
int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)
```

Write GPO2_ITTIME register.

Table 547. ST25DVxxKC_SetGPO2_ITTIME() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO2_RFOFF()

```
int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO2_RFOFF register.

Table 548. ST25DVxxKC_SetGPO2_RFOFF() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_DYN_ALL()

```
int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO_DYN_ALL register.

Table 549. ST25DVxxKC_SetGPO_DYN_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_DYN_ENABLE()

```
int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write GPO_DYN_ENABLE register.

Table 550. ST25DVxxKC_SetGPO_DYN_ENABLE() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetGPO_en_Dyn()

```
int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj )
```

Set dynamique GPO enable configuration.

Table 551. ST25DVxxKC_SetGPO_en_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Table 552. ST25DVxxKC_SetGPO_en_Dyn() return values

NFCTAG	enum status.

ST25DVxxKC_SetI2CPASSWD()

```
int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CPASSWD register.

Table 553. ST25DVxxKC_SetI2CPASSWD() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_ALL()

```
int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_ALL register.

Table 554. ST25DVxxKC_SetI2CSS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ1()

```
int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)
```

Write I2CSS_PZ1 register.

Table 555. ST25DVxxKC_SetI2CSS_PZ1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ2()

```
int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ2 register.

Table 556. ST25DVxxKC_SetI2CSS_PZ2() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ3()

```
int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ3 register.

Table 557. ST25DVxxKC_SetI2CSS_PZ3() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetI2CSS_PZ4()

```
int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write I2CSS_PZ4 register.

Table 558. ST25DVxxKC_SetI2CSS_PZ4() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetITTIME_DELAY()

```
int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write ITTIME_DELAY register.

Table 559. ST25DVxxKC_SetITTIME_DELAY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_ALL()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write LOCKCCFILE_ALL register.

Table 560. ST25DVxxKC_SetLOCKCCFILE_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_BLCK0()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write LOCKCCFILE_BLCK0 register.

Table 561. ST25DVxxKC_SetLOCKCCFILE_BLCK0() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCCFILE_BLCK1()

```
int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write LOCKCCFILE_BLCK1 register.

Table 562. ST25DVxxKC_SetLOCKCCFILE_BLCK1() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetLOCKCFG_B0()

```
int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write LOCKCFG_B0 register.

Table 563. ST25DVxxKC_SetLOCKCFG_B0() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_CTRL_DYN_MBEN()

```
int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_CTRL_DYN_MBEN register.

Table 564. ST25DVxxKC_SetMB_CTRL_DYN_MBEN() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_MODE_RW()

```
int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_MODE_RW register.

Table 565. ST25DVxxKC_SetMB_MODE_RW() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMB_WDG_DELAY()

```
int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write MB_WDG_DELAY register.

Table 566. ST25DVxxKC_SetMB_WDG_DELAY() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetMBEN_Dyn()

```
int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj )
```

Sets the Mailbox Enable dynamic configuration.

Table 567. ST25DVxxKC_SetMBEN_Dyn() parameters

in	*pObj*	pointer to the device structure object.

Returns

int32_t enum status.

ST25DVxxKC_SetRF_MNGT_ALL()

```
int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)
```

Write RF_MNGT_ALL register.

Table 568. ST25DVxxKC_SetRF_MNGT_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_ALL()

```
int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_DYN_ALL register.

Table 569. ST25DVxxKC_SetRF_MNGT_DYN_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_RFDIS()

```
int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)
```

Write RF_MNGT_DYN_RFDIS register.

Table 570. ST25DVxxKC_SetRF_MNGT_DYN_RFDIS() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP()

```
int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP ( const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_DYN_RFSLEEP register.

Table 571. ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_RFDIS()

```
int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_RFDIS register.

Table 572. ST25DVxxKC_SetRF_MNGT_RFDIS() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRF_MNGT_RFSLEEP()

```
int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RF_MNGT_RFSLEEP register.

Table 573. ST25DVxxKC_SetRF_MNGT_RFSLEEP() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_ALL()

```
int32_t ST25DVxxKC_SetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_ALL register.

Table 574. ST25DVxxKC_SetRFA1SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_PWDCTRL register.

Table 575. ST25DVxxKC_SetRFA1SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA1SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA1SS_RWPROT register.

Table 576. ST25DVxxKC_SetRFA1SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_ALL()

```
int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_ALL register.

Table 577. ST25DVxxKC_SetRFA2SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_PWDCTRL register.

Table 578. ST25DVxxKC_SetRFA2SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA2SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA2SS_RWPROT register.

Table 579. ST25DVxxKC_SetRFA2SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_ALL()

```
int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA3SS_ALL register.

Table 580. ST25DVxxKC_SetRFA3SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA3SS_PWDCTRL register.

Table 581. ST25DVxxKC_SetRFA3SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA3SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA3SS_RWPROT register.

Table 582. ST25DVxxKC_SetRFA3SS_RWPROT() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_ALL()

```
int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA4SS_ALL register.

Table 583. ST25DVxxKC_SetRFA4SS_ALL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_PWDCTRL()

```
int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA4SS_PWDCTRL register.

Table 584. ST25DVxxKC_SetRFA4SS_PWDCTRL() parameters

in	*ctx*	structure containing context driver
in	*value*	data pointer to write to register

Returns

0 in case of success, an error code otherwise

ST25DVxxKC_SetRFA4SS_RWPROT()

```
int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value )
```

Write RFA4SS_RWPROT register.

Table 585. ST25DVxxKC_SetRFA4SS_RWPROT() parameters

in	*ctx*	structurecontaining context driver
in	*value*	datapointer to write to register

Returns
0 in case of success, an error code otherwise

ST25DVxxKC_SetRFDisable()

`int32_tST25DVxxKC_SetRFDisable ( constST25DVxxKC_Object_t*const`*`pObj`*`)`
Sets the RF Disable configuration.
Needs the I²C Password presentation to be effective.

Table 586. ST25DVxxKC_SetRFDisable() parameters

in	*pObj*	pointerto the device structure object.

Returns
int32_tenum status.

ST25DVxxKC_SetRFDisable_Dyn()

`int32_tST25DVxxKC_SetRFDisable_Dyn (constST25DVxxKC_Object_t*const`*`pObj`*`)`
Sets the RF Disable dynamic configuration.

Table 587. ST25DVxxKC_SetRFDisable_Dyn() parameters

in	*pObj*	pointerto the device structure object.

Returns
int32_tenum status.

ST25DVxxKC_SetRFSleep()

`int32_tST25DVxxKC_SetRFSleep ( constST25DVxxKC_Object_t*const`*`pObj`*`)`
Sets the RF Sleep configuration.
Needs the I²C Password presentation to be effective.

Table 588. ST25DVxxKC_SetRFSleep() parameters

in	*pObj*	pointerto the device structure object.

Returns
int32_tenum status.

ST25DVxxKC_SetRFSleep_Dyn()

`int32_tST25DVxxKC_SetRFSleep_Dyn ( constST25DVxxKC_Object_t*const`*`pObj`*`)`
Sets the RF Sleep dynamic configuration.

Table 589. ST25DVxxKC_SetRFSleep_Dyn() parameters

in	*pObj*	pointerto the device structure object.

Returns
int32_tenum status.

ST25DVxxKC_WriteEHMode()

```
int32_tST25DVxxKC_WriteEHMode (constST25DVxxKC_Object_t*constpObj,
constST25DVxxKC_EH_MODE_STATUS_EEH_mode)
```

Sets the Energy harvesting mode.

Needs the I²C Password presentation to be effective.

Table 590. ST25DVxxKC_WriteEHMode() parameters

in	*pObj*	pointerto the device structure object.
in	*EH_mode*	ST25DVxxKC_EH_MODE_STATUS value for theEnergy harvesting mode to be set.

Returns

int32_tenum status.

ST25DVxxKC_WriteEndZonex()

```
int32_tST25DVxxKC_WriteEndZonex (constST25DVxxKC_Object_t*constpObj,
constST25DVxxKC_END_ZONE_EEndZone,constuint8_t EndZ)
```

Sets the end address of an area.

Needs the I²C Password presentation to be effective.

Note: TheST25DVxxKC answers a NACK when setting the EndZone2 & EndZone3 to same value than repectively EndZone1 & EndZone2. TheseNACKs are ok.

Table 591. ST25DVxxKC_WriteEndZonex() parameters

in	*pObj*	pointerto the device structure object.
in	*EndZone*	ST25DVxxKC_END_ZONE value correspondingto an area.
in	*EndZ*	Endzone value to be written.

Returns

int32_tenum status.

ST25DVxxKC_WriteI2CPassword()

```
int32_tST25DVxxKC_WriteI2CPassword (constST25DVxxKC_Object_t*constpObj,
constST25DVxxKC_PASSWD_tPassWord)
```

Writes a new I²C password.

Needs the I²C Password presentation to be effective.

Table 592. ST25DVxxKC_WriteI2CPassword() parameters

in	*pObj*	pointerto the device structure object.
in	*PassWord*	New I2CPassWord value on 32bits.

Returns

int32_tenum status.

ST25DVxxKC_WriteI2CProtectZonex()

```
int32_tST25DVxxKC_WriteI2CProtectZonex (constST25DVxxKC_Object_t*constpObj,
constST25DVxxKC_PROTECTION_ZONE_EZone,
const ST25DVxxKC_PROTECTION_CONF_EReadWriteProtection)
```

Sets the I²C write-protected state to an EEPROM Area. Needs the I²C Password presentation to be effective.

Table 593. ST25DVxxKC_WriteI2CProtectZonex() parameters

in	*pObj*	pointerto the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value corespondingto the area to protect.
in	*ReadWriteProtection*	ST25DVxxKC_PROTECTION_CONF value correspondingto the protection to be set.

Returns

int32_tenum status.

ST25DVxxKC_WriteITPulse()

```
int32_tST25DVxxKC_WriteITPulse (
const ST25DVxxKC_Object_t*constpObj,
const ST25DVxxKC_PULSE_DURATION_E ITtime)
```

Configures the ST25DVxxKC ITtime duration for the GPO pulse. Needs the I²C Password presentation to be effective.

Table 594. ST25DVxxKC_WriteITPulse() parameters

in	*pObj*	pointerto the device structure object.
in	*ITtime*	Coefficientfor the Pulse duration to be written (Pulse duration = 302,06 us - ITtime ∗512/ fc)

Table 595. ST25DVxxKC_WriteITPulse() return values

int32←, _t	enum status.

ST25DVxxKC_WriteLockCCFile()

```
int32_tST25DVxxKC_WriteLockCCFile ( constST25DVxxKC_Object_t*constpObj,
const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile,
constST25DVxxKC_LOCK_STATUS_ELockCCFile)
```

Locks the CCile to prevent any RF write access. Needs the I²C Password presentation to be effective.

Table 596. ST25DVxxKC_WriteLockCCFile() parameters

in	*pObj*	pointerto the device structure object.
in	*NbBlockCCFile*	ST25DVxxKC_CCFILE_BLOCK value correspondingto the number of blocks to be locked.
in	*LockCCFile*	ST25DVxxKC_LOCK_CCFILE value correspondingto the lock state to apply on the CCFile.

Returns

int32_tenum status.

ST25DVxxKC_WriteLockCFG()

```
int32_tST25DVxxKC_WriteLockCFG ( constST25DVxxKC_Object_t*constpObj,
constST25DVxxKC_LOCK_STATUS_E LockCfg)
```

Lock/Unlock theCfg registers, to prevent any RF write access. Needs the I²C Password presentation to be effective.

Table 597. **ST25DVxxKC_WriteLockCFG() parameters**

in	*pObj*	pointerto the device structure object.
in	*LockCfg*	ST25DVxxKC_LOCK_STATUS value correspondingto the lock state to be written.

Returns

int32_tenum status.

ST25DVxxKC_WriteMailboxData()

```
int32_tST25DVxxKC_WriteMailboxData (const ST25DVxxKC_Object_t*constpObj,
constuint8_t *constpData,
constuint16_t NbByte)
```

Writes N bytes of data in the Mailbox, starting from first Mailbox Address.

Table 598. **ST25DVxxKC_WriteMailboxData() parameters**

in	*pObj*	pointerto the device structure object.
in	*pData*	Pointer tothe buffer containing the data to be written.
in	*NbByte*	Numberof bytes to be written.

Returns

int32_tenum status.

ST25DVxxKC_WriteMailboxRegister()

```
int32_tST25DVxxKC_WriteMailboxRegister (constST25DVxxKC_Object_t*constpObj,
constuint8_t *constpData,const uint16_t TarAddr,const uint16_t NbByte)
```

Writes N bytes to the specified mailbox register.

Table 599. **ST25DVxxKC_WriteMailboxRegister() parameters**

in	*pObj*	pointerto the device structure object.
in	*pData*	Pointer onthe data to be written.
in	*TarAddr*	I2Cregister address to be written.
in	*NbByte*	Numberof bytes to be written.

Returns

int32_tenum status.

ST25DVxxKC_WriteMBMode()

```
int32_tST25DVxxKC_WriteMBMode (constST25DVxxKC_Object_t*constpObj,
const ST25DVxxKC_EN_STATUS_EMB_mode)
```

Sets the Mailbox mode.

Needs the I²C Password presentation to be effective.

Table 600. **ST25DVxxKC_WriteMBMode() parameters**

in	*pObj*	pointerto the device structure object.
out	*MB_mode*	ST25DVxxKC_EN_STATUS value correspondingto the Mailbox mode to be set.

Returns

int32_tenum status.

ST25DVxxKC_WriteMBWDG()

```
int32_tST25DVxxKC_WriteMBWDG ( constST25DVxxKC_Object_t*constpObj,
constuint8_t WdgDelay)
```

Writes the Mailbox watchdog coefficient delay. Needs the I²C Password presentation to be effective.

Table 601. ST25DVxxKC_WriteMBWDG() parameters

in	*pObj*	pointerto the device structure object.
in	*WdgDelay*	Watchdog duration coefficientto be written (Watch dog duration = MB_WDG∗30 ms+/- 6%).

Returns
int32_tenum status.

ST25DVxxKC_WriteRegister()

```
int32_tST25DVxxKC_WriteRegister ( constST25DVxxKC_Object_t*constpObj,
constuint8_t *constpData,const uint16_t TarAddr,const uint16_t NbByte)
```

Writes N bytes to the specified register.
Needs the I²C Password presentation to be effective.

Table 602. ST25DVxxKC_WriteRegister() parameters

in	*pObj*	pointerto the device structure object.
in	*pData*	Pointer onthe data to be written.
in	*TarAddr*	I2Cregister address to written.
in	*NbByte*	Numberof bytes to be written.

Returns
int32_tenum status.

ST25DVxxKC_WriteRFMngt()

```
int32_tST25DVxxKC_WriteRFMngt (
const ST25DVxxKC_Object_t*constpObj,
constuint8_t Rfmngt)
```

Sets the RF Management configuration.
Needs the I²C Password presentation to be effective.

Table 603. ST25DVxxKC_WriteRFMngt() parameters

in	*pObj*	pointerto the device structure object.
in	*Rfmngt*	Value ofthe RF Management configuration to be written.

Returns
int32_tenum status.

ST25DVxxKC_WriteRFMngt_Dyn()

```
int32_tST25DVxxKC_WriteRFMngt_Dyn (
const ST25DVxxKC_Object_t*constpObj,
```

constuint8_t *RF_Mngt*)

Writes a value to the RF Management dynamic register.

Table 604. ST25DVxxKC_WriteRFMngt_Dyn() parameters

in	*pObj*	pointerto the device structure object.
in	*RF_Mngt*	Value tobe written to the RF Management dynamic register.

Returns

int32_tenum status.

ST25DVxxKC_WriteRFZxSS()

int32_tST25DVxxKC_WriteRFZxSS (constST25DVxxKC_Object_t*const*pObj*,
const ST25DVxxKC_PROTECTION_ZONE_E*Zone*,
constST25DVxxKC_RF_PROT_ZONE_t*RfProtZone*)

Writes the RF Zone Security Status (defining the allowed RF accesses) Needs the I²C Password presentation to be effective.

Table 605. ST25DVxxKC_WriteRFZxSS() parameters

in	*pObj*	pointerto the device structure object.
in	*Zone*	ST25DVxxKC_PROTECTION_ZONE value correspondingto the area on which to set the RF protection.
in	*RfProtZone*	Pointer on a ST25DVxxKC_RF_PROT_ZONE value defininf the protection to be set on

Returns

int32_tenum status.

8.2.2 Variable documentation

St25Dvxxkc_Drv

ST25DVxxKC_Drv_t St25Dvxxkc_Drv [extern]

Standard NFC tag driver API for the ST25DVxxKC.

Provides a generic way to access the ST25DVxxKC implementation of the NFC tag standard driver functions.

9 Data structure documentation

9.1 NFCTAG_DrvTypeDef Struct reference

NFCTAG standard driver API structure definition.

9.1.1 Detailed description

NFCTAG standard driver API structure definition.

The documentation for this struct was generated from the following file:

- `Drivers/BSP/ST25-Discovery/st25_discovery_nfctag.h`

9.2 Drivers/BSP/Components/ST25DVxxKC/st25dvxxkc.c file reference

This file provides set of driver functions to manage communication between BSP and ST25DVxxKC chip.

Macros

- `#define ST25DVXXKC_MAX_INSTANCE 1`

  This component driver only supports 1 instance of the component.

Functions

- `int32_t ST25DVxxKC_Init (ST25DVxxKC_Object_t *const pObj)`

  ST25DVxxKC nfctag Initialization.
- `int32_t ST25DVxxKC_ReadID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRef)`

  Reads the ST25DVxxKC ID.
- `int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRev)`

  Reads the ST25DVxxKC IC Revision.
- `int32_t ST25DVxxKC_IsDeviceReady (const ST25DVxxKC_Object_t *const pObj, const uint32_t Trials)`

  Checks the ST25DVxxKC availability.
- `int32_t ST25DVxxKC_GetGPOStatus (const ST25DVxxKC_Object_t *const pObj, uint16_t *const p←, GPOStatus)`

  Reads the ST25DVxxKC GPO configuration.
- `int32_t ST25DVxxKC_ConfigureGPO (const ST25DVxxKC_Object_t *const pObj, const uint16_t ITConf)`

  Configures the ST25DVxxKC GPO.
- `int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_PULSE_DURATION_E*const pITtime)`

  Reads the ST25DVxxKC ITtime duration for the GPO pulses.
- `int32_t ST25DVxxKC_WriteITPulse (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PULSE_DURATION_E ITtime)`

  Configures the ST25DVxxKC ITtime duration for the GPO pulse.
- `int32_t ST25DVxxKC_ReadData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes of Data, starting from the specified I2C address.
- `int32_t ST25DVxxKC_WriteData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes of Data starting from the specified I2C Address.

- `int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes from Registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes to the specified register.
- `int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t *const p←, Uid)`

  Reads the ST25DVxxKC UID.
- `int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid)`

  Reads the ST25DVxxKC DSFID.
- `int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid)`

  Reads the ST25DVxxKC DSFID RF Lock state.
- `int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi)`

  Reads the ST25DVxxKC AFI.
- `int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi)`

  Reads the AFI RF Lock state.
- `int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone)`

  Reads the I2C Protected Area state.
- `int32_t ST25DVxxKC_WriteI2CProtectZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ Zone, const ST25DVxxKC_PROTECTION_CONF_E ReadWriteProtection)`

  Sets the I2C write-protected state to an EEPROM Area.
- `int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile)`

  Reads the CCile protection state.
- `int32_t ST25DVxxKC_WriteLockCCFile (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile, const ST25DVxxKC_LOCK_STATUS_E LockCCFile)`

  Locks the CCile to prevent any RF write access.
- `int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockCfg)`

  Reads the Cfg registers protection.
- `int32_t ST25DVxxKC_WriteLockCFG (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_LOCK_STATUS_E LockCfg)`

  Lock/Unlock the Cfg registers, to prevent any RF write access.
- `int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Presents I2C password, to authorize the I2C writes to protected areas.
- `int32_t ST25DVxxKC_WriteI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Writes a new I2C password.
- `int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, ST25DVxxKC_RF_PROT_ZONE_t *const pRfprotZone)`

  Reads the RF Zone Security Status (defining the allowed RF accesses).

- `int32_t ST25DVxxKC_WriteRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, const ST25DVxxKC_RF_PROT_ZONE_t RfProtZone)`

  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, uint8_t *pEndZ)`

  Reads the value of the an area end address.
- `int32_t ST25DVxxKC_WriteEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, const uint8_t EndZ)`

  Sets the end address of an area.
- `int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj)`

  Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).
- `int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, const uint16_t Zone1←, Length, const uint16_t Zone2Length, const uint16_t Zone3Length, const uint16_t Zone4Length)`

  Creates user areas with defined lengths.
- `int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo)`

  Reads the ST25DVxxKC Memory Size.
- `int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode)`

  Reads the Energy harvesting mode.
- `int32_t ST25DVxxKC_WriteEHMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EH_MODE_STATUS_E EH_mode)`

  Sets the Energy harvesting mode.
- `int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`

  Reads the RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt (const ST25DVxxKC_Object_t *const pObj, const uint8_t Rfmngt)`

  Sets the RF Management configuration.
- `int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`

  Reads the RFDisable register information.
- `int32_t ST25DVxxKC_SetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Disable configuration.
- `int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Disable configuration.
- `int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`

  Reads the RFSleep register information.
- `int32_t ST25DVxxKC_SetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Sets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj)`

  Resets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMB_mode)`

  Reads the Mailbox mode.
- `int32_t ST25DVxxKC_WriteMBMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EN_STATUS_E MB_mode)`

  Sets the Mailbox mode.

- `int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pWdgDelay)`
  Reads the Mailbox watchdog duration coefficient.
- `int32_t ST25DVxxKC_WriteMBWDG (const ST25DVxxKC_Object_t *const pObj, const uint8_t WdgDelay)`
  Writes the Mailbox watchdog coefficient delay.
- `int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t Offset, const uint16_t NbByte)`
  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- `int32_t ST25DVxxKC_WriteMailboxData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t NbByte)`
  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.
- `int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, Data, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteMailboxRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified mailbox register.
- `int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession)`
  Reads the status of the security session open register.
- `int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, ITStatus)`
  Reads the IT status register from the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *GPOConfig)`
  Read value of dynamic GPO register configuration.
- `int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pGPO_en)`
  Get dynamique GPO enable status.
- `int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Set dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Reset dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)`
  Read value of dynamic EH Ctrl register configuration.
- `int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEH_Val)`
  Reads the Energy Harvesting dynamic status.
- `int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Dynamically sets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Dynamically unsets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEHON)`
  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_FIELD_STATUS_E *const pRF_Field)`
  Checks if RF Field is present in front of the ST25DVxxKC.

- `int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC)`
  Check if VCC is supplying the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`
  Read value of dynamic RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, const uint8_t RF_←, Mngt)`
  Writes a value to the RF Management dynamic register.
- `int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`
  Reads the RFDisable dynamic register information.
- `int32_t ST25DVxxKC_SetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t ST25DVxxKC_SetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MB_CTRL_DYN_STATUS_*const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E*const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, MBLength)`
  Reads the Mailbox message length dynamic register.
- `int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_IO_t *const pIO)`
  Register Component Bus IO operations.

Variables

- `ST25DVxxKC_Drv_t St25Dvxxkc_Drv`
  Standard NFC tag driver API for the ST25DVxxKC.

9.3 Drivers/BSP/Components/ST25DVxxKC/st25dvxxkc.h file reference

This file provides set of driver functions to manage communication.

Data structures

- struct ST25DVxxKC_EH_CTRL_t
  ST25DVxxKC EH Ctrl structure definition.

- struct ST25DVxxKC_GPO_t
  ST25DVxxKC GPO structure definition.
- struct ST25DVxxKC_RF_MNGT_t
  ST25DVxxKC RF Management structure definition.
- struct ST25DVxxKC_RF_PROT_ZONE_t
  ST25DVxxKC RF Area protection structure definition.
- struct ST25DVxxKC_I2C_PROT_ZONE_t
  ST25DVxxKC I2C Area protection structure definition.
- struct ST25DVxxKC_MB_CTRL_DYN_STATUS_t
  ST25DVxxKC MB_CTRL_DYN register structure definition.
- struct ST25DVxxKC_LOCK_CCFILE_t
  ST25DVxxKC Lock CCFile structure definition.
- struct ST25DVxxKC_MEM_SIZE_t
  ST25DVxxKC Memory size structure definition.
- struct ST25DVxxKC_UID_t
  ST25DVxxKC UID information structure definition.
- struct ST25DVxxKC_PASSWD_t
  ST25DVxxKC Password structure definition.
- struct ST25DVxxKC_IO_t
  ST25DVxxKC IO API structure definition.
- struct ST25DVxxKC_Object_t
  ST25DVxxKC device structure definition.

Macros

- `#define I_AM_ST25DV04KC 0x50`
  ST25DVxxKC 4Kbits ICref.
- `#define I_AM_ST25DV64KC 0x51`
  ST25DVxxKC 16/64Kbits ICref.
- `#define ST25DVXXKC_AM_I_OPEN_DRAIN(x) (((x) == 0x50) || ((x) == 0x51))`
  Check ST25DVxxKC Open Drain version.
- `#define ST25DVXXKC_AM_I_CMOS(x) (((x) == 0x52) || ((x) == 0x53))`
  Check ST25DVxxKC CMOS version.
- `#define ST25DVXXKC_ADDR_DATA_I2C 0xA6`
  I2C address to be used for ST25DVxxKC Data accesses.
- `#define ST25DVXXKC_ADDR_SYST_I2C 0xAE`
  I2C address to be used for ST25DVxxKC System accesses.
- `#define ST25DVXXKC_WRITE_TIMEOUT 320`
  I2C Time out (ms), min value : (Max write bytes) / (Internal page write) ∗ tw (256/4)∗5.
- `#define ST25DVXXKC_MAX_WRITE_BYTE 256`
  Size of the ST25DVxxKC write buffer.
- `#define ST25DVXXKC_MAX_MAILBOX_LENGTH 256`
  Size of the ST25DVxxKC Mailbox memory.
- `#define ST25DVXXKC_IS_DYNAMIC_REGISTER 0x2000`
  Offset of ST25DVxxKC dynamic registers.

Enumerations

- enum ST25DVxxKC_EN_STATUS_E
  ST25DVxxKC Enable Disable enumerator definition.

- enum ST25DVxxKC_EH_MODE_STATUS_E
  ST25DVxxKC Energy Harvesting mode enumerator definition.
- enum ST25DVxxKC_FIELD_STATUS_E
  ST25DVxxKC FIELD status enumerator definition.
- enum ST25DVxxKC_VCC_STATUS_E
  ST25DVxxKC VCC status enumerator definition.
- enum ST25DVxxKC_PROTECTION_CONF_E
  ST25DVxxKC protection status enumerator definition.
- enum ST25DVxxKC_PROTECTION_ZONE_E
  ST25DVxxKC area protection enumerator definition.
- enum ST25DVxxKC_PASSWD_PROT_STATUS_E
  ST25DVxxKC password protection status enumerator definition.
- enum ST25DVxxKC_LOCK_STATUS_E
  ST25DVxxKC lock status enumerator definition.
- enum ST25DVxxKC_CCFILE_BLOCK_E
  ST25DVxxKC Number of Blocks for the CCFile enumerator definition.
- enum ST25DVxxKC_I2CSSO_STATUS_E
  ST25DVxxKC session status enumerator definition.
- enum ST25DVxxKC_END_ZONE_E
  ST25DVxxKC area end address enumerator definition.
- enum ST25DVxxKC_PULSE_DURATION_E
  ST25DVxxKC IT pulse duration enumerator definition.
- enum ST25DVxxKC_CURRENT_MSG_E
  ST25DVxxKC Mailbox Current Message enumerator definition.

Functions

- `int32_t ST25DVxxKC_ReadRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from Registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified register.
- `int32_t ST25DVxxKC_RegisterBusIO (ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_IO_t *const pIO)`
  Register Component Bus IO operations.
- `int32_t ST25DVxxKC_ReadMemSize (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MEM_SIZE_t *const pSizeInfo)`
  Reads the ST25DVxxKC Memory Size.
- `int32_t ST25DVxxKC_ReadICRev (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pICRev)`
  Reads the ST25DVxxKC IC Revision.
- `int32_t ST25DVxxKC_ReadITPulse (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_PULSE_DURATION_E *const pITtime)`
  Reads the ST25DVxxKC ITtime duration for the GPO pulses.
- `int32_t ST25DVxxKC_WriteITPulse (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PULSE_DURATION_E ITtime)`
  Configures the ST25DVxxKC ITtime duration for the GPO pulse.
- `int32_t ST25DVxxKC_ReadUID (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_UID_t *const p←, Uid)`
  Reads the ST25DVxxKC UID.

- `int32_t ST25DVxxKC_ReadDSFID (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pDsfid)`

  Reads the ST25DVxxKC DSFID.
- `int32_t ST25DVxxKC_ReadDsfidRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockDsfid)`

  Reads the ST25DVxxKC DSFID RF Lock state.
- `int32_t ST25DVxxKC_ReadAFI (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pAfi)`

  Reads the ST25DVxxKC AFI.
- `int32_t ST25DVxxKC_ReadAfiRFProtection (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockAfi)`

  Reads the AFI RF Lock state.
- `int32_t ST25DVxxKC_ReadI2CProtectZone (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2C_PROT_ZONE_t *const pProtZone)`

  Reads the I2C Protected Area state.
- `int32_t ST25DVxxKC_WriteI2CProtectZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ Zone, const ST25DVxxKC_PROTECTION_CONF_E ReadWriteProtection)`

  Sets the I2C write-protected state to an EEPROM Area.
- `int32_t ST25DVxxKC_ReadLockCCFile (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_CCFILE_t *const pLockCCFile)`

  Reads the CCile protection state.
- `int32_t ST25DVxxKC_WriteLockCCFile (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_CCFILE_BLOCK_E NbBlockCCFile, const ST25DVxxKC_LOCK_STATUS_E LockCCFile)`

  Locks the CCile to prevent any RF write access.
- `int32_t ST25DVxxKC_ReadLockCFG (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_LOCK_STATUS_E *const pLockCfg)`

  Reads the Cfg registers protection.
- `int32_t ST25DVxxKC_WriteLockCFG (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_LOCK_STATUS_E LockCfg)`

  Lock/Unlock the Cfg registers, to prevent any RF write access.
- `int32_t ST25DVxxKC_PresentI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Presents I2C password, to authorize the I2C writes to protected areas.
- `int32_t ST25DVxxKC_WriteI2CPassword (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PASSWD_t PassWord)`

  Writes a new I2C password.
- `int32_t ST25DVxxKC_ReadRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, ST25DVxxKC_RF_PROT_ZONE_t *const pRfprotZone)`

  Reads the RF Zone Security Status (defining the allowed RF accesses).
- `int32_t ST25DVxxKC_WriteRFZxSS (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_PROTECTION_ZONE_E Zone, const ST25DVxxKC_RF_PROT_ZONE_t RfProtZone)`

  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t ST25DVxxKC_ReadEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, uint8_t *pEndZ)`

  Reads the value of the an area end address.
- `int32_t ST25DVxxKC_WriteEndZonex (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_END_ZONE_E EndZone, const uint8_t EndZ)`

  Sets the end address of an area.

- `int32_t ST25DVxxKC_InitEndZone (const ST25DVxxKC_Object_t *const pObj)`
  Initializes the end address of the ST25DVxxKC areas with their default values (end of memory).
- `int32_t ST25DVxxKC_CreateUserZone (const ST25DVxxKC_Object_t *const pObj, uint16_t Zone1Length, uint16_t Zone2Length, uint16_t Zone3Length, uint16_t Zone4Length)`
  Creates user areas with defined lengths.
- `int32_t ST25DVxxKC_ReadEHMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_MODE_STATUS_E *const pEH_mode)`
  Reads the Energy harvesting mode.
- `int32_t ST25DVxxKC_WriteEHMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EH_MODE_STATUS_E EH_mode)`
  Sets the Energy harvesting mode.
- `int32_t ST25DVxxKC_ReadRFMngt (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`
  Reads the RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt (const ST25DVxxKC_Object_t *const pObj, const uint8_t Rfmngt)`
  Sets the RF Management configuration.
- `int32_t ST25DVxxKC_GetRFDisable (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`
  Reads the RFDisable register information.
- `int32_t ST25DVxxKC_SetRFDisable (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Disable configuration.
- `int32_t ST25DVxxKC_ResetRFDisable (const ST25DVxxKC_Object_t *const pObj)`
  Resets the RF Disable configuration.
- `int32_t ST25DVxxKC_GetRFSleep (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`
  Reads the RFSleep register information.
- `int32_t ST25DVxxKC_SetRFSleep (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ResetRFSleep (const ST25DVxxKC_Object_t *const pObj)`
  Resets the RF Sleep configuration.
- `int32_t ST25DVxxKC_ReadMBMode (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMB_mode)`
  Reads the Mailbox mode.
- `int32_t ST25DVxxKC_WriteMBMode (const ST25DVxxKC_Object_t *const pObj, const ST25DVxxKC_EN_STATUS_E MB_mode)`
  Sets the Mailbox mode.
- `int32_t ST25DVxxKC_ReadMBWDG (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pWdgDelay)`
  Reads the Mailbox watchdog duration coefficient.
- `int32_t ST25DVxxKC_WriteMBWDG (const ST25DVxxKC_Object_t *const pObj, const uint8_t WdgDelay)`
  Writes the Mailbox watchdog coefficient delay.
- `int32_t ST25DVxxKC_ReadMailboxData (const ST25DVxxKC_Object_t *const pObj, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- `int32_t ST25DVxxKC_WriteMailboxData (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t NbByte)`
  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.

- `int32_t ST25DVxxKC_ReadMailboxRegister (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, Data, const uint16_t TarAddr, const uint16_t NbByte)`

  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- `int32_t ST25DVxxKC_WriteMailboxRegister (const ST25DVxxKC_Object_t *const pObj, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`

  Writes N bytes to the specified mailbox register.
- `int32_t ST25DVxxKC_ReadI2CSecuritySession_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_I2CSSO_STATUS_E *const pSession)`

  Reads the status of the security session open register.
- `int32_t ST25DVxxKC_ReadITSTStatus_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, ITStatus)`

  Reads the IT status register from the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadGPO_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *GPOConfig)`

  Read value of dynamic GPO register configuration.
- `int32_t ST25DVxxKC_GetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pGPO_en)`

  Get dynamique GPO enable status.
- `int32_t ST25DVxxKC_SetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Set dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ResetGPO_en_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Reset dynamique GPO enable configuration.
- `int32_t ST25DVxxKC_ReadEHCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EH_CTRL_t *const pEH_CTRL)`

  Read value of dynamic EH Ctrl register configuration.
- `int32_t ST25DVxxKC_GetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEH_Val)`

  Reads the Energy Harvesting dynamic status.
- `int32_t ST25DVxxKC_SetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Dynamically sets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_ResetEHENMode_Dyn (const ST25DVxxKC_Object_t *const pObj)`

  Dynamically unsets the Energy Harvesting mode.
- `int32_t ST25DVxxKC_GetEHON_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pEHON)`

  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t ST25DVxxKC_GetRFField_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_FIELD_STATUS_E *const pRF_Field)`

  Checks if RF Field is present in front of the ST25DVxxKC.
- `int32_t ST25DVxxKC_GetVCC_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_VCC_STATUS_E *const pVCC)`

  Check if VCC is supplying the ST25DVxxKC.
- `int32_t ST25DVxxKC_ReadRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_RF_MNGT_t *const pRF_Mngt)`

  Read value of dynamic RF Management configuration.
- `int32_t ST25DVxxKC_WriteRFMngt_Dyn (const ST25DVxxKC_Object_t *const pObj, const uint8_t RF_←, Mngt)`

  Writes a value to the RF Management dynamic register.
- `int32_t ST25DVxxKC_GetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFDisable)`

  Reads the RFDisable dynamic register information.

- `int32_t ST25DVxxKC_SetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFDisable_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Disable dynamic configuration.
- `int32_t ST25DVxxKC_GetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t ST25DVxxKC_SetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ResetRFSleep_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBCtrl_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_MB_CTRL_DYN_STATUS_ *const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t ST25DVxxKC_GetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj, ST25DVxxKC_EN_STATUS_E *const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_SetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ResetMBEN_Dyn (const ST25DVxxKC_Object_t *const pObj)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t ST25DVxxKC_ReadMBLength_Dyn (const ST25DVxxKC_Object_t *const pObj, uint8_t *const p←, MBLength)`
  Reads the Mailbox message length dynamic register.

Variables

- `ST25DVxxKC_Drv_t St25Dvxxkc_Drv`
  Standard NFC tag driver API for the ST25DVxxKC.

9.4 Drivers/BSP/Components/ST25DVxxKC/st25dvxxkc_reg.c file reference

Functions

- `int32_t ST25DVxxKC_ReadReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, uint8_t *const Data, const uint16_t len)`
  Read register from component.
- `int32_t ST25DVxxKC_WriteReg (const ST25DVxxKC_Ctx_t *const ctx, const uint16_t Reg, const uint8_t *const Data, const uint16_t len)`
  Write register to component.
- `int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read IC Ref register.
- `int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA1 register.
- `int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA1 register.
- `int32_t ST25DVxxKC_GetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA2 register.

- `int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ENDA2 register.
- `int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ENDA3 register.
- `int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ENDA3 register.
- `int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read DSFID register.
- `int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read AFI register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MEM_SIZE_MSB register.
- `int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read BLK_SIZE register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MEM_SIZE_LSB register.
- `int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ICREV register.
- `int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read UID register.
- `int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CPASSWD register.
- `int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CPASSWD register.
- `int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKDSFI register.
- `int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKAFI register.
- `int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_MODE_RW register.
- `int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_MODE_RW register.
- `int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_WDG_DELAY register.

- `int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MBLEN_DYN_MBLEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_STRESERVED register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_CURRENTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_DEVICECODE register.
- `int32_t ST25DVxxKC_SetI2CCFG_DEVICECODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write I2CCFG_DEVICECODE register.
- `int32_t ST25DVxxKC_GetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_E0 register.
- `int32_t ST25DVxxKC_SetI2CCFG_E0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CCFG_E0 register.
- `int32_t ST25DVxxKC_GetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CCFG_RFSWITCHOFF register.
- `int32_t ST25DVxxKC_SetI2CCFG_RFSWITCHOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write I2CCFG_RFSWITCHOFF register.

- `int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ALL register.

- `int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ALL register.
- `int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_RFOFF register.
- `int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_RFOFF register.
- `int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ITTIME register.
- `int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ITTIME register.
- `int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ALL register.
- `int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ALL register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITTIME_DELAY register.
- `int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ITTIME_DELAY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFACTIVITY register.

- `int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_FIELDFALLING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_FIELDRISING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFGETMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFWRITE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_ALL register.
- `int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_MODE register.
- `int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write EH_MODE register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_FIELD_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_VCC_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_RFDIS register.

- int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_RFSLEEP register.
- int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RF_MNGT_RFSLEEP register.
- int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_ALL register.
- int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RF_MNGT_ALL register.
- int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_DYN_RFDIS register.
- int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)

  Write RF_MNGT_DYN_RFDIS register.
- int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_DYN_RFSLEEP register.
- int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RF_MNGT_DYN_RFSLEEP register.
- int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_DYN_RFOFF register.
- int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)

  Write RF_MNGT_DYN_RFOFF register.
- int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RF_MNGT_DYN_ALL register.
- int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RF_MNGT_DYN_ALL register.
- int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RFA1SS_PWDCTRL register.
- int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RFA1SS_PWDCTRL register.
- int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RFA1SS_RWPROT register.
- int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)

  Write RFA1SS_RWPROT register.
- int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)

  Read RFA1SS_ALL register.

- `int32_t ST25DVxxKC_SetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_RWPROT register.

- `int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_ALL register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_ALL register.
- `int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_ALL register.

- `int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read LOCKCFG_B0 register.
- `int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write LOCKCFG_B0 register.
- `int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2C_SSO_DYN_I2CSSO register.

9.5 Drivers/BSP/Components/ST25DVxxKC/st25dvxxkc_reg.h file reference

Data structures

- struct ST25DVxxKC_Ctx_t
  ST25DVxxKC context structure.

Macros

- `#define ST25DVXXKC_GPO1_REG 0x0000`
  ST25DVxxKC GPO1 register address.
- `#define ST25DVXXKC_GPO2_REG 0x0001`
  ST25DVxxKC GPO2 register address.
- `#define ST25DVXXKC_EH_MODE_REG 0x0002`
  ST25DVxxKC Energy Harvesting register address.
- `#define ST25DVXXKC_RF_MNGT_REG 0x0003`
  ST25DVxxKC RF management register address.
- `#define ST25DVXXKC_RFA1SS_REG 0x0004`
  ST25DVxxKC Area 1 security register address.
- `#define ST25DVXXKC_ENDA1_REG 0x0005`
  ST25DVxxKC Area 1 end address register address.
- `#define ST25DVXXKC_RFA2SS_REG 0x0006`
  ST25DVxxKC Area 2 security register address.
- `#define ST25DVXXKC_ENDA2_REG 0x0007`
  ST25DVxxKC Area 2 end address register address.
- `#define ST25DVXXKC_RFA3SS_REG 0x0008`
  ST25DVxxKC Area 3 security register address.
- `#define ST25DVXXKC_ENDA3_REG 0x0009`
  ST25DVxxKC Area 3 end address register address.
- `#define ST25DVXXKC_RFA4SS_REG 0x000A`
  ST25DVxxKC Area 4 security register address.
- `#define ST25DVXXKC_I2CSS_REG 0x000B`
  ST25DVxxKC I2C security register address.
- `#define ST25DVXXKC_LOCKCCFILE_REG 0x000C`
  ST25DVxxKC Capability Container lock register address.
- `#define ST25DVXXKC_MB_REG 0x000D`
  ST25DVxxKC Mailbox mode register address.

- `#define ST25DVXXKC_I2CCFG_REG 0x000E`
  ST25DVxxKC I2C configuration register address.
- `#define ST25DVXXKC_LOCKCFG_REG 0x000F`
  ST25DVxxKC Configuration lock register address.
- `#define ST25DVXXKC_LOCKDSFID_REG 0x0010`
  ST25DVxxKC DSFID lock register address.
- `#define ST25DVXXKC_LOCKAFI_REG 0x0011`
  ST25DVxxKC AFI lock register address.
- `#define ST25DVXXKC_DSFID_REG 0x0012`
  ST25DVxxKC DSFID register address.
- `#define ST25DVXXKC_AFI_REG 0x0013`
  ST25DVxxKC AFI register address.
- `#define ST25DVXXKC_MEM_SIZE_LSB_REG 0x0014`
  ST25DVxxKC Memory size register address.
- `#define ST25DVXXKC_MEM_SIZE_MSB_REG 0x0015`
  ST25DVxxKC Memory size register address.
- `#define ST25DVXXKC_BLK_SIZE_REG 0x0016`
  ST25DVxxKC Block size register address.
- `#define ST25DVXXKC_ICREF_REG 0x0017`
  ST25DVxxKC ICref register address.
- `#define ST25DVXXKC_UID_REG 0x0018`
  ST25DVxxKC UID register address.
- `#define ST25DVXXKC_ICREV_REG 0x0020`
  ST25DVxxKC IC revision register address.
- `#define ST25DVXXKC_I2CPASSWD_REG 0x0900`
  ST25DVxxKC I2C password register address.
- `#define ST25DVXXKC_GPO_DYN_REG 0x2000`
  ST25DVxxKC GPO dynamic register address.
- `#define ST25DVXXKC_EH_CTRL_DYN_REG 0x2002`
  ST25DVxxKC Energy Harvesting control dynamic register address.
- `#define ST25DVXXKC_RF_MNGT_DYN_REG 0x2003`
  ST25DVxxKC RF management dynamic register address.
- `#define ST25DVXXKC_I2C_SSO_DYN_REG 0x2004`
  ST25DVxxKC I2C secure session opened dynamic register address.
- `#define ST25DVXXKC_ITSTS_DYN_REG 0x2005`
  ST25DVxxKC Interrupt status dynamic register address.
- `#define ST25DVXXKC_MB_CTRL_DYN_REG 0x2006`
  ST25DVxxKC Mailbox control dynamic register address.
- `#define ST25DVXXKC_MBLEN_DYN_REG 0x2007`
  ST25DVxxKC Mailbox message length dynamic register address.
- `#define ST25DVXXKC_MAILBOX_RAM_REG 0x2008`
  ST25DVxxKC Mailbox buffer address.
- `#define NFCTAG_OK (0)`
  NFCTAG status enumerator definition.

Typedefs

- `typedef int32_t(* ST25DVxxKC_WriteReg_Func) (const void *const, const uint16_t, const uint8_t *const, const uint16_t)`
  ST25DVxxKC Write register function typedef definition.

- `typedef int32_t(* ST25DVxxKC_ReadReg_Func) (const void *const, const uint16_t, uint8_t *const, const uint16_t)`
  ST25DVxxKC Read register function typedef definition.

Functions

- `int32_t ST25DVxxKC_GetICREF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read IC Ref register.
- `int32_t ST25DVxxKC_GetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA1 register.
- `int32_t ST25DVxxKC_SetENDA1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA1 register.
- `int32_t ST25DVxxKC_GetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA2 register.
- `int32_t ST25DVxxKC_SetENDA2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA2 register.
- `int32_t ST25DVxxKC_GetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ENDA3 register.
- `int32_t ST25DVxxKC_SetENDA3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`
  Write ENDA3 register.
- `int32_t ST25DVxxKC_GetDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read DSFID register.
- `int32_t ST25DVxxKC_GetAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read AFI register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_MSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read MEM_SIZE_MSB register.
- `int32_t ST25DVxxKC_GetBLK_SIZE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read BLK_SIZE register.
- `int32_t ST25DVxxKC_GetMEM_SIZE_LSB (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read MEM_SIZE_LSB register.
- `int32_t ST25DVxxKC_GetICREV (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read ICREV register.
- `int32_t ST25DVxxKC_GetUID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read UID register.
- `int32_t ST25DVxxKC_GetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`
  Read I2CPASSWD register.

- `int32_t ST25DVxxKC_SetI2CPASSWD (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CPASSWD register.
- `int32_t ST25DVxxKC_GetLOCKDSFID (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKDSFI register.
- `int32_t ST25DVxxKC_GetLOCKAFI (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKAFI register.
- `int32_t ST25DVxxKC_GetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_MODE_RW register.
- `int32_t ST25DVxxKC_SetMB_MODE_RW (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_MODE_RW register.
- `int32_t ST25DVxxKC_GetMBLEN_DYN_MBLEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MBLEN_DYN_MBLEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_SetMB_CTRL_DYN_MBEN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write MB_CTRL_DYN_MBEN register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_STRESERVED (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_STRESERVED register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_HOSTMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_HOSTMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_RFMISSMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read MB_CTRL_DYN_RFMISSMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_CURRENTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_CURRENTMSG register.
- `int32_t ST25DVxxKC_GetMB_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read MB_WDG_DELAY register.
- `int32_t ST25DVxxKC_SetMB_WDG_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write MB_WDG_DELAY register.

- `int32_t ST25DVxxKC_GetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_SetGPO1_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_SetGPO1_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFACTIVITY register.
- `int32_t ST25DVxxKC_GetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_SetGPO1_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_SetGPO1_FIELDCHANGE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write GPO1_FIELDCHANGE register.
- `int32_t ST25DVxxKC_GetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_SetGPO1_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFGETMSG register.
- `int32_t ST25DVxxKC_GetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_SetGPO1_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_RFWRITE register.
- `int32_t ST25DVxxKC_GetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO1_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO1_ALL register.

- `int32_t ST25DVxxKC_SetGPO1_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO1_ALL register.
- `int32_t ST25DVxxKC_GetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_SetGPO2_I2CWRITEENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_I2CWRITEENABLE register.
- `int32_t ST25DVxxKC_GetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_RFOFF register.
- `int32_t ST25DVxxKC_SetGPO2_RFOFF (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_RFOFF register.
- `int32_t ST25DVxxKC_GetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ITTIME register.
- `int32_t ST25DVxxKC_SetGPO2_ITTIME (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ITTIME register.
- `int32_t ST25DVxxKC_GetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO2_ALL register.
- `int32_t ST25DVxxKC_SetGPO2_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO2_ALL register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ENABLE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ENABLE register.
- `int32_t ST25DVxxKC_GetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_SetGPO_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write GPO_DYN_ALL register.
- `int32_t ST25DVxxKC_GetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITTIME_DELAY register.
- `int32_t ST25DVxxKC_SetITTIME_DELAY (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write ITTIME_DELAY register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFUSERSTATE (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFUSERSTATE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFACTIVITY (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFACTIVITY register.

- `int32_t ST25DVxxKC_GetITSTS_DYN_RFINTERRUPT (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_RFINTERRUPT register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDFALLING (const ST25DVxxKC_Ctx_t *const ctx, uint8_←, t *const value)`

  Read ITSTS_DYN_FIELDFALLING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_FIELDRISING (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_FIELDRISING register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFPUTMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFPUTMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFGETMSG (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFGETMSG register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_RFWRITE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_RFWRITE register.
- `int32_t ST25DVxxKC_GetITSTS_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read ITSTS_DYN_ALL register.
- `int32_t ST25DVxxKC_GetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_MODE register.
- `int32_t ST25DVxxKC_SetEH_MODE (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write EH_MODE register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_SetEH_CTRL_DYN_EH_EN (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write EH_CTRL_DYN_EH_EN register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_EH_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_EH_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_FIELD_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_FIELD_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_VCC_ON (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_VCC_ON register.
- `int32_t ST25DVxxKC_GetEH_CTRL_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read EH_CTRL_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_RFDIS register.

- `int32_t ST25DVxxKC_GetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_RFSLEEP register.
- `int32_t ST25DVxxKC_GetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_ALL register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_RFDIS register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFDIS (const ST25DVxxKC_Ctx_t *const ctx, const uint8_←, t *const value)`

  Write RF_MNGT_DYN_RFDIS register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_RFSLEEP (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_DYN_RFSLEEP register.
- `int32_t ST25DVxxKC_GetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_SetRF_MNGT_DYN_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RF_MNGT_DYN_ALL register.
- `int32_t ST25DVxxKC_GetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA1SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA1SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA1SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA1SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA1SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_PWDCTRL register.

- `int32_t ST25DVxxKC_SetRFA2SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA2SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA2SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA2SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA2SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA3SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA3SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA3SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA3SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA3SS_ALL register.
- `int32_t ST25DVxxKC_GetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_SetRFA4SS_PWDCTRL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_PWDCTRL register.
- `int32_t ST25DVxxKC_GetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_SetRFA4SS_RWPROT (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_RWPROT register.
- `int32_t ST25DVxxKC_GetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read RFA4SS_ALL register.
- `int32_t ST25DVxxKC_SetRFA4SS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write RFA4SS_ALL register.

- `int32_t ST25DVxxKC_GetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ1 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ2 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ2 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ3 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ3 register.
- `int32_t ST25DVxxKC_GetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_SetI2CSS_PZ4 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_PZ4 register.
- `int32_t ST25DVxxKC_GetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read I2CSS_ALL register.
- `int32_t ST25DVxxKC_SetI2CSS_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write I2CSS_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK0 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_BLCK1 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_BLCK1 register.
- `int32_t ST25DVxxKC_GetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_SetLOCKCCFILE_ALL (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)`

  Write LOCKCCFILE_ALL register.
- `int32_t ST25DVxxKC_GetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)`

  Read LOCKCFG_B0 register.

- int32_t ST25DVxxKC_SetLOCKCFG_B0 (const ST25DVxxKC_Ctx_t *const ctx, const uint8_t *const value)
  Write LOCKCFG_B0 register.
- int32_t ST25DVxxKC_GetI2C_SSO_DYN_I2CSSO (const ST25DVxxKC_Ctx_t *const ctx, uint8_t *const value)
  Read I2C_SSO_DYN_I2CSSO register.

9.6 Drivers/BSP/ST25-Discovery/st25_discovery_nfctag.c file reference

This file provides a set of functions needed to manage a nfc dual interface eeprom memory.

Functions

- int32_t BSP_NFCTAG_Init (const uint32_t Instance)
  Initializes peripherals used by the I2C NFCTAG driver.
- void BSP_NFCTAG_DeInit (const uint32_t Instance)
  Deinitializes peripherals used by the I2C NFCTAG driver.
- int32_t BSP_NFCTAG_isInitialized (const uint32_t Instance)
  Check if the nfctag is initialized.
- int32_t BSP_NFCTAG_ReadID (const uint32_t Instance, uint8_t *const wai_id)
  Read the ID of the nfctag.
- int32_t BSP_NFCTAG_IsDeviceReady (const uint32_t Instance, const uint32_t Trials)
  Check if the nfctag is available.
- int32_t BSP_NFCTAG_ConfigIT (const uint32_t Instance, const uint16_t ITConfig)
  Configure nfctag interrupt.
- int32_t BSP_NFCTAG_GetITStatus (const uint32_t Instance, uint16_t *const ITConfig)
  Get nfctag interrupt configutration.
- int32_t BSP_NFCTAG_ReadData (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)
  Reads data in the nfctag at specific address.
- int32_t BSP_NFCTAG_WriteData (const uint32_t Instance, const uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t Size)
  Writes data in the nfctag at specific address.
- int32_t BSP_NFCTAG_ReadRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)
  Reads nfctag Register.
- int32_t BSP_NFCTAG_WriteRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)
  Writes nfctag Register.
- uint32_t BSP_NFCTAG_GetByteSize (const uint32_t Instance)
  Return the size of the nfctag.
- int32_t BSP_NFCTAG_ReadICRev (const uint32_t Instance, uint8_t *const pICRev)
  Reads the ST25DV-I2C IC Revision.
- int32_t BSP_NFCTAG_ReadITPulse (const uint32_t Instance, void *const pITtime)
  Reads the ST25DV-I2C ITtime duration for the GPO pulses.
- int32_t BSP_NFCTAG_WriteITPulse (const uint32_t Instance, const uint8_t ITtime)
  Configures the ST25DV-I2C ITtime duration for the GPO pulse.
- int32_t BSP_NFCTAG_ReadUID (const uint32_t Instance, void *const pUid)
  Reads the ST25DV-I2C UID.

- `int32_t BSP_NFCTAG_ReadDSFID (const uint32_t Instance, uint8_t *const pDsfid)`

  Reads the ST25DV-I2C DSFID.
- `int32_t BSP_NFCTAG_ReadDsfidRFProtection (const uint32_t Instance, void *const pLockDsfid)`

  Reads the ST25DV-I2C DSFID RF Lock state.
- `int32_t BSP_NFCTAG_ReadAFI (const uint32_t Instance, uint8_t *const pAfi)`

  Reads the ST25DV-I2C AFI.
- `int32_t BSP_NFCTAG_ReadAfiRFProtection (const uint32_t Instance, void *const pLockAfi)`

  Reads the AFI RF Lock state.
- `int32_t BSP_NFCTAG_ReadI2CProtectZone (const uint32_t Instance, void *const pProtZone)`

  Reads the I2C Protected Area state.
- `int32_t BSP_NFCTAG_WriteI2CProtectZonex (const uint32_t Instance, const uint8_t Zone, const uint8_←, t ReadWriteProtection)`

  Sets the I2C write-protected state to an EEPROM Area.
- `int32_t BSP_NFCTAG_ReadLockCCFile (const uint32_t Instance, void *const pLockCCFile)`

  Reads the CCile protection state.
- `int32_t BSP_NFCTAG_WriteLockCCFile (const uint32_t Instance, const uint8_t NbBlockCCFile, const uint8←,_t LockCCFile)`

  Locks the CCfile to prevent any RF write access.
- `int32_t BSP_NFCTAG_ReadLockCFG (const uint32_t Instance, void *const pLockCfg)`

  Reads the Cfg registers protection.
- `int32_t BSP_NFCTAG_WriteLockCFG (const uint32_t Instance, const uint8_t LockCfg)`

  Lock/Unlock the Cfg registers, to prevent any RF write access.
- `int32_t BSP_NFCTAG_PresentI2CPassword (const uint32_t Instance, const void *const PassWord)`

  Presents I2C password, to authorize the I2C writes to protected areas.
- `int32_t BSP_NFCTAG_WriteI2CPassword (const uint32_t Instance, const void *const PassWord)`

  Writes a new I2C password.
- `int32_t BSP_NFCTAG_ReadRFZxSS (const uint32_t Instance, const uint8_t Zone, void *const pRfprotZone)`

  Reads the RF Zone Security Status (defining the allowed RF accesses).
- `int32_t BSP_NFCTAG_WriteRFZxSS (const uint32_t Instance, const uint8_t Zone, const void *const Rf←, ProtZone)`

  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t BSP_NFCTAG_ReadEndZonex (const uint32_t Instance, const uint8_t EndZone, uint8_t *pEndZ)`

  Reads the value of the end area address.
- `int32_t BSP_NFCTAG_WriteEndZonex (const uint32_t Instance, const uint8_t EndZone, const uint8_t EndZ)`

  Sets the end address of an area.
- `int32_t BSP_NFCTAG_InitEndZone (const uint32_t Instance)`

  Initializes the end address of the ST25DV-I2C areas with their default values (end of memory).
- `int32_t BSP_NFCTAG_CreateUserZone (const uint32_t Instance, uint16_t Zone1Length, uint16_t Zone2←, Length, uint16_t Zone3Length, uint16_t Zone4Length)`

  Creates user areas with defined lengths.

- int32_t BSP_NFCTAG_ReadMemSize (const uint32_t Instance, void *const pSizeInfo)
  Reads the ST25DV-I2C Memory Size.
- int32_t BSP_NFCTAG_ReadEHMode (const uint32_t Instance, void *const pEH_mode)
  Reads the Energy harvesting mode.
- int32_t BSP_NFCTAG_WriteEHMode (const uint32_t Instance, const uint8_t EH_mode)
  Sets the Energy harvesting mode.
- int32_t BSP_NFCTAG_ReadRFMngt (const uint32_t Instance, void *const pRF_Mngt)
  Reads the RF Management configuration.
- int32_t BSP_NFCTAG_WriteRFMngt (const uint32_t Instance, const uint8_t Rfmngt)
  Sets the RF Management configuration.
- int32_t BSP_NFCTAG_GetRFDisable (const uint32_t Instance, void *const pRFDisable)
  Reads the RFDisable register information.
- int32_t BSP_NFCTAG_SetRFDisable (const uint32_t Instance)
  Sets the RF Disable configuration.
- int32_t BSP_NFCTAG_ResetRFDisable (const uint32_t Instance)
  Resets the RF Disable configuration.
- int32_t BSP_NFCTAG_GetRFSleep (const uint32_t Instance, void *const pRFSleep)
  Reads the RFSleep register information.
- int32_t BSP_NFCTAG_SetRFSleep (const uint32_t Instance)
  Sets the RF Sleep configuration.
- int32_t BSP_NFCTAG_ResetRFSleep (const uint32_t Instance)
  Resets the RF Sleep configuration.
- int32_t BSP_NFCTAG_ReadMBMode (const uint32_t Instance, void *const pMB_mode)
  Reads the Mailbox mode.
- int32_t BSP_NFCTAG_WriteMBMode (const uint32_t Instance, const uint8_t MB_mode)
  Sets the Mailbox mode.
- int32_t BSP_NFCTAG_ReadMBWDG (const uint32_t Instance, uint8_t *const pWdgDelay)
  Reads the Mailbox watchdog duration coefficient.
- int32_t BSP_NFCTAG_WriteMBWDG (const uint32_t Instance, const uint8_t WdgDelay)
  Writes the Mailbox watchdog coefficient delay.
- int32_t BSP_NFCTAG_ReadMailboxData (const uint32_t Instance, uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t NbByte)
  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- int32_t BSP_NFCTAG_WriteMailboxData (const uint32_t Instance, const uint8_t *const pData, const uint16_t NbByte)
  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.
- int32_t BSP_NFCTAG_ReadMailboxRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)
  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- int32_t BSP_NFCTAG_WriteMailboxRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)
  Writes N bytes to the specified mailbox register.
- int32_t BSP_NFCTAG_ReadI2CSecuritySession_Dyn (const uint32_t Instance, void *const pSession)
  Reads the status of the security session open register.
- int32_t BSP_NFCTAG_ReadITSTStatus_Dyn (const uint32_t Instance, uint8_t *const pITStatus)
  Reads the IT status register from the ST25DV-I2C.

- `int32_t BSP_NFCTAG_ReadGPO_Dyn (const uint32_t Instance, uint8_t *GPOConfig)`
  Read value of dynamic GPO register configuration.
- `int32_t BSP_NFCTAG_GetGPO_en_Dyn (const uint32_t Instance, void *const pGPO_en)`
  Get dynamique GPO enable status.
- `int32_t BSP_NFCTAG_SetGPO_en_Dyn (const uint32_t Instance)`
  Set dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ResetGPO_en_Dyn (const uint32_t Instance)`
  Reset dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ReadEHCtrl_Dyn (const uint32_t Instance, void *const pEH_CTRL)`
  Read value of dynamic EH Ctrl register configuration.
- `int32_t BSP_NFCTAG_GetEHENMode_Dyn (const uint32_t Instance, void *const pEH_Val)`
  Reads the Energy Harvesting dynamic status.
- `int32_t BSP_NFCTAG_SetEHENMode_Dyn (const uint32_t Instance)`
  Dynamically sets the Energy Harvesting mode.
- `int32_t BSP_NFCTAG_ResetEHENMode_Dyn (const uint32_t Instance)`
  Dynamically unsets the Energy Harvesting mode.
- `int32_t BSP_NFCTAG_GetEHON_Dyn (const uint32_t Instance, void *const pEHON)`
  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t BSP_NFCTAG_GetRFField_Dyn (const uint32_t Instance, void *const pRF_Field)`
  Checks if RF Field is present in front of the ST25DV-I2C.
- `int32_t BSP_NFCTAG_GetVCC_Dyn (const uint32_t Instance, void *const pVCC)`
  Check if VCC is supplying the ST25DV-I2C.
- `int32_t BSP_NFCTAG_ReadRFMngt_Dyn (const uint32_t Instance, void *const pRF_Mngt)`
  Read value of dynamic RF Management configuration.
- `int32_t BSP_NFCTAG_WriteRFMngt_Dyn (const uint32_t Instance, const uint8_t RF_Mngt)`
  Writes a value to the RF Management dynamic register.
- `int32_t BSP_NFCTAG_GetRFDisable_Dyn (const uint32_t Instance, void *const pRFDisable)`
  Reads the RFDisable dynamic register information.
- `int32_t BSP_NFCTAG_SetRFDisable_Dyn (const uint32_t Instance)`
  Sets the RF Disable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetRFDisable_Dyn (const uint32_t Instance)`
  Unsets the RF Disable dynamic configuration.
- `int32_t BSP_NFCTAG_GetRFSleep_Dyn (const uint32_t Instance, void *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t BSP_NFCTAG_SetRFSleep_Dyn (const uint32_t Instance)`
  Sets the RF Sleep dynamic configuration.
- `int32_t BSP_NFCTAG_ResetRFSleep_Dyn (const uint32_t Instance)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBCtrl_Dyn (const uint32_t Instance, void *const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t BSP_NFCTAG_GetMBEN_Dyn (const uint32_t Instance, void *const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.

- `int32_t BSP_NFCTAG_SetMBEN_Dyn (const uint32_t Instance)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetMBEN_Dyn (const uint32_t Instance)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBLength_Dyn (const uint32_t Instance, uint8_t *const pMBLength)`
  Reads the Mailbox message length dynamic register.

9.7 Drivers/BSP/ST25-Discovery/st25_discovery_nfctag.h File Reference

This file contains definitions for the `st25_discovery_nfctag.c` specific functions.

Data structures

- struct NFCTAG_DrvTypeDef
  NFCTAG standard driver API structure definition.

Macros

- `#define NFCTAG_4K_SIZE ((uint32_t) 0x200)`
  Number of Bytes for the NFCTAG 4Kbits.
- `#define NFCTAG_16K_SIZE ((uint32_t) 0x800)`
  Number of Bytes for the NFCTAG 16Kbits.
- `#define NFCTAG_64K_SIZE ((uint32_t) 0x2000)`
  Number of Bytes for the NFCTAG 64Kbits.
- `#define NFCTAG_OK (0)`
  NFCTAG status enumerator definition.
- `#define BSP_NFCTAG_INSTANCE 0`
  NFCTAG INSTANCE.

Functions

- `int32_t BSP_NFCTAG_Init (const uint32_t Instance)`
  Initializes peripherals used by the I2C NFCTAG driver.
- `void BSP_NFCTAG_DeInit (const uint32_t Instance)`
  Deinitializes peripherals used by the I2C NFCTAG driver.
- `int32_t BSP_NFCTAG_isInitialized (const uint32_t Instance)`
  Check if the nfctag is initialized.
- `int32_t BSP_NFCTAG_ReadID (const uint32_t Instance, uint8_t *const wai_id)`
  Read the ID of the nfctag.
- `int32_t BSP_NFCTAG_ConfigIT (const uint32_t Instance, const uint16_t ITConfig)`
  Configure nfctag interrupt.
- `int32_t BSP_NFCTAG_GetITStatus (const uint32_t Instance, uint16_t *const ITConfig)`
  Get nfctag interrupt configutration.
- `int32_t BSP_NFCTAG_ReadData (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Reads data in the nfctag at specific address.
- `int32_t BSP_NFCTAG_WriteData (const uint32_t Instance, const uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t Size)`
  Writes data in the nfctag at specific address.

- `int32_t BSP_NFCTAG_ReadRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Reads nfctag Register.
- `int32_t BSP_NFCTAG_WriteRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t Size)`
  Writes nfctag Register.
- `int32_t BSP_NFCTAG_IsDeviceReady (const uint32_t Instance, const uint32_t Trials)`
  Check if the nfctag is available.
- `uint32_t BSP_NFCTAG_GetByteSize (const uint32_t Instance)`
  Return the size of the nfctag.
- `int32_t BSP_NFCTAG_ReadICRev (const uint32_t Instance, uint8_t *const pICRev)`
  Reads the ST25DV-I2C IC Revision.
- `int32_t BSP_NFCTAG_ReadITPulse (const uint32_t Instance, void *const pITtime)`
  Reads the ST25DV-I2C ITtime duration for the GPO pulses.
- `int32_t BSP_NFCTAG_WriteITPulse (const uint32_t Instance, const uint8_t ITtime)`
  Configures the ST25DV-I2C ITtime duration for the GPO pulse.
- `int32_t BSP_NFCTAG_ReadUID (const uint32_t Instance, void *const pUid)`
  Reads the ST25DV-I2C UID.
- `int32_t BSP_NFCTAG_ReadDSFID (const uint32_t Instance, uint8_t *const pDsfid)`
  Reads the ST25DV-I2C DSFID.
- `int32_t BSP_NFCTAG_ReadDsfidRFProtection (const uint32_t Instance, void *const pLockDsfid)`
  Reads the ST25DV-I2C DSFID RF Lock state.
- `int32_t BSP_NFCTAG_ReadAFI (const uint32_t Instance, uint8_t *const pAfi)`
  Reads the ST25DV-I2C AFI.
- `int32_t BSP_NFCTAG_ReadAfiRFProtection (const uint32_t Instance, void *const pLockAfi)`
  Reads the AFI RF Lock state.
- `int32_t BSP_NFCTAG_ReadI2CProtectZone (const uint32_t Instance, void *const pProtZone)`
  Reads the I2C Protected Area state.
- `int32_t BSP_NFCTAG_WriteI2CProtectZonex (const uint32_t Instance, const uint8_t Zone, const uint8_←, t ReadWriteProtection)`
  Sets the I2C write-protected state to an EEPROM Area.
- `int32_t BSP_NFCTAG_ReadLockCCFile (const uint32_t Instance, void *const pLockCCFile)`
  Reads the CCile protection state.
- `int32_t BSP_NFCTAG_WriteLockCCFile (const uint32_t Instance, const uint8_t NbBlockCCFile, const uint8←,_t LockCCFile)`
  Locks the CCile to prevent any RF write access.
- `int32_t BSP_NFCTAG_ReadLockCFG (const uint32_t Instance, void *const pLockCfg)`
  Reads the Cfg registers protection.
- `int32_t BSP_NFCTAG_WriteLockCFG (const uint32_t Instance, const uint8_t LockCfg)`
  Lock/Unlock the Cfg registers, to prevent any RF write access.
- `int32_t BSP_NFCTAG_PresentI2CPassword (const uint32_t Instance, const void *const PassWord)`
  Presents I2C password, to authorize the I2C writes to protected areas.
- `int32_t BSP_NFCTAG_WriteI2CPassword (const uint32_t Instance, const void *const PassWord)`
  Writes a new I2C password.

- `int32_t BSP_NFCTAG_ReadRFZxSS (const uint32_t Instance, const uint8_t Zone, void *const pRfprotZone)`
  Reads the RF Zone Security Status (defining the allowed RF accesses).
- `int32_t BSP_NFCTAG_WriteRFZxSS (const uint32_t Instance, const uint8_t Zone, const void *const Rf←, ProtZone)`
  Writes the RF Zone Security Status (defining the allowed RF accesses)
- `int32_t BSP_NFCTAG_ReadEndZonex (const uint32_t Instance, const uint8_t EndZone, uint8_t *pEndZ)`
  Reads the value of the end area address.
- `int32_t BSP_NFCTAG_WriteEndZonex (const uint32_t Instance, const uint8_t EndZone, const uint8_t EndZ)`
  Sets the end address of an area.
- `int32_t BSP_NFCTAG_InitEndZone (const uint32_t Instance)`
  Initializes the end address of the ST25DV-I2C areas with their default values (end of memory).
- `int32_t BSP_NFCTAG_CreateUserZone (const uint32_t Instance, uint16_t Zone1Length, uint16_t Zone2←, Length, uint16_t Zone3Length, uint16_t Zone4Length)`
  Creates user areas with defined lengths.
- `int32_t BSP_NFCTAG_ReadMemSize (const uint32_t Instance, void *const pSizeInfo)`
  Reads the ST25DV-I2C Memory Size.
- `int32_t BSP_NFCTAG_ReadEHMode (const uint32_t Instance, void *const pEH_mode)`
  Reads the Energy harvesting mode.
- `int32_t BSP_NFCTAG_WriteEHMode (const uint32_t Instance, const uint8_t EH_mode)`
  Sets the Energy harvesting mode.
- `int32_t BSP_NFCTAG_ReadRFMngt (const uint32_t Instance, void *const pRF_Mngt)`
  Reads the RF Management configuration.
- `int32_t BSP_NFCTAG_WriteRFMngt (const uint32_t Instance, const uint8_t Rfmngt)`
  Sets the RF Management configuration.
- `int32_t BSP_NFCTAG_GetRFDisable (const uint32_t Instance, void *const pRFDisable)`
  Reads the RFDisable register information.
- `int32_t BSP_NFCTAG_SetRFDisable (const uint32_t Instance)`
  Sets the RF Disable configuration.
- `int32_t BSP_NFCTAG_ResetRFDisable (const uint32_t Instance)`
  Resets the RF Disable configuration.
- `int32_t BSP_NFCTAG_GetRFSleep (const uint32_t Instance, void *const pRFSleep)`
  Reads the RFSleep register information.
- `int32_t BSP_NFCTAG_SetRFSleep (const uint32_t Instance)`
  Sets the RF Sleep configuration.
- `int32_t BSP_NFCTAG_ResetRFSleep (const uint32_t Instance)`
  Resets the RF Sleep configuration.
- `int32_t BSP_NFCTAG_ReadMBMode (const uint32_t Instance, void *const pMB_mode)`
  Reads the Mailbox mode.
- `int32_t BSP_NFCTAG_WriteMBMode (const uint32_t Instance, const uint8_t MB_mode)`
  Sets the Mailbox mode.
- `int32_t BSP_NFCTAG_ReadMBWDG (const uint32_t Instance, uint8_t *const pWdgDelay)`
  Reads the Mailbox watchdog duration coefficient.
- `int32_t BSP_NFCTAG_WriteMBWDG (const uint32_t Instance, const uint8_t WdgDelay)`
  Writes the Mailbox watchdog coefficient delay.

- `int32_t BSP_NFCTAG_ReadMailboxData (const uint32_t Instance, uint8_t *const pData, const uint16_t Tar←, Addr, const uint16_t NbByte)`
  Reads N bytes of data from the Mailbox, starting at the specified byte offset.
- `int32_t BSP_NFCTAG_WriteMailboxData (const uint32_t Instance, const uint8_t *const pData, const uint16_t NbByte)`
  Writes N bytes of data in the Mailbox, starting from first Mailbox Address.
- `int32_t BSP_NFCTAG_ReadMailboxRegister (const uint32_t Instance, uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Reads N bytes from the mailbox registers, starting at the specified I2C address.
- `int32_t BSP_NFCTAG_WriteMailboxRegister (const uint32_t Instance, const uint8_t *const pData, const uint16_t TarAddr, const uint16_t NbByte)`
  Writes N bytes to the specified mailbox register.
- `int32_t BSP_NFCTAG_ReadI2CSecuritySession_Dyn (const uint32_t Instance, void *const pSession)`
  Reads the status of the security session open register.
- `int32_t BSP_NFCTAG_ReadITSTStatus_Dyn (const uint32_t Instance, uint8_t *const pITStatus)`
  Reads the IT status register from the ST25DV-I2C.
- `int32_t BSP_NFCTAG_ReadGPO_Dyn (const uint32_t Instance, uint8_t *GPOConfig)`
  Read value of dynamic GPO register configuration.
- `int32_t BSP_NFCTAG_GetGPO_en_Dyn (const uint32_t Instance, void *const pGPO_en)`
  Get dynamique GPO enable status.
- `int32_t BSP_NFCTAG_SetGPO_en_Dyn (const uint32_t Instance)`
  Set dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ResetGPO_en_Dyn (const uint32_t Instance)`
  Reset dynamique GPO enable configuration.
- `int32_t BSP_NFCTAG_ReadEHCtrl_Dyn (const uint32_t Instance, void *const pEH_CTRL)`
  Read value of dynamic EH Ctrl register configuration.
- `int32_t BSP_NFCTAG_GetEHENMode_Dyn (const uint32_t Instance, void *const pEH_Val)`
  Reads the Energy Harvesting dynamic status.
- `int32_t BSP_NFCTAG_SetEHENMode_Dyn (const uint32_t Instance)`
  Dynamically sets the Energy Harvesting mode.
- `int32_t BSP_NFCTAG_ResetEHENMode_Dyn (const uint32_t Instance)`
  Dynamically unsets the Energy Harvesting mode.
- `int32_t BSP_NFCTAG_GetEHON_Dyn (const uint32_t Instance, void *const pEHON)`
  Reads the EH_ON status from the EH_CTRL_DYN register.
- `int32_t BSP_NFCTAG_GetRFField_Dyn (const uint32_t Instance, void *const pRF_Field)`
  Checks if RF Field is present in front of the ST25DV-I2C.
- `int32_t BSP_NFCTAG_GetVCC_Dyn (const uint32_t Instance, void *const pVCC)`
  Check if VCC is supplying the ST25DV-I2C.
- `int32_t BSP_NFCTAG_ReadRFMngt_Dyn (const uint32_t Instance, void *const pRF_Mngt)`
  Read value of dynamic RF Management configuration.
- `int32_t BSP_NFCTAG_WriteRFMngt_Dyn (const uint32_t Instance, const uint8_t RF_Mngt)`
  Writes a value to the RF Management dynamic register.

- `int32_t BSP_NFCTAG_GetRFDisable_Dyn (const uint32_t Instance, void *const pRFDisable)`
  Reads the RFDisable dynamic register information.
- `int32_t BSP_NFCTAG_SetRFDisable_Dyn (const uint32_t Instance)`
  Sets the RF Disable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetRFDisable_Dyn (const uint32_t Instance)`
  Unsets the RF Disable dynamic configuration.
- `int32_t BSP_NFCTAG_GetRFSleep_Dyn (const uint32_t Instance, void *const pRFSleep)`
  Reads the RFSleep dynamic register information.
- `int32_t BSP_NFCTAG_SetRFSleep_Dyn (const uint32_t Instance)`
  Sets the RF Sleep dynamic configuration.
- `int32_t BSP_NFCTAG_ResetRFSleep_Dyn (const uint32_t Instance)`
  Unsets the RF Sleep dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBCtrl_Dyn (const uint32_t Instance, void *const pCtrlStatus)`
  Reads the Mailbox ctrl dynamic register.
- `int32_t BSP_NFCTAG_GetMBEN_Dyn (const uint32_t Instance, void *const pMBEN)`
  Reads the Mailbox Enable dynamic configuration.
- `int32_t BSP_NFCTAG_SetMBEN_Dyn (const uint32_t Instance)`
  Sets the Mailbox Enable dynamic configuration.
- `int32_t BSP_NFCTAG_ResetMBEN_Dyn (const uint32_t Instance)`
  Unsets the Mailbox Enable dynamic configuration.
- `int32_t BSP_NFCTAG_ReadMBLength_Dyn (const uint32_t Instance, uint8_t *const pMBLength)`
  Reads the Mailbox message length dynamic register.

9.7.1 Detailed description

This file contains definitions for the st25_discovery_nfctag.c specific functions.

9.8 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/commonfunc.h file reference

Data structures

- `struct IT_GPO_STATUS`
  ST25DV-I2C GPO interrupt status structure.

Macros

- `#define ST25_RETRY(cmd)`
  Iterate ST25DV-I2C command depending on the command return status.

Functions

- `ST25DV_I2CSSO_STATUS PresentPasswd (const bool passwd)`
  Present password to the ST25DV-I2C to open or close an i2c session.
- `int32_t InitITGPOMode (const uint16_t ITConfig)`
  Enable & initialize the GPO interrupt.
- `void DeInitITGPOMode (void)`
  Disable the GPO interrupt.

- `void ManageGPO (IT_GPO_STATUS *const gpo)`
  Reads the GPO interrupt source.
- `void st25_error (int32_t status)`
  Display an error message depending on the return value of a NFCTAG driver function.

9.8.1 Detailed description

Header for commonfunc.c module

9.9 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/ff.h file reference

Simple file system api header.

Data structures

- struct FIL
  File object structure.

Macros

- `#define FA_READ 0x01`
  File access control: Read-Only (no other value supported)

Enumerations

- enum FRESULT { FR_OK = 0 , FR_DENIED = 7 , FR_INVALID_OBJECT = 9 }
  Function return codes.

Functions

- `FRESULTf_open (FIL *fp, const char *start, uint8_t mode)`
  Open a stream of data from an address in memory.
- `FRESULTf_close (FIL *fp)`
  Close a stream of data from an address in memory.
- `FRESULTf_read (FIL *fp, void *buff, uint32_t btr, uint32_t *br)`
  Reads bytes from a stream of data.
- `FRESULTf_write (FIL *fp, const void *buff, uint32_t btw, uint32_t *bw)`
  Do nothing.

9.9.1 Detailed description

Simple file system api header.

9.10 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/fw_ command.h file reference

Header file for fw_command.c.

Functions

- `uint32_t COMMAND_EraseFlash (const uint32_t Address)`
  Command To erase specific Flash memory area.
- `uint32_t Command_WriteBufferToFlash (const uint32_t StartAddress, const uint32_t offset, const uint8_←, t *const pData, const uint32_t size)`
  Writes buffer to Flash memory.

- `void COMMAND_Jump (void)`
  Jump to user program.

9.10.1 Detailed description

Header file for fw_command.c.

9.11 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/jpeg_decode.h file reference

Macros

- `#define IS_JPEG (ptr) ((ptr[0] == 0xFF) && (ptr[1] == 0xD8))`
  Macro checking if a pointer points to the beginning of a JPEG picture.

Functions

- `uint32_t jpeg_decode (const char *jpeg, uint8_t(*callback)(uint8_t *, uint32_t))`
  Decode a jpeg formatted picture calling the Cube libJPEG middleware.
- `void jpeg_getsize (const char *jpeg, uint32_t *Width, uint32_t *Height)`
  Get the geometry of a JPEG picture.
- `uint32_t jpeg_GetBufferSize (uint8_t *jpeg)`
  Gut not decompressed Jpeg buffer size.

9.11.1 Detailed description

Header for jpeg_decode.c module

9.12 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/mailbox.h file reference

Functions

Header for mailbox.c module

- `void FTMManagement (void)`
  Manage data exchange protocol with reader through Mailbox.

9.12.1 Detailed description

Header for mailbox.c module.

9.13 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/mailboxfunc.h file reference

Header for mailboxfunc.c module

Data structures

- struct MB_HEADER_T
  Mailbox global header information structure definition.
- struct MB_STOPWATCH_T

Functions

- int32_t InitMailBoxMode (void)
  Initializes the Mailbox mode, Enables Mailbox and disables Mailbox Watchdog.
- int32_t DeInitMailBoxMode (void)
  DeInitializes the Mailbox mode, disables mailbox mode.
- int32_t InitGPOforMailBoxMode (void)
  Initializes the Mailbox mode, Enables Mailbox and disables Mailbox Watchdog.
- int32_t WriteMailBoxMsg (const uint8_t ∗const pData, const uint16_t NbBytes)
  Writes message in Mailbox.
- int32_t ReadCompleteMailBoxMsg (uint8_t ∗const pData, uint16_t ∗const pLength)
  Reads entire Mailbox Message from the tag.
- int32_t ReadFragmentMailBoxMsg (uint8_t ∗const pData, const uint8_t Offset, const uint16_t NbBytes)
  Reads part of Mailbox Message from the tag.
- void MBDecodeHeader (const uint8_t ∗const pData, MB_HEADER_T ∗const mb_header)
  Extracts global information from header in Fast transfer mode protocol.
- void PrepareMBMsg (uint8_t ∗const pData, const MB_HEADER_T ∗const mb_header)
  Prepares header message to send to Mailbox.
- bool SendMBData (MB_HEADER_T ∗const mb_header, const uint8_t nbretry)
  Prepares and writes frame in Mailbox.

9.13.1 Detailed description

Header for mailboxfunc.c module.

9.14 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/main.h file reference

Functions

- void SplashScreen (void)
  Display splash screen.
- void MenuAbout (void)
  Display "_about_" screen.

9.14.1 Detailed description

Header for main.c module.

9.15 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/Menu_definition.h file reference

Header for Menu_definition.c.

Functions

- `void Menu_Start (void)`
  Starts the main loop for the demo menu.

9.15.1 Detailed description

Header for Menu_definition.c.

9.16 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/ndef_demo.h file reference

This file defines the api for the NDEF demo.

Functions

- `void NDEF_DEMO_Init_Tag (void)`
  This function initializes the NFC Tag to perform the NDEF demos.
- `void NDEF_DEMO_Write_URI_URL (void)`
  This function writes an URL to the Tag.
- `void NDEF_DEMO_Write_URI_Tel (void)`
  This function writes a phone number to the Tag.
- `void NDEF_DEMO_Read_URI (void)`
  This function reads and URI from the Tag and displays its content.
- `void NDEF_DEMO_Read_SMS (void)`
  This function reads a SMS from the Tag and displays it.
- `void NDEF_DEMO_Write_SMS (void)`
  This function writes a SMS to the Tag.
- `void NDEF_DEMO_Read_Email (void)`
  This function reads an Email from the Tag and displays its content.
- `void NDEF_DEMO_Write_Email (void)`
  This function writes an Email to the Tag.
- `void NDEF_DEMO_Read_Vcard (void)`
  This function reads a vCard record from the Tag and display its content.
- `void NDEF_DEMO_Write_Vcard (void)`
  This function writes a vCard record to the Tag.
- `void NDEF_DEMO_Write_Picture_Vcard (void)`
  This function writes a vCards with an embedded picture to the Tag.
- `void NDEF_DEMO_Read_Geo (void)`
  This function reads a Geolocation record from the Tag and displays it content.
- `void NDEF_DEMO_Write_Geo (void)`
  This function writes a Geolocation record to the Tag.
- `void NDEF_DEMO_Read_MyAPP (void)`
  This function reads a MyApp record from the Tag and starts the associated demo.
- `void NDEF_DEMO_Write_AAR (void)`
  This function writes an AAR record (selecting the ST NFC application) to the Tag.
- `void NDEF_DEMO_MultiRecord_With_AAR (void)`
  This function adds an AAR record (selecting the ST NFC application) to an existing NDEF file on the Tag.
- `void NDEF_DEMO_BLE_ChangeDeviceAddr (void)`
  This function change the Bluetooth Low Energy Device Address (to be used when the module is paired with a nearby smartphone).
- `void NDEF_DEMO_Write_BLE_OOB (void)`
  This function writes a Bluetooth Low Energy OOB record to the Tag and starts the BLE module, waiting for a connec- tion to occur.
- `void NDEF_DEMO_Read_Bluetooth_OOB (void)`
  This function reads a BLE OOB record from the Tag and displays it content.
- `void NDEF_DEMO_Write_Wifi_OOB (void)`
  This function writes a Wifi OOB record to the Tag and starts the Wifi module, waiting for a connection to occur.
- `void NDEF_DEMO_Read_Wifi_OOB (void)`

9.16.1 Detailed description

This file defines the api for the NDEF demo.

9.17 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/st25dv_features_demo.h file reference

This file defines the api for the ST25DV-I2C features demo.

Functions

- `void ST25DV_DEMO_Led2_Pwm (void)`
  This function switches the LED2 on/off as a PWM.
- `void ST25DV_DEMO_EnergyHarvesting (void)`
  This functions runs the Energy Harvesting demo with digital potentiometer set to 240 Ohms.
- `void ST25DV_DEMO_GPO (void)`
  This function runs the GPO interrupts demos.
- `void ST25DV_DEMO_RF_Off (void)`
  This function sets the ST25DV-I2C RF off.
- `void ST25DV_DEMO_RF_Sleep (void)`
  This function sets the ST25DV-I2C RF to sleep state.
- `void ST25DV_DEMO_Low_Power_Down (void)`
  This function sets the ST25DV-I2C in Low Power Mode.
- `void ST25DV_DEMO_Memory_Mapping_Password (void)`
  This function configures the ST25DV-I2C memory mapping and sets a password protection.
- `void ST25DVxxKC_DEMO_CompareWriteEEPROM (void)`
  This function emulate an ST25DVxxK EEPROM write and compare it to the ST25DVxxKC write.

9.17.1 Detailed description

This file defines the api for the ST25DV-I2C features demo.

9.17.2 Function documentation

ST25DV_DEMO_Led2_Pwm()

```
void ST25DV_DEMO_Led2_Pwm ( void )
```

This function switches the LED2 on/off as a PWM.

This function has to be regularly called from a Tick routine. It emulates the effect of a LED powered by the Energy Harvesting from the ST25DV-I2C. It relies on local variables: led_pwm_value, led_pwm_period.

9.18 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/stm32l4xx_hal_conf.h file reference

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

Macros

- `#define HAL_MODULE_ENABLED`
  This is the list of modules to be used in the HAL driver.
- `#define HSE_VALUE ((uint32_t)8000000)`
  Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

- `#define HSE_STARTUP_TIMEOUT ((uint32_t)100)`
- `#define MSI_VALUE ((uint32_t)4000000)`

  Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset.
- `#define HSI_VALUE ((uint32_t)16000000)`

  Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- `#define LSI_VALUE ((uint32_t)32000)`

  Internal Low Speed oscillator (LSI) value.
- `#define LSE_VALUE ((uint32_t)32768)`

  External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.
- `#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)`
- `#define EXTERNAL_SAI1_CLOCK_VALUE ((uint32_t)48000)`

  External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.
- `#define EXTERNAL_SAI2_CLOCK_VALUE ((uint32_t)48000)`

  External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 & SAI2 clock source frequency.
- `#define VDD_VALUE ((uint32_t)3300U)`

  This is the HAL system configuration section.
- `#define TICK_INT_PRIORITY ((uint32_t)0x0FU)`
- `#define USE_SPI_CRC 1U`

  Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- `#define assert_param(expr) ((void)0U)`

  Include module's header file.

9.18.1 Detailed description

HAL configuration template file. This file should be copied to the application folder and renamed to stm32l4xx_hal_conf.h.

9.18.2 Macro definition documentation

EXTERNAL_SAI1_CLOCK_VALUE

```
#define EXTERNAL_SAI1_CLOCK_VALUE ((uint32_t)48000)
```

External clock source for SAI1 peripheral This value is used by the RCC HAL module to compute the SAI1 and SAI2 clock source frequency.

Value of the SAI1 External clock source in Hz

EXTERNAL_SAI2_CLOCK_VALUE

```
#define EXTERNAL_SAI2_CLOCK_VALUE ((uint32_t)48000)
```

External clock source for SAI2 peripheral This value is used by the RCC HAL module to compute the SAI1 and SAI2 clock source frequency.

Value of the SAI2 External clock source in Hz

HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100)
```

Time out for HSE start up, in ms

HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz.

HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz.

LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up, in ms.

LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

External Low Speed oscillator (LSE) value. This value is used by the UART, RTC HAL module to compute the system frequency.

Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External oscillator in Hz

LSI_VALUE

```
#define LSI_VALUE ((uint32_t)32000) Internal Low Speed oscillator (LSI) value. LSI
Typical Value in Hz
```

MSI_VALUE

```
#define MSI_VALUE ((uint32_t)4000000)
```

Internal Multiple Speed oscillator (MSI) default value. This value is the default MSI range value after Reset. Value of the Internal oscillator in Hz

TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)0x0FU)
```

tick interrupt priority

VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300U) This is the HAL system configuration section.
Value of VDD in mv
```

9.19 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/stm32l4xx_it.h file reference

This file contains the headers of the interrupt handlers.

Functions

- `void NMI_Handler (void)`
  This function handles NMI exception.
- `void HardFault_Handler (void)`
  This function handles Hard Fault exception.
- `void MemManage_Handler (void)`
  This function handles Memory Manage exception.

- `void BusFault_Handler (void)`
  This function handles Bus Fault exception.
- `void UsageFault_Handler (void)`
  This function handles Usage Fault exception.
- `void SVC_Handler (void)`
  This function handles SVCall exception.
- `void DebugMon_Handler (void)`
  This function handles Debug Monitor exception.
- `void PendSV_Handler (void)`
  This function handles PendSVC exception.
- `void SysTick_Handler (void)`
  This function handles SysTick Handler.

Variables

- `volatile uint8_t GPO_Activated`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.

9.19.1 Detailed description

This file contains the headers of the interrupt handlers.

9.20 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Inc/wifi,_interface.h file reference

Header file for X-CUBE-WIFI1 API.

Data structures

- struct wifi_security
- struct wifi_scan
- struct wifi_config

Functions

- `Void ind_wifi_ap_client_joined (uint8_t *client_mac_address)`
  This function is the callback for the Wifi Middleware, called when a client has joined.

9.20.1 Detailed description

Header file for X-CUBE-WIFI1 API.

9.21 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/commonfunc.c file reference

Common functions for the ST25DV-I2C management.

Functions

- `ST25DV_I2CSSO_STATUS PresentPasswd (const bool passwd)`
  Present password to the ST25DV-I2C to open or close an i2c session.
- `int32_t InitITGPOMode (const uint16_t ITConfig)`
  Enable & initialize the GPO interrupt.
- `void DeInitITGPOMode (void)`
  Disable the GPO interrupt.

- `void ManageGPO (IT_GPO_STATUS *const gpo)`
  Reads the GPO interrupt source.
- `void st25_error (int32_t status)`
  Display an error message depending on the return value of a NFCTAG driver function.

Variables

- `uint8_t GPO_Activated`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.

9.21.1 Detailed description

Common functions for the ST25DV-I2C management.

9.22 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/fs_api.c file reference

Simple filesystem api implementation.

Functions

- `FRESULT f_open (FIL *fp, const char *start, uint8_t mode)`
  Open a stream of data from an address in memory.
- `FRESULT f_close (FIL *fp)`
  Close a stream of data from an address in memory.
- `FRESULT f_read (FIL *fp, void *buff, uint32_t btr, uint32_t *br)`
  Reads bytes from a stream of data.
- `FRESULT f_write (FIL *fp, const void *buff, uint32_t btw, uint32_t *bw)`
  Do nothing.

9.22.1 Detailed description

Simple filesystem api implementation.

9.23 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/fw_command.c file reference

This file provides all the flash programming command functions.

Functions

- uint32_t COMMAND_EraseFlash (const uint32_t Address)
  Command To erase specific Flash memory area.
- uint32_t Command_WriteBufferToFlash (const uint32_t StartAddress, const uint32_t offset, const uint8_←, t *const pData, const uint32_t size)
  Writes buffer to Flash memory.
- void COMMAND_Jump (void)
  Jump to user program.

9.23.1 Detailed description

This file provides all the flash programming command functions.

9.24 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/jpeg_decode.c file reference

This file contains the JPEG decompressing methods.

Functions

- `void jpeg_decode_exit (j_common_ptr cinfo)`
  Callback for the libJPEG, executed when an unrecoverable error occured.
- `uint32_t jpeg_decode (const char *jpeg, uint8_t(*callback)(uint8_t *, uint32_t))`
  Decode a jpeg formatted picture calling the Cube libJPEG middleware.
- `void jpeg_getsize (const char *jpeg, uint32_t *Width, uint32_t *Height)`
  Get the geometry of a JPEG picture.
- `uint32_t jpeg_GetBufferSize (uint8_t *jpeg)`
  Gut not decompressed Jpeg buffer size.

9.24.1 Detailed description

This file contains the JPEG decompressing methods.

9.25 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/ndef_demo.c file reference

This file provides functions to execute NDEF demos.

Macros

- `#define NDEF_DEMO_CC3_COMPLIANT_VALUE ((uint8_t)0x5)`
- `#define NDEF_DEMO_PHONE_NUMBER "+33612345678"`
- `#define NDEF_DEMO_URL "st.com/st25-demo"`
- `#define NDEF_DEMO_BLE_MODULE_VERSION ((uint8_t)0x31)`
- `#define NDEF_DEMO_BLE_PIN ((uint32_t)123456)`
- `#define NDEF_DEMO_BLE_NAME "HID"`
- `#define NDEF_DEMO_BLE_ADDR_BIG_ENDIAN {0x02, 0x80, 0xE1, 0x00, 0x34, 0x12}`
- `#define NDEF_DEMO_SET_WIFI_SSID (str)`
- `#define NDEF_DEMO_WIFI_CHANNEL_NUM ((uint8_t)6)`
- `#define NDEF_DEMO_WIFI_UART_BAUDRATE ((uint32_t)115200)`

Functions

- `void Hid_Profile_Application (void)`
  Starts the HID application, using touchscreen detection to send mouse input reports and user button as mouse button.
- `void NFCMEM_IO_MemWriteCompleted_Callback (uint32_t size)`
  Callback from NFCMEM_IO_MemWrite to manage the progress bar when writing a big NDEF.
- `void ind_wifi_ap_client_joined (uint8_t *mac)`
  This function is the callback for the Wifi Middleware, called when a client has joined.
- `void NDEF_DEMO_Init_Tag (void)`
  This function initializes the NFC Tag to perform the NDEF demos.
- `void NDEF_DEMO_Write_URI_URL (void)`
  This function writes an URL to the Tag.
- `void NDEF_DEMO_Write_URI_Tel (void)`
  This function writes a phone number to the Tag.

- `void NDEF_DEMO_Read_URI (void)`
  This function reads and URI from the Tag and displays its content.
- `void NDEF_DEMO_Read_SMS (void)`
  This function reads a SMS from the Tag and displays it.
- `void NDEF_DEMO_Write_SMS (void)`
  This function writes a SMS to the Tag.
- `void NDEF_DEMO_Read_Email (void)`
  This function reads an Email from the Tag and displays its content.
- `void NDEF_DEMO_Write_Email (void)`
  This function writes an Email to the Tag.
- `void NDEF_DEMO_Read_Vcard (void)`
  This function reads a vCard record from the Tag and display its content.
- `void NDEF_DEMO_Write_NoPicture_Vcard (void)`
  This function writes a small vCard record to the Tag.
- `void NDEF_DEMO_Write_Picture_Vcard (void)`
  This function writes a vCards with an embedded picture to the Tag.
- `void NDEF_DEMO_Write_Vcard (void)`
  This function writes a vCard record to the Tag.
- `void NDEF_DEMO_Read_Geo (void)`
  This function reads a Geolocation record from the Tag and displays it content.
- `void NDEF_DEMO_Write_Geo (void)`
  This function writes a Geolocation record to the Tag.
- `void NDEF_DEMO_Read_MyAPP (void)`
  This function reads a MyApp record from the Tag and starts the associated demo.
- `void NDEF_DEMO_Write_AAR (void)`
  This function writes an AAR record (selecting the ST NFC application) to the Tag.
- `void NDEF_DEMO_MultiRecord_With_AAR (void)`
  This function adds an AAR record (selecting the ST NFC application) to an existing NDEF file on the Tag.
- `void NDEF_DEMO_BLE_ChangeDeviceAddr (void)`
  This function change the Bluetooth Low Energy Device Address (to be used when the module is paired with a nearby smartphone).
- `void NDEF_DEMO_Write_BLE_OOB (void)`
  This function writes a Bluetooth Low Energy OOB record to the Tag and starts the BLE module, waiting for a connec- tion to occur.
- `void NDEF_DEMO_Read_Bluetooth_OOB (void)`
  This function reads a BLE OOB record from the Tag and displays it content.
- `void NDEF_DEMO_Write_Wifi_OOB (void)`
  This function writes a Wifi OOB record to the Tag and starts the Wifi module, waiting for a connection to occur.
- `void NDEF_DEMO_Read_Wifi_OOB (void)`
  This function reads a Wifi OOB record from the Tag and displays it's content.
- `void NDEF_DEMO_Write_empty_NDEF (void)`
  This function writes an empty NDEF message.
- `void NDEF_DEMO_Erase_CCFile (void)`
  This function writes 0xFF to the 4 first bytes in EEPROM.
- `void NDEF_DEMO_Clear_Eeprom (void)`
  This function writes 0xFF to the entire EEPROM.

Variables

- volatile uint8_t WifiConnected = 0
- volatile char WifiClientMac [20]
- TIM_HandleTypeDef TimHandle
  Wifi timer handle.

9.25.1 Detailed description

This file provides functions to execute NDEF demos.
The demos covers the following usecases:

- URI NDEF records: URL & Phone.
- SMS NDEF record.
- Email NDEF record.
- vCard NDEF record (with or without an embedded picture).
- Geolocation NDEF record.
- MyApp custom NDEF record.
- Multi NDEF record with AAR.
- Bluetooth Low Energy OOB NDEF record.
- Wifi OOB NDEF record.

9.25.2 Macro definition documentation

NDEF_DEMO_BLE_ADDR_BIG_ENDIAN

```
#define NDEF_DEMO_BLE_ADDR_BIG_ENDIAN {0x02, 0x80, 0xE1, 0x00, 0x34, 0x12}
```

Bluetooth Device Address big endian (default value, partly overriden to uniquify the DeviceAddress)

NDEF_DEMO_BLE_MODULE_VERSION

```
#define NDEF_DEMO_BLE_MODULE_VERSION ((uint8_t)0x31)
```

Bluetooth expected HW module version

NDEF_DEMO_BLE_NAME

```
#define NDEF_DEMO_BLE_NAME "HID"
```

Bluetooth module name

NDEF_DEMO_BLE_PIN

```
#define NDEF_DEMO_BLE_PIN ((uint32_t)123456)
```

Bluetooth pin code (not used in the demo)

NDEF_DEMO_CC3_COMPLIANT_VALUE

```
#define NDEF_DEMO_CC3_COMPLIANT_VALUE ((uint8_t)0x5)
```

Type5 Capabillity Container byte3 value for android compliancy

NDEF_DEMO_PHONE_NUMBER

```
#define NDEF_DEMO_PHONE_NUMBER "+33612345678"
```

Dummy phone number used in the demo

NDEF_DEMO_SET_WIFI_SSID

```
#define NDEF_DEMO_SET_WIFI_SSID ( str )
```

Wifi module SSID

NDEF_DEMO_URL

```
#define NDEF_DEMO_URL "st.com/st25-demo"
```

ST URL to ST25 products page

NDEF_DEMO_WIFI_CHANNEL_NUM

#define NDEF_DEMO_WIFI_CHANNEL_NUM ((uint8_t)6)

Wifi module number of channels

NDEF_DEMO_WIFI_UART_BAUDRATE

```
#define NDEF_DEMO_WIFI_UART_BAUDRATE ((uint32_t)115200)
```

Wifi module baudrate

9.25.3 Function documentation

NFCMEM_IO_MemWriteCompleted_Callback()

```
void NFCMEM_IO_MemWriteCompleted_Callback ( uint32_t size )
```

Callback from NFCMEM_IO_MemWrite to manage the progress bar when writing a big NDEF.

Table 606. NFCMEM_IO_MemWriteCompleted_Callback() parameters

size	: number of byte written in memory.

9.25.4 Variable documentation

WifiClientMac

```
volatile char WifiClientMac[20]
```

Wifi client mac address

WifiConnected

```
volatile uint8_t WifiConnected = 0
```

Wifi conexion state

9.26 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/st25dv_features_demo.c file reference

This file provides functions to execute ST25DV demonstration boards.

Data structures

- `struct EH_container`

Macros

- `#define DIFF(a, b) ((a)>(b)?(a)-(b):(b)-(a))`
- `#define ADC_RESOLUTION ((uint16_t) 4096)`
- `#define RMEAS_K ((float)0.0046)`
- `#define EH_DISPLAY_TEMPO_MS ((uint8_t) 50)`
- `#define EH_REFRESH_FILTER ((uint8_t) 30)`
- `#define EH_0_FILTER ((uint8_t) 150)`
- `#define EH_LED_PWM_PER ((uint16_t) 20)`

- `#define EH_LED_PWM_DIV ((uint16_t) (ADC_RESOLUTION - EH_0_FILTER) / EH_LED_PWM_PER)`
- `#define NDEF_DEMO_URL "st.com/st25-demo"`
- `#define PREV_VAL_NB ((uint8_t) 50)`
- `#define INPUT_NB ((uint8_t) 3)`

Functions

- `void ST25DV_DEMO_Led2_Pwm (void)`
  This function switches the LED2 on/off as a PWM.
- `void ST25DV_DEMO_EnergyHarvesting (void)`
  This functions runs the Energy Harvesting demo with digital potentiometer set to 240 Ohms.
- `void ST25DV_DEMO_GPO (void)`
  This function runs the GPO interrupts demos.
- `void ST25DV_DEMO_RF_Off (void)`
  This function sets the ST25DV-I2C RF off.
- `void ST25DV_DEMO_RF_Sleep (void)`
  This function sets the ST25DV-I2C RF to sleep state.
- `void ST25DV_DEMO_Low_Power_Down (void)`
  This function sets the ST25DV-I2C in Low Power Mode.
- `void ST25DV_DEMO_Memory_Mapping_Password (void)`
  This function configures the ST25DV-I2C memory mapping and sets a password protection.
- `void ST25DVxxKC_DEMO_CompareWriteEEPROM (void)`
  This function emulate an ST25DVxxK EEPROM write and compare it to the ST25DVxxKC write.

Variables

- `volatile uint8_t GPO_Activated`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.

Detailed description

This file provides functions to execute ST25DV demonstration boards..
The demos covers the following usecases:

- Energy harvesting.
- GPO interrupts from RF.
- ST25DV states: RF off, RF sleep, Low power down.
- Multi-area and Password protection.
- EEPROM writing time comparison between ST25DVxxK (emulated) and ST25DVxxKC.

9.27 Macro definition documentation

ADC_RESOLUTION

```
#define ADC_RESOLUTION ((uint16_t) 4096)
```

Analog to digital converter max value

DIFF

```
#define DIFF( a, b ) ((a)>(b)?(a)-(b):(b)-(a))
```

Compute the positive difference between two numbers

EH_0_FILTER

```
#define EH_0_FILTER ((uint8_t) 150)
```

Min Energy Harvesting value to display a non-zero value

EH_DISPLAY_TEMPO_MS

```
#define EH_DISPLAY_TEMPO_MS ((uint8_t) 50)
```

Min delay between 2 Energy Harvesting measures

EH_LED_PWM_DIV

```
#define EH_LED_PWM_DIV ((uint16_t) (ADC_RESOLUTION - EH_0_FILTER) / EH_LED_PWM_PER)
```

Energy Harvesting LED PWM divider

EH_LED_PWM_PER

```
#define EH_LED_PWM_PER ((uint16_t) 20)
```

Energy Harvesting LED PWM period

EH_REFRESH_FILTER

```
#define EH_REFRESH_FILTER ((uint8_t) 30)
```

Min Energy Harvesting variation causing a change in the display

INPUT_NB

```
#define INPUT_NB ((uint8_t) 3)
```

Number of values measured for the Energy Harvesting, on which to compute the average.

NDEF_DEMO_URL

```
#define NDEF_DEMO_URL "st.com/st25-demo"
```

ST URL for ST25 products

PREV_VAL_NB

```
#define PREV_VAL_NB ((uint8_t) 50)
```

Number of previous values stored to compute the average value of the Energy Harvesting

RMEAS_K

```
#define RMEAS_K ((float)0.0046)
```

Min resistance resistance on the board Value of the resistor used for current measurement. (Could be 4.3 Ohm or formerly 2 kOhms)

9.27.1 Function documentation

ST25DV_DEMO_Led2_Pwm()

```
void ST25DV_DEMO_Led2_Pwm ( void )
```

This function switches the LED2 on/off as a PWM.

This function has to be regularly called from a Tick routine. It emulates the effect of a LED powered by the Energy Harvesting from the ST25DV-I2C. It relies on local variables: led_pwm_value, led_pwm_period

9.28 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/stm32l4xx_hal_msp.c file reference

HAL MSP module.

Functions

- `void HAL_MspInit (void)`
  Initializes the Global MSP.
- `void HAL_MspDeInit (void)`
  DeInitializes the Global MSP.
- `void HAL_SPI_MspInit (SPI_HandleTypeDef *hspi)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for SPI.
- `void HAL_SPI_MspDeInit (SPI_HandleTypeDef *hspi)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_UART_MspInit (UART_HandleTypeDef *huart)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_UART_MspDeInit (UART_HandleTypeDef *huart)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for UART.
- `void HAL_DMA_MspInit (void)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for DMA.
- `void HAL_DMA_MspDeInit (void)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for DMA.
- `void HAL_CRC_MspInit (CRC_HandleTypeDef *hcrc)`
  Initializes the low level hardware: GPIO, CLOCK, NVIC for CRC.
- `void HAL_CRC_MspDeInit (CRC_HandleTypeDef *hcrc)`
  DeInitializes the low level hardware: GPIO, CLOCK, NVIC for CRC.
- `void SystemClock_Config (void)`
  System Clock Configuration The system Clock is configured as follow : System Clock source = PLL (HSE) SYSCLK(Hz) = 80000000 HCLK(Hz) = 80000000 AHB Prescaler = 1 APB1 Prescaler = 1 APB2 Prescaler = 1 HSE Frequency(Hz) = 8000000 PLL_M = 1 PLL_N = 20 PLL_P = 7 PLL_Q = 4 PLL_R = 2 Flash Latency(WS) = 4.
- `void MX_GPIO_Init (void)`

9.29 Projects/ST25DV-Discovery/Demonstrations/ST25DVDemo/Src/stm32l4xx_it.c file reference

Main Interrupt Service Routines.

Functions

- `void NMI_Handler (void)`
  This function handles NMI exception.
- `void HardFault_Handler (void)`
  This function handles Hard Fault exception.
- `void MemManage_Handler (void)`
  This function handles Memory Manage exception.
- `void BusFault_Handler (void)`
  This function handles Bus Fault exception.
- `void UsageFault_Handler (void)`
  This function handles Usage Fault exception.
- `void SVC_Handler (void)`
  This function handles SVCall exception.
- `void DebugMon_Handler (void)`
  This function handles Debug Monitor exception.

- `void PendSV_Handler (void)`
  This function handles PendSVC exception.
- `void SysTick_Handler (void)`
  This function handles SysTick Handler.
- `void TIMx_IRQHandler (void)`
  This function handles TIM interrupt request.
- `void TIMp_IRQHandler (void)`
  This function handles TIM interrupt request.
- `void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)`
  Period elapsed callback in non blocking mode This timer is used for calling back User registered functions with information.
- `void BNRG_SPI_EXTI_IRQHandler (void)`
  This function handles External line interrupt request for BlueNRG bluetooth module.
- `void USART3_IRQHandler (void)`
  This function handles USART3 interrupts for the Wifi module.
- `void NFCMEM_GPO_EXTIHandler (void)`
  This function handles External line 0 interrupt request.
- `void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)`
  This function handles callback from external line interrupt request.

Variables

- `volatile uint32_t ms_counter = 0`
  Current millisecond count.
- `volatile uint8_t GPO_Activated = 0`
  Polling variable for the ST25DV GPO interrupt, updated from GPO interrupt callback.
- `TIM_HandleTypeDef TimHandle`
  Timer handle from st25_spwf_wifi.c.
- `TIM_HandleTypeDef PushTimHandle`
  Timer handle from st25_spwf_wifi.c.
- `UART_HandleTypeDef UartWiFiHandle`
  UART handle from wifi_module.c.

9.29.1 Detailed description

Main Interrupt Service Routines.

Revision history

Table 607. **Document revision history**

Date	Version	Changes
17-Dec-2021	1	Initial release.

Contents

List of tables

List of figures