

# BLUEBOXLib

ANSI C Function Library

**BLUEBOX**  
RFid System

Up to library release 8.41.0

## Preface

iDTRONIC GmbH (iDTRONIC) reserves the right to make changes to its products or services or to discontinue any product or service at any time without notice. iDTRONIC provides customer assistance in various technical areas, but does not have full access to data concerning the use and applications of customer's products. Therefore, iDTRONIC assumes no liability and is not responsible for customer applications or product or software design or performance relating to systems or applications incorporating iDTRONIC products. In addition, iDTRONIC assumes no liability and is not responsible for infringement of patents and/or any other intellectual or industrial property rights of third parties, which may result from assistance provided by iDTRONIC. iDTRONIC products are not designed, intended, authorized or warranted to be suitable for life support applications or any other life critical applications that could involve potential risk of death, personal injury or severe property or environmental damage. With the edition of this document, all previous editions become void. Indications made in this manual may be changed without previous notice. Composition of the information in this manual has been done to the best of our knowledge. iDTRONIC does not guarantee the correctness and completeness of the details given in this manual and may not be held liable for damages ensuing from incorrect or incomplete information. Since, despite all our efforts, errors may not be completely avoided, we are always grateful for your useful tips. The installation instructions given in this manual are based on advantageous boundary conditions. iDTRONIC does not give any guarantee promise for perfect function in cross environments. The companies or products mentioned in this document might be brands or brand names of the different suppliers or their subsidiaries in any country. This document may be downloaded onto a computer, stored and duplicated as necessary to support the use of the related iDTRONIC products. Any other type of duplication, circulation or storage on data carriers in any manner not authorized by iDTRONIC represents a violation of the applicable copyright laws and shall be prosecuted.

### Safety Instructions / Warning - Read before start-up!

- The device may only be used for the intended purpose designed by the manufacturer. The operation manual should be conveniently kept available at all times for each user.
- Unauthorized changes and the use of spare parts and additional devices that have not been sold or recommended by the manufacturer may cause fire, electric shocks or injuries. Such unauthorized measures shall exclude any liability by the manufacturer.

- The liability-prescriptions of the manufacturer in the issue valid at the time of purchase are valid for the device. The manufacturer shall not be held legally responsible for inaccuracies, errors, or omissions in the manual or automatically set parameters for a device or for an incorrect application of a device.
- Repairs may be executed by the manufacturer only.
- Only qualified personnel should carry out installation, operation, and maintenance procedures.
- Use of the device and its installation must be in accordance with national legal requirements and local electrical codes.
- When working on devices the valid safety regulations must be observed.

## Table of Contents

1	Introduction.....	8
2	Library Description .....	9
2.1	Typedefs.....	9
2.1.1	BLUEBOX_Handle .....	9
2.2	Enumerations .....	9
2.2.1	BLUEBOX_ErrorCodes .....	9
2.2.2	BLUEBOX_Output .....	10
2.2.3	BLUEBOX_Input .....	10
2.2.4	BLUEBOX_Antenna .....	10
2.2.5	BLUEBOX_TagType.....	11
2.2.6	BLUEBOX_ICODE_SLI_S_PasswordIdentifier .....	13
2.2.7	BLUEBOX_ICODE_SLI_S_ProtectionStatus.....	13
2.2.8	BLUEBOX_MIFARE_Key .....	14
2.2.9	BLUEBOX_ISO18K6C_Bank .....	14
2.2.10	BLUEBOX_ISO18K6C_PasswordPermission .....	15
2.2.11	BLUEBOX_ISO18K6C_MemoryPermission .....	15
2.2.12	BLUEBOX_Reader.....	16
2.3	Definitions.....	16
2.3.1	BLUEBOX_EM4305_ID_SIZE .....	16
2.3.2	BLUEBOX_T5557_ID_SIZE .....	16
2.3.3	BLUEBOX_Q5_ID_SIZE .....	17
2.3.4	BLUEBOX_HITAG1_ID_SIZE .....	17
2.3.5	BLUEBOX_HITAG1_PAGE_SIZE .....	17
2.3.6	BLUEBOX_HITAGS_ID_SIZE .....	17
2.3.7	BLUEBOX_HITAGS_PAGE_SIZE .....	17
2.3.8	BLUEBOX_TITAN_ID_SIZE .....	17
2.3.9	BLUEBOX_TITAN_PASSWORD_SIZE .....	17
2.3.10	BLUEBOX_TITAN_PAGE_SIZE .....	18
2.3.11	BLUEBOX_ISO15693_UID_SIZE .....	18
2.3.12	BLUEBOX_ICODE_SLI_S_RND_SIZE .....	18
2.3.13	BLUEBOX_ICODE_SLI_S_PWD_SIZE.....	18
2.3.14	BLUEBOX_MIFARE_MINI_UID_SIZE .....	18
2.3.15	BLUEBOX_MIFARE_1k_UID_SIZE .....	18
2.3.16	BLUEBOX_MIFARE_1k_BLOCK_SIZE .....	18
2.3.17	BLUEBOX_MIFARE_4k_UID_SIZE .....	19
2.3.18	BLUEBOX_MIFARE_4k_BLOCK_SIZE .....	19
2.3.19	BLUEBOX_MIFARE_UL_UID_SIZE .....	19
2.3.20	BLUEBOX_MIFARE_UL_BLOCK_SIZE.....	19
2.3.21	BLUEBOX_MIFARE_KEY_SIZE .....	19
2.3.22	BLUEBOX_MIFARE_DESFIRE_UID_SIZE .....	19
2.3.23	BLUEBOX_MIFARE_7BUID_2k_UID_SIZE .....	19
2.3.24	BLUEBOX_MIFARE_7BUID_4k_UID_SIZE .....	20

2.3.24	BLUEBOX_MIFARE_PLUS_2k_UID_SIZE .....	20
2.3.25	BLUEBOX_MIFARE_PLUS_4k_UID_SIZE .....	20
2.3.26	BLUEBOX_NTAG21x_UID_SIZE .....	20
2.3.27	BLUEBOX_NTAG21x_BLOCK_SIZE .....	20
2.3.28	BLUEBOX_SR176_UID_SIZE .....	20
2.3.29	BLUEBOX_SR176_BLOCK_SIZE .....	20
2.3.30	BLUEBOX_JCOS_UID_SIZE .....	21
2.3.31	BLUEBOX_PICOPASS_UID_SIZE .....	21
2.3.32	BLUEBOX_ISO18K6B_UID_SIZE .....	21
2.3.33	BLUEBOX_ISO18K6B_BLOCK_SIZE .....	21
2.3.34	BLUEBOX_ISO18K6C_UID_SIZE .....	21
2.3.35	BLUEBOX_ISO18K6C_BLOCK_SIZE .....	21
2.3.36	BLUEBOX_ISO18K6C_ACC_PWD_SIZE .....	21
2.3.37	BLUEBOX_ISO18K6C_KILL_PWD_SIZE .....	22
2.3.38	BLUEBOX_ACTIVE_UID_SIZE .....	22
2.4	Data Structures .....	22
2.4.1	BLUEBOX_Tag .....	22
2.4.2	BLUEBOX_Notify .....	22
2.4.3	BLUEBOX_ICODE_SLI_S_BlockProtectionStatus .....	23
2.4.4	BLUEBOX_Registration .....	23
2.5	Functions .....	24
2.5.1	BLUEBOX_GetSwRelease .....	24
2.5.2	BLUEBOX_Init .....	24
2.5.3	BLUEBOX_End .....	24
2.5.4	BLUEBOX_Open .....	25
2.5.5	BLUEBOX_Close .....	25
2.5.6	BLUEBOX_SetAddress .....	26
2.5.7	BLUEBOX_SetDevice .....	26
2.5.8	BLUEBOX_GetDevice .....	27
2.5.9	BLUEBOX_SetChannel .....	29
2.5.10	BLUEBOX_GetFwRelease .....	29
2.5.11	BLUEBOX_Reset .....	30
2.5.12	BLUEBOX_GetDateTime .....	31
2.5.13	BLUEBOX_SetDateTime .....	31
2.5.14	BLUEBOX_ReadParameters .....	32
2.5.15	BLUEBOX_WriteParameters .....	32
2.5.16	BLUEBOX_DefaultParameters .....	33
2.5.17	BLUEBOX_ReadSerialNumber .....	33
2.5.18	BLUEBOX_ReadMACAddress .....	34
2.5.19	BLUEBOX_ReadConfiguration .....	35
2.5.20	BLUEBOX_WriteConfiguration .....	35
2.5.21	BLUEBOX_DefaultConfiguration .....	36
2.5.22	BLUEBOX_DataRequest .....	36
2.5.23	BLUEBOX_QueueRequest .....	37
2.5.24	BLUEBOX_FreeTagsMemory .....	37
2.5.25	BLUEBOX_AllocateNotifyChannel .....	38

2.5.26	BLUEBOX_DeallocateNotifyChannel .....	38
2.5.27	BLUEBOX_GetNotification .....	39
2.5.28	BLUEBOX_FreeNotifyMemory .....	40
2.5.29	BLUEBOX_SetOutput .....	40
2.5.30	BLUEBOX_GetReaderStatus .....	41
2.5.31	BLUEBOX_GetTemperature.....	41
2.5.32	BLUEBOX_RfOnOff.....	42
2.5.33	BLUEBOX_SelectiveRfOnOff .....	42
2.5.34	BLUEBOX_ReadID_EM4305 .....	43
2.5.35	BLUEBOX_Write_EM4305 .....	44
2.5.36	BLUEBOX_ReadID_T5557.....	45
2.5.37	BLUEBOX_Write_T5557.....	46
2.5.38	BLUEBOX_ReadID_Q5.....	47
2.5.39	BLUEBOX_Write_Q5 .....	48
2.5.40	BLUEBOX_ReadID_HITAG1 .....	49
2.5.41	BLUEBOX_ReadPage_HITAG1 .....	50
2.5.42	BLUEBOX_WritePage_HITAG1 .....	51
2.5.43	BLUEBOX_ReadID_HITAGS.....	52
2.5.44	BLUEBOX_Write_HITAGS .....	53
2.5.45	BLUEBOX_ReadPage_HITAGS .....	54
2.5.46	BLUEBOX_WritePage_HITAGS.....	55
2.5.47	BLUEBOX_ReadID_TITAN .....	55
2.5.48	BLUEBOX_Reset_TITAN .....	56
2.5.49	BLUEBOX_Login_TITAN.....	57
2.5.50	BLUEBOX_WritePassword_TITAN.....	58
2.5.51	BLUEBOX_SelectiveRead_TITAN.....	59
2.5.52	BLUEBOX_SelectiveWrite_TITAN .....	60
2.5.53	BLUEBOX_Inventory_ISO15693 .....	61
2.5.54	BLUEBOX_ReadPage_ISO15693 .....	62
2.5.55	BLUEBOX_ReadMultiPage_ISO15693 .....	63
2.5.56	BLUEBOX_WritePage_ISO15693.....	64
2.5.57	BLUEBOX_WriteMultiPage_ISO15693 .....	65
2.5.58	BLUEBOX_LockPage_ISO15693.....	66
2.5.59	BLUEBOX_Write_AFI_ISO15693.....	67
2.5.60	BLUEBOX_Lock_AFI_ISO15693 .....	68
2.5.61	BLUEBOX_GetRandomNumber_ICODE_SLI_S .....	69
2.5.62	BLUEBOX_SetPassword_ICODE_SLI_S.....	70
2.5.63	BLUEBOX_WritePassword_ICODE_SLI_S .....	71
2.5.64	BLUEBOX_LockPassword_ICODE_SLI_S .....	71
2.5.65	BLUEBOX_64BitPasswordProtection_ICODE_SLI_S .....	72
2.5.66	BLUEBOX_ProtectPage_ICODE_SLI_S.....	73
2.5.67	BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S .....	74
2.5.68	BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S .....	75
2.5.69	BLUEBOX_Destroy_SLI_S_ICODE_SLI_S.....	76
2.5.70	BLUEBOX_EnablePrivacy_ICODE_SLI_S .....	77
2.5.71	BLUEBOX_Inventory_ISO14443A .....	78

2.5.72	BLUEBOX_ReadBlock_MIFARE_1k.....	79
2.5.73	BLUEBOX_WriteBlock_MIFARE_1k .....	80
2.5.74	BLUEBOX_ReadBlock_MIFARE_4k.....	81
2.5.75	BLUEBOX_WriteBlock_MIFARE_4k .....	82
2.5.76	BLUEBOX_ReadBlock_MIFARE_Ultralight .....	83
2.5.77	BLUEBOX_WriteBlock_MIFARE_Ultralight.....	84
2.5.78	BLUEBOX_ReadBlock_NTAG213 .....	85
2.5.79	BLUEBOX_WriteBlock_NTAG213.....	86
2.5.80	BLUEBOX_ReadBlock_NTAG215 .....	87
2.5.81	BLUEBOX_WriteBlock_NTAG215.....	88
2.5.82	BLUEBOX_ReadBlock_NTAG216 .....	89
2.5.83	BLUEBOX_WriteBlock_NTAG216.....	90
2.5.84	BLUEBOX_Inventory_ISO14443B .....	91
2.5.85	BLUEBOX_ReadBlock_SR176 .....	92
2.5.86	BLUEBOX_WriteBlock_SR176.....	93
2.5.87	BLUEBOX_Inventory_PICOPASS .....	94
2.5.88	BLUEBOX_ReadRfParameters.....	95
2.5.89	BLUEBOX_WriteRfParameters .....	96
2.5.90	BLUEBOX_Inventory_ISO18K6B.....	96
2.5.91	BLUEBOX_Read_ISO18K6B .....	97
2.5.92	BLUEBOX_Write_ISO18K6B .....	98
2.5.93	BLUEBOX_Inventory_ISO18K6C.....	98
2.5.94	BLUEBOX_ProgramEPC_ISO18K6C .....	99
2.5.95	BLUEBOX_Read_ISO18K6C .....	100
2.5.96	BLUEBOX_Write_ISO18K6C .....	101
2.5.97	BLUEBOX_BlockWrite_ISO18K6C.....	102
2.5.98	BLUEBOX_Lock_ISO18K6C .....	103
2.5.99	BLUEBOX_Kill_ISO18K6C .....	106
2.5.100	BLUEBOX_FwUpgrade .....	107
2.5.101	BLUEBOX_ReadNumberOfRegistrations.....	108
2.5.102	BLUEBOX_ReadOlderRegistration.....	109
2.5.103	BLUEBOX_CancelOlderRegistration.....	109
2.5.104	BLUEBOX_CancelAllRegistrations .....	110
2.5.105	BLUEBOX_ReadPreviousRegistration .....	110
2.5.106	BLUEBOX_GenericCommand .....	111
3	BlueBox Gen1 Functions Table .....	112
4	BlueBox Gen2 Functions Table .....	114
5	BlueBox CX Functions Table .....	116
6	Document Revision History .....	118

## 1 Introduction

This manual describes the BLUEBOXLib library and its implemented functions. BLUEBOXLib is a set of ANSI C functions which allows a user program to use and configure all the Soltec BLUEBOX readers. The library is available in the following formats:

- Win32 DLL (Soltec provides the BLUEBOXLib.lib stub for Microsoft Visual C++ 6.0).
- x64 DLL (Soltec provides the BLUEBOXLib.lib stub for Microsoft Visual C++ 6.0).



## 2 Library Description

### 2.1 Typedefs

#### 2.1.1 BLUEBOX\_Handle

**Name:** BLUEBOX\_Handle  
**Description:** Handle type used to identify readers.  
**Syntax** typedef int BLUEBOX\_Handle;

### 2.2 Enumerations

#### 2.2.1 BLUEBOX\_ErrorCodes

**Name:** BLUEBOX\_ErrorCodes  
**Description:** Error codes enum.  
**Enumerator:** *BLUEBOX\_StatusOk*: Operation completed successfully.  
*BLUEBOX\_InitError*: Initialization error.  
*BLUEBOX\_InvalidHandle*: Invalid handle.  
*BLUEBOX\_InvalidChannel*: Invalid channel.  
*BLUEBOX\_InvalidParams*: Invalid parameters.  
*BLUEBOX\_GenericError*: Generic error.  
*BLUEBOX\_TimeoutError*: Communication error.  
*BLUEBOX\_ConnectionError*: Connection error.  
*BLUEBOX\_MemoryError*: Memory allocation error.  
*BLUEBOX\_InvalidCommand*: Invalid command.  
*BLUEBOX\_TagNotFound*: Tag not found.  
*BLUEBOX\_TagError*: Tag error.  
*BLUEBOX\_AllocationError*: Notify channel allocation error.  
*BLUEBOX\_FileError*: File error.  
*BLUEBOX\_RegistrationNotFound*: Registration not found.  
**Syntax** typedef enum BLUEBOX\_ErrorCodes  
{  
    BLUEBOX\_StatusOk = 0,  
    BLUEBOX\_InitError = -1,  
    BLUEBOX\_InvalidHandle = -2,  
    BLUEBOX\_InvalidChannel = -3,  
    BLUEBOX\_InvalidParams = -4,  
    BLUEBOX\_GenericError = -5,

```

BLUEBOX_TimeoutError = -6,
BLUEBOX_CommunicationError = -7,
BLUEBOX_ConnectionError = -8,
BLUEBOX_MemoryError = -9,
BLUEBOX_InvalidCommand = -10,
BLUEBOX_TagNotFound = -11,
BLUEBOX_TagError = -12,
BLUEBOX_AllocationError = -13,
BLUEBOX_FileError = -14,
BLUEBOX_RegistrationNotFound = -15

} BLUEBOX_ErrorCodes;

```

### 2.2.2 BLUEBOX\_Output

**Name:** BLUEBOX\_Output

**Description:** Output enum.

**Enumerator:** *BLUEBOX\_OUTPUT\_1*: Output 1.  
*BLUEBOX\_OUTPUT\_2*: Output 2.

**Syntax**

```

typedef enum BLUEBOX_Output
{
    BLUEBOX_OUTPUT_1 = 1,
    BLUEBOX_OUTPUT_2 = 2
} BLUEBOX_Output;

```

### 2.2.3 BLUEBOX\_Input

**Name:** BLUEBOX\_Input

**Description:** Input enum.

**Enumerator:** *BLUEBOX\_NOINPUT*: No input information.  
*BLUEBOX\_INPUT\_1*: Input 1.  
*BLUEBOX\_INPUT\_2*: Input 2.

**Syntax**

```

typedef enum BLUEBOX_Input
{
    BLUEBOX_NPINPUT= 0,
    BLUEBOX_INPUT_1 = 1,
    BLUEBOX_INPUT_2 = 2
} BLUEBOX_Input;

```

### 2.2.4 BLUEBOX\_Antenna

**Name:** BLUEBOX\_Antenna

**Description:** Antenna enum.

**Enumarator:** *BLUEBOX\_NOANT*: No antenna information.  
*BLUEBOX\_ANT\_1*: Antenna nr. 1.  
*BLUEBOX\_ANT\_2*: Antenna nr. 2.  
*BLUEBOX\_ANT\_3*: Antenna nr. 3.  
*BLUEBOX\_ANT\_4*: Antenna nr. 4.

**Syntax**

```
typedef enum BLUEBOX_Antenna
{
    BLUEBOX_NOANT = 0,
    BLUEBOX_ANT_1 = 1,
    BLUEBOX_ANT_2 = 2,
    BLUEBOX_ANT_3 = 3,
    BLUEBOX_ANT_4 = 4
} BLUEBOX_Antenna;
```

### 2.2.5 BLUEBOX\_TagType

**Name:** BLUEBOX\_TagType

**Description:** Tag type enum.

**Enumarator:** *BLUEBOX\_UNDEFINED*: Undefined tag.  
*BLUEBOX\_SHORT*: BLUEBOX SHORT.  
*BLUEBOX\_MEDIUM*: BLUEBOX MEDIUM.  
*BLUEBOX\_LARGE*: BLUEBOX LARGE.  
*BLUEBOX\_EM4305*: EM4305.  
*BLUEBOX\_T5557*: T5557.  
*BLUEBOX\_Q5*: Q5.  
*BLUEBOX\_HITAG\_1*: HITAG 1.  
*BLUEBOX\_HITAG\_S256*: HITAG S 256.  
*BLUEBOX\_HITAG\_S2048*: HITAG S 2048.  
*BLUEBOX\_TITAN*: TITAN.  
*BLUEBOX\_ISO14443A*: ISO 14443A.  
*BLUEBOX\_MIFARE\_1k*: MIFARE 1k.  
*BLUEBOX\_MIFARE\_4k*: MIFARE 4k.  
*BLUEBOX\_MIFARE\_UL*: MIFARE Ultralight.  
*BLUEBOX\_ISO15693*: ISO 15693.  
*BLUEBOX\_ICODE2*: ICODE SLI.  
*BLUEBOX\_ICODE\_SLI\_S*: ICODE SLI-S.  
*BLUEBOX\_TAG\_IT\_HF\_I*: Tag-It HF-I.  
*BLUEBOX\_EM4035*: EM4035.  
*BLUEBOX\_LRI\_64\_512*: LRI 64/512.  
*BLUEBOX\_MB89R118*: MB89R118.  
*BLUEBOX\_ISO14443B*: ISO 14443B.  
*BLUEBOX\_SR176*: SR176.  
*BLUEBOX\_ISO18K6B*: ISO 18000-6B.

*BLUEBOX\_ISO18K6C*: ISO 18000-6C.  
*BLUEBOX\_ACTIVE*: ACTIVE.  
*BLUEBOX\_EM4305*: EM4305.  
*BLUEBOX\_T5557*: T5557.  
*BLUEBOX\_ICODE\_SLI\_S*: ICODE SLI-S.  
*BLUEBOX\_HITAG\_1*: HITAG 1.  
*BLUEBOX\_MIFARE\_MINI*: MIFARE Mini.  
*BLUEBOX\_MIFARE\_DESFIRE*: MIFARE Desfire.  
*BLUEBOX\_MIFARE\_7BUID\_2k*: MIFARE 2k with 7 bytes UID.  
*BLUEBOX\_MIFARE\_7BUID\_4k*: MIFARE 4k with 7 bytes UID.  
*BLUEBOX\_MIFARE\_PLUS\_2k*: MIFARE Plus 2k.  
*BLUEBOX\_MIFARE\_PLUS\_4k*: MIFARE Plus 4k.  
*BLUEBOX\_SRI512*: SRI 512.  
*BLUEBOX\_JCOS*: JCos.  
*BLUEBOX\_PICOPASS*: Picopass

## Syntax

```

typedef enum BLUEBOX_TagType
{
    BLUEBOX_UNDEFINED = 0,
    BLUEBOX_SHORT = 1,
    BLUEBOX_MEDIUM = 2,
    BLUEBOX_LARGE = 3,
    BLUEBOX_Q5 = 4,
    BLUEBOX_HITAG_S256 = 5,
    BLUEBOX_HITAG_S2048 = 6,
    BLUEBOX_TITAN = 7,
    BLUEBOX_ISO14443A = 8,
    BLUEBOX_MIFARE_1k = 9,
    BLUEBOX_MIFARE_4k = 10,
    BLUEBOX_MIFARE_UL = 11,
    BLUEBOX_ISO15693 = 12,
    BLUEBOX_ICODE2 = 13,
    BLUEBOX_TAG_IT_HF_I = 14,
    BLUEBOX_EM4035 = 15,
    BLUEBOX_LRI_64_512 = 16,
    BLUEBOX_MB89R118 = 17,
    BLUEBOX_ISO14443B = 18,
    BLUEBOX_SR176 = 19,
    BLUEBOX_ISO18K6B = 20,
    BLUEBOX_ISO18K6C = 21,
    BLUEBOX_ACTIVE = 22,
    BLUEBOX_EM4305 = 23,
    BLUEBOX_T5557 = 24,
    BLUEBOX_ICODE_SLI_S = 25,

```

```

BLUEBOX_HITAG_1 = 26,
BLUEBOX_MIFARE_MINI = 27,
BLUEBOX_MIFARE_DESFIRE = 28,
BLUEBOX_MIFARE_7BUID_2k = 29,
BLUEBOX_MIFARE_7BUID_4k = 30,
BLUEBOX_MIFARE_PLUS_2k = 31,
BLUEBOX_MIFARE_PLUS_4k = 32,
BLUEBOX_SRI512 = 33,
BLUEBOX_JCOS = 34,
BLUEBOX_PICOPASS = 35

} BLUEBOX_TagType;

```

#### 2.2.6 BLUEBOX\_ICODE\_SLI\_S\_PasswordIdentifier

**Name:** BLUEBOX\_ICODE\_SLI\_S\_PasswordIdentifier

**Description:** ICODE SLI-S password identifier enum.

**Enumerator:** *BLUEBOX\_ICODE\_SLI\_S\_PWD\_READ*: Read.  
*BLUEBOX\_ICODE\_SLI\_S\_PWD\_WRITE*: Write.  
*BLUEBOX\_ICODE\_SLI\_S\_PWD\_PRIVACY*: Privacy.  
*BLUEBOX\_ICODE\_SLI\_S\_DESTROY\_SLI\_S*: Destroy SLI-S.  
*BLUEBOX\_ICODE\_SLI\_S\_EAS*: EAS.

**Syntax**

```

typedef enum BLUEBOX_ICODE_SLI_S_PasswordIdentifier
{
    BLUEBOX_ICODE_SLI_S_PWD_READ = 0x01,
    BLUEBOX_ICODE_SLI_S_PWD_WRITE = 0x02,
    BLUEBOX_ICODE_SLI_S_PWD_PRIVACY = 0x04,
    BLUEBOX_ICODE_SLI_S_PWD_DESTROY_SLI_S = 0x08,
    BLUEBOX_ICODE_SLI_S_PWD_EAS = 0x10

} BLUEBOX_ICODE_SLI_S_PasswordIdentifier;

```

#### 2.2.7 BLUEBOX\_ICODE\_SLI\_S\_ProtectionStatus

**Name:** BLUEBOX\_ICODE\_SLI\_S\_ProtectionStatus

**Description:** ICODE SLI-S protection status enum.

**Enumerator:** **32-bit Password Protection:**  
*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_PUBLIC*: Public.  
*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_READ\_AND\_WRITE\_BY\_READ\_PWD*: Protect Read and Write by Read password.  
*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_WRITE\_BY\_WRITE\_PWD*: Protect Write by Write password.  
*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_READ\_BY\_READ\_PWD\_AND\_WRITE\_BY\_WRITE\_PWD*: Protect Read by Read

password and Write by Write password.

**64-bit Password Protection:**

*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_PUBLIC*: Public.

*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_READ\_AND\_WRITE\_BY\_READ\_AND\_WRITE\_PWD*: Protect Read and Write by Read plus Write password.

*BLUEBOX\_ICODE\_SLI\_S\_PROTECT\_WRITE\_BY\_READ\_AND\_WRITE\_PWD*: Protect Write by Read plus Write password.

**Syntax**

```
typedef enum BLUEBOX_ICODE_SLI_S_ProtectionStatus
{
    BLUEBOX_ICODE_SLI_S_PROTECT_PUBLIC = 0x00,
    BLUEBOX_ICODE_SLI_S_PROTECT_READ_AND_WRITE_BY_READ_PWD = 0x01,
    BLUEBOX_ICODE_SLI_S_PROTECT_WRITE_BY_WRITE_PWD = 0x10,
    BLUEBOX_ICODE_SLI_S_PROTECT_READ_BY_READ_PWD_AND_WRITE_BY_WRITE_PWD = 0x11,
    BLUEBOX_ICODE_SLI_S_PROTECT_READ_AND_WRITE_BY_READ_AND_WRITE_PWD = 0x01,
    BLUEBOX_ICODE_SLI_S_PROTECT_WRITE_BY_READ_AND_WRITE_PWD = 0x10,
} BLUEBOX_ICODE_SLI_S_ProtectionStatus;
```

### 2.2.8 BLUEBOX\_MIFARE\_Key

**Name:** BLUEBOX\_MIFARE\_Key

**Description:** MIFARE key enum.

**Enumarator:** *BLUEBOX\_MIFARE\_KEY\_A*: Key A.  
*BLUEBOX\_MIFARE\_KEY\_B*: Key B.

**Syntax**

```
typedef enum BLUEBOX_MIFARE_Key
{
    BLUEBOX_MIFARE_KEY_A = 0,
    BLUEBOX_MIFARE_KEY_B = 1
} BLUEBOX_MIFARE_Key;
```

### 2.2.9 BLUEBOX\_ISO18K6C\_Bank

**Name:** BLUEBOX\_ISO18K6C\_Bank

**Description:** ISO18000-6B tag's memory bank enum.

**Enumarator:** *BLUEBOX\_ISO18K6C\_BANK\_RESERVED*: Reserved.  
*BLUEBOX\_ISO18K6C\_BANK\_EPC*: EPC.  
*BLUEBOX\_ISO18K6C\_BANK\_TID*: TID.  
*BLUEBOX\_ISO18K6C\_BANK\_USER*: User.

**Syntax**

```
typedef enum BLUEBOX_ISO18K6C_Bank
{
    BLUEBOX_ISO18K6C_BANK_RESERVED = 0,
    BLUEBOX_ISO18K6C_BANK_EPC = 1,
    BLUEBOX_ISO18K6C_BANK_TID = 2,
    BLUEBOX_ISO18K6C_BANK_USER = 3

} BLUEBOX_ISO18K6C_Bank;
```

2.2.10 BLUEBOX\_ISO18K6C\_PasswordPermission

**Name:**

BLUEBOX\_ISO18K6C\_PasswordPermission

**Description:**

The ISO 18000-6C tag password permission values enum.

**Enumerator:**

*BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ACCESSIBLE*: Accessible from either opened and secured states.  
*BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_ACCESSIBLE*: Permanently accessible from either opened and secured states. It couldn't be locked.  
*BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_SECURED\_ACCESSIBLE*: Accessible only from secured state.  
*BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_NOT\_ACCESSIBLE*: Not accessible from either opened or secured states.  
*BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_NO\_CHANGE*: No change in accessible options.

**Syntax**

```
typedef enum BLUEBOX_ISO18K6C_PasswordPermission
{
    BLUEBOX_ISO18K6C_TAG_PWD_PERM_ACCESSIBLE = 0,
    BLUEBOX_ISO18K6C_TAG_PWD_PERM_ALWAYS_ACCESSIBLE = 1,
    BLUEBOX_ISO18K6C_TAG_PWD_PERM_SECURED_ACCESSIBLE = 2,
    BLUEBOX_ISO18K6C_TAG_PWD_PERM_ALWAYS_NOT_ACCESSIBLE = 3,
    BLUEBOX_ISO18K6C_TAG_PWD_PERM_NO_CHANGE = 4

} BLUEBOX_ISO18K6C_PasswordPermission;
```

2.2.11 BLUEBOX\_ISO18K6C\_MemoryPermission

**Name:**

BLUEBOX\_ISO18K6C\_MemoryPermission

**Description:**

The ISO 18000-6C tag memory permission values enum.

**Enumerator:**

*BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_WRITABLE*: Writable from either opened and secured states.  
*BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_WRITABLE*: Permanently writable from either opened and secured states. It couldn't be locked.  
*BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_SECURED\_WRITABLE*: Writable only from secured state.

*BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_NOT\_WRITABLE*: Not writable from either opened or secured states.

*BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_NO\_CHANGE*: No change in writable options.

## Syntax

```
typedef enum BLUEBOX_ISO18K6C_MemoryPermission
{
    BLUEBOX_ISO18K6C_TAG_MEM_PERM_WRITABLE= 0,
    BLUEBOX_ISO18K6C_TAG_MEM_PERM_ALWAYS_WRITABLE= 1,
    BLUEBOX_ISO18K6C_TAG_MEM_PERM_SECURED_WRITABLE= 2,
    BLUEBOX_ISO18K6C_TAG_MEM_PERM_ALWAYS_NOT_WRITABLE= 3,
    BLUEBOX_ISO18K6C_TAG_MEM_PERM_NO_CHANGE= 4
} BLUEBOX_ISO18K6C_MemoryPermission;
```

### 2.2.12 BLUEBOX\_Reader

**Name:** BLUEBOX\_Reader

**Description:** Reader (primary, auxiliary, ...) ID.

**Enumerator:** *BLUEBOX\_PRIMARY\_READER*: Primary reader.  
*BLUEBOX\_AUXILIARY\_1\_READER*: First auxiliary reader.  
*BLUEBOX\_AUXILIARY\_2\_READER*: 2nd auxiliary reader.

## Syntax

```
typedef enum BLUEBOX_Reader
{
    BLUEBOX_PRIMARY_READER = 0,
    BLUEBOX_AUXILIARY_1_READER = 1,
    BLUEBOX_AUXILIARY_2_READER = 1,
} BLUEBOX_Reader;
```

## 2.3 Definitions

### 2.3.1 BLUEBOX\_EM4305\_ID\_SIZE

**Name:** BLUEBOX\_EM4305\_ID\_SIZE

**Description:** EM4305 tag's ID size in bytes.

**Syntax** #define BLUEBOX\_EM4305\_ID\_SIZE (4)

### 2.3.2 BLUEBOX\_T5557\_ID\_SIZE

**Name:** BLUEBOX\_T5557\_ID\_SIZE

**Description:** T5557 tag's ID size in bytes.

**Syntax** #define BLUEBOX\_T5557\_ID\_SIZE (8)



### 2.3.3 BLUEBOX\_Q5\_ID\_SIZE

**Name:** BLUEBOX\_Q5\_ID\_SIZE  
**Description:** Q5 tag's ID size in bytes.  
**Syntax** #define BLUEBOX\_Q5\_ID\_SIZE (5)

### 2.3.4 BLUEBOX\_HITAG1\_ID\_SIZE

**Name:** BLUEBOX\_HITAG1\_ID\_SIZE  
**Description:** HITAG 1 tag's ID size in bytes.  
**Syntax** #define BLUEBOX\_HITAG1\_ID\_SIZE (4)

### 2.3.5 BLUEBOX\_HITAG1\_PAGE\_SIZE

**Name:** BLUEBOX\_HITAG1\_PAGE\_SIZE  
**Description:** HITAG 1 tag's memory page size in bytes.  
**Syntax** #define BLUEBOX\_HITAG1\_PAGE\_SIZE (4)

### 2.3.6 BLUEBOX\_HITAGS\_ID\_SIZE

**Name:** BLUEBOX\_HITAGS\_ID\_SIZE  
**Description:** HITAG S tag's ID size in bytes.  
**Syntax** #define BLUEBOX\_HITAGS\_ID\_SIZE (4)

### 2.3.7 BLUEBOX\_HITAGS\_PAGE\_SIZE

**Name:** BLUEBOX\_HITAGS\_PAGE\_SIZE  
**Description:** HITAG S tag's memory page size in bytes.  
**Syntax** #define BLUEBOX\_HITAGS\_PAGE\_SIZE (4)

### 2.3.8 BLUEBOX\_TITAN\_ID\_SIZE

**Name:** BLUEBOX\_TITAN\_ID\_SIZE  
**Description:** TITAN tag's ID size in bytes.  
**Syntax** #define BLUEBOX\_TITAN\_ID\_SIZE (8)

### 1.1.1 BLUEBOX\_TITAN\_PASSWORD\_SIZE

**Name:** BLUEBOX\_TITAN\_PASSWORD\_SIZE  
**Description:** TITAN tag's password size in bytes.  
**Syntax** #define BLUEBOX\_TITAN\_PASSWORD\_SIZE (4)

### 2.3.9 BLUEBOX\_TITAN\_PAGE\_SIZE

**Name:** BLUEBOX\_TITAN\_PAGE\_SIZE  
**Description:** TITAN tag's memory page size in bytes.  
**Syntax** #define BLUEBOX\_TITAN\_PAGE\_SIZE (4)

### 2.3.10 BLUEBOX\_ISO15693\_UID\_SIZE

**Name:** BLUEBOX\_ISO15693\_UID\_SIZE  
**Description:** ISO 15693 tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_ISO15693\_UID\_SIZE (8)

### 2.3.11 BLUEBOX\_ICODE\_SLI\_S\_RND\_SIZE

**Name:** BLUEBOX\_ICODE\_SLI\_S\_RND\_SIZE  
**Description:** ICODE SLI-S tag's random number size in bytes.  
**Syntax** #define BLUEBOX\_ICODE\_SLI\_S\_RND\_SIZE (2)

### 2.3.12 BLUEBOX\_ICODE\_SLI\_S\_PWD\_SIZE

**Name:** BLUEBOX\_ICODE\_SLI\_S\_PWD\_SIZE  
**Description:** ICODE SLI-S tag's password size in bytes.  
**Syntax** #define BLUEBOX\_ICODE\_SLI\_S\_PWD\_SIZE (4)

### 2.3.13 BLUEBOX\_MIFARE\_MINI\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_MINI\_UID\_SIZE  
**Description:** MIFARE Mini tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_MINI\_UID\_SIZE (4)

### 2.3.14 BLUEBOX\_MIFARE\_1k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_1k\_UID\_SIZE  
**Description:** MIFARE 1k tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_1k\_UID\_SIZE (4)

### 2.3.15 BLUEBOX\_MIFARE\_1k\_BLOCK\_SIZE

**Name:** BLUEBOX\_MIFARE\_1k\_BLOCK\_SIZE  
**Description:** MIFARE 1k tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_1k\_BLOCK\_SIZE (16)

#### 2.3.16 BLUEBOX\_MIFARE\_4k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_4k\_UID\_SIZE  
**Description:** MIFARE 4k tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_4k\_UID\_SIZE (4)

#### 2.3.17 BLUEBOX\_MIFARE\_4k\_BLOCK\_SIZE

**Name:** BLUEBOX\_MIFARE\_4k\_BLOCK\_SIZE  
**Description:** MIFARE 4k tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_4k\_BLOCK\_SIZE (16)

#### 2.3.18 BLUEBOX\_MIFARE\_UL\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_UL\_UID\_SIZE  
**Description:** MIFARE Ultralight tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_UL\_UID\_SIZE (7)

#### 2.3.19 BLUEBOX\_MIFARE\_UL\_BLOCK\_SIZE

**Name:** BLUEBOX\_MIFARE\_UL\_BLOCK\_SIZE  
**Description:** MIFARE Ultralight tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_UL\_BLOCK\_SIZE (4)

#### 2.3.20 BLUEBOX\_MIFARE\_KEY\_SIZE

**Name:** BLUEBOX\_MIFARE\_KEY\_SIZE  
**Description:** MIFARE tag's key size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_KEY\_SIZE (6)

#### 2.3.21 BLUEBOX\_MIFARE\_DESFIRE\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_DESFIRE\_UID\_SIZE  
**Description:** MIFARE Desfire tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_DESFIRE\_UID\_SIZE (7)

#### 2.3.22 BLUEBOX\_MIFARE\_7BUID\_2k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_7BUID\_2k\_UID\_SIZE  
**Description:** MIFARE 2k 7-bytes UID tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_7BUID\_2k\_UID\_SIZE (7)

### 2.3.23 BLUEBOX\_MIFARE\_7BUID\_4k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_7BUID\_4k\_UID\_SIZE  
**Description:** MIFARE 4k 7-bytes UID tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_7BUID\_4k\_UID\_SIZE (7)

### 2.3.24 BLUEBOX\_MIFARE\_PLUS\_2k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_PLUS\_2k\_UID\_SIZE  
**Description:** MIFARE Plus 2k tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_PLUS\_2k\_UID\_SIZE (7)

### 2.3.25 BLUEBOX\_MIFARE\_PLUS\_4k\_UID\_SIZE

**Name:** BLUEBOX\_MIFARE\_PLUS\_4k\_UID\_SIZE  
**Description:** MIFARE Plus 4k tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_MIFARE\_PLUS\_4k\_UID\_SIZE (7)

### 2.3.26 BLUEBOX\_NTAG21x\_UID\_SIZE

**Name:** BLUEBOX\_NTAG21x\_UID\_SIZE  
**Description:** NTAG21x tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_NTAG21x\_UID\_SIZE (7)

### 2.3.27 BLUEBOX\_NTAG21x\_BLOCK\_SIZE

**Name:** BLUEBOX\_NTAG21x\_BLOCK\_SIZE  
**Description:** NTAG21x tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_NTAG21x\_BLOCK\_SIZE (4)

### 2.3.28 BLUEBOX\_SR176\_UID\_SIZE

**Name:** BLUEBOX\_SR176\_UID\_SIZE  
**Description:** SR176 tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_SR176\_UID\_SIZE (8)

### 2.3.29 BLUEBOX\_SR176\_BLOCK\_SIZE

**Name:** BLUEBOX\_SR176\_BLOCK\_SIZE  
**Description:** SR176 tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_SR176\_BLOCK\_SIZE (2)

### 2.3.30 BLUEBOX\_JCOS\_UID\_SIZE

**Name:** BLUEBOX\_JCOS\_UID\_SIZE  
**Description:** JCos tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_JCOS\_UID\_SIZE (8)

### 2.3.31 BLUEBOX\_PICOPASS\_UID\_SIZE

**Name:** BLUEBOX\_PICOPASS\_UID\_SIZE  
**Description:** Picopass tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_PICOPASS\_UID\_SIZE (8)

### 2.3.32 BLUEBOX\_ISO18K6B\_UID\_SIZE

**Name:** BLUEBOX\_ISO18K6B\_UID\_SIZE  
**Description:** ISO 18000-6B tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6B\_UID\_SIZE (8)

### 2.3.33 BLUEBOX\_ISO18K6B\_BLOCK\_SIZE

**Name:** BLUEBOX\_ISO18K6B\_BLOCK\_SIZE  
**Description:** ISO 18000-6B tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6B\_BLOCK\_SIZE (8)

### 2.3.34 BLUEBOX\_ISO18K6C\_UID\_SIZE

**Name:** BLUEBOX\_ISO18K6C\_UID\_SIZE  
**Description:** ISO 18000-6C tag's UID maximum size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6C\_UID\_SIZE (66)

### 2.3.35 BLUEBOX\_ISO18K6C\_BLOCK\_SIZE

**Name:** BLUEBOX\_ISO18K6C\_BLOCK\_SIZE  
**Description:** ISO 18000-6C tag's memory block size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6C\_BLOCK\_SIZE (2)

### 2.3.36 BLUEBOX\_ISO18K6C\_ACC\_PWD\_SIZE

**Name:** BLUEBOX\_ISO18K6C\_ACC\_PWD\_SIZE  
**Description:** ISO 18000-6C tag's access password size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6C\_ACC\_PWD\_SIZE (4)

### 2.3.37 BLUEBOX\_ISO18K6C\_KILL\_PWD\_SIZE

**Name:** BLUEBOX\_ISO18K6C\_KILL\_PWD\_SIZE  
**Description:** ISO 18000-6C tag's kill password size in bytes.  
**Syntax** #define BLUEBOX\_ISO18K6C\_KILL\_PWD\_SIZE (4)

### 2.3.38 BLUEBOX\_ACTIVE\_UID\_SIZE

**Name:** BLUEBOX\_ACTIVE\_UID\_SIZE  
**Description:** ACTIVE tag's UID size in bytes.  
**Syntax** #define BLUEBOX\_ACTIVE\_UID\_SIZE (8)

## 2.4 Data Structures

### 2.4.1 BLUEBOX\_Tag

**Name:** BLUEBOX\_Tag  
**Description:** Tag identification struct.  
**Data fields:** *TagType*: Tag type.  
*Id*: Pointer to tag ID.  
*Length*: The length of the tag ID in bytes.  
*Antenna*: The antenna that identifies the tag.  
*Input*: The input (direction) of tag identification.  
**Syntax** typedef struct BLUEBOX\_Tag  
{  
BLUEBOX\_TagType TagType;  
unsigned char \*Id;  
int Length;  
BLUEBOX\_Antenna Antenna;  
BLUEBOX\_Input Input;  
} BLUEBOX\_Tag;

### 2.4.2 BLUEBOX\_Notify

**Name:** BLUEBOX\_Notify  
**Description:** Tag notification struct.  
**Data fields:** *Address*: The address of the reader that identifies the tag.  
*TagType*: Tag type.  
*Id*: Pointer to the tag ID.  
*Length*: The length of the tag ID in bytes.

*Antenna:* The antenna which have identified the tag.

*Input:* The input (direction) of tag identification.

## Syntax

```
typedef struct BLUEBOX_Tag
{
    unsigned char Address;
    BLUEBOX_TagType TagType;
    unsigned char *Id;
    int Length;
    BLUEBOX_Antenna Antenna;
    BLUEBOX_Input Input;
} BLUEBOX_Tag;
```

### 2.4.3 BLUEBOX\_ICODE\_SLI\_S\_BlockProtectionStatus

**Name:** BLUEBOX\_ICODE\_SLI\_S\_BlockProtectionStatus

**Description:** ICODE SLI-S block protection status struct.

**Data fields:** *Lock:* Lock bit (Write access condition).  
*ReadPasswordProtected:* Read password protected.  
*WritePasswordProtected:* Write password protected.  
*PageProtectionLock:* Page protection lock.

## Syntax

```
typedef struct BLUEBOX_ICODE_SLI_S_BlockProtectionStatus
{
    int LockBit;
    int ReadPasswordProtected;
    int WritePasswordProtected;
    int PageProtectionLock;
} BLUEBOX_ICODE_SLI_S_BlockProtectionStatus;
```

### 2.4.4 BLUEBOX\_Registration

**Name:** BLUEBOX\_Registration

**Description:** Registration struct.

**Data fields:** *TagType:* Tag type.  
*Id:* Pointer to the tag ID.  
*Length:* The length of the tag ID in bytes.  
*Antenna:* The antenna which have identified the tag.  
*Input:* The input which have activated the identification procedure.

## Syntax

```
typedef struct BLUEBOX_Registration
{
    BLUEBOX_TagType TagType;
    unsigned char *Id;
    int Length;
```

```
BLUEBOX_Antenna Antenna;
BLUEBOX_Input Input;

} BLUEBOX_Registration;
```

## 2.5 Functions

### 2.5.1 BLUEBOX\_GetSwRelease

**Name:** BLUEBOX\_GetSwRelease

**Reader:** All readers.

**Description:** This function gets the software release of the library.

**Parameters:** [out] SwRel: Software release of the library.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetSwRelease (char \*SwRel);

### 2.5.2 BLUEBOX\_Init

**Name:** BLUEBOX\_Init

**Reader:** All readers.

**Description:** This function creates an opaque handle to identify a module attached to PC.

**Parameters:** [out] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InitError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Init (BLUEBOX\_Handle \*Handle);

### 2.5.3 BLUEBOX\_End

**Name:** BLUEBOX\_End

**Reader:** All readers.

**Description:** This function notifies the library the end of operation and



frees the allocated memory. Implicitly calls the BLUEBOX\_Close function if the connection with the reader is open.

**Parameters:** [in] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_End (BLUEBOX\_Handle \*Handle);

#### 2.5.4 BLUEBOX\_Open

**Name:** BLUEBOX\_Open

**Reader:** All readers.

**Description:** This function opens the connection with the reader. If already connected it tries to close the connection and then opens it.

**Parameters:** [in] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_GenericError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Open (BLUEBOX\_Handle \*Handle);

#### 2.5.5 BLUEBOX\_Close

**Name:** BLUEBOX\_Close

**Reader:** All readers.

**Description:** This function closes the connection with the reader.

**Parameters:** [in] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Close (BLUEBOX\_Handle \*Handle);

#### 2.5.6 BLUEBOX\_SetAddress

**Name:** BLUEBOX\_SetAddress  
**Reader:** All readers.  
**Description:** This function sets the reader address.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Address: Address to use to communicate with the reader.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:  

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_SetAddress (BLUEBOX\_Handle \*Handle, unsigned char Address);

#### 2.5.7 BLUEBOX\_SetDevice

**Name:** BLUEBOX\_SetDevice  
**Reader:** All readers.  
**Description:** This function sets the reader type, frequency, range, and other parameters needed to communicate correctly with the reader  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Type: The reader type string. Use one of the strings listed below:  

- "DESKTOP": Desktop reader.
- "INDUSTRIAL": Industrial reader.
- "TINYOEM": OEM reader like OEM HF or OEM LF.
- "EASYOEM": OEM reader like OEM HF E.
- "BB OEM": OEM reader like OEM UHF.
- "PORTAL": Portal reader.

- "BB2 DESKTOP": Gen2 desktop reader.
- "BB2 INDUSTRIAL": Gen2 industrial reader.
- "BB2 BASIC": Gen2 basic reader.
- "CX": CX readers.

[in] Frequency: The reader frequency string. Use one of the strings listed below:

- "LF": Low Frequency (125 kHz).
- "HF": High Frequency (13.56 MHz).
- "UHF": Ultra High Frequency (860 – 960 MHz).
- "MICROWAVE": Microwave 2.4 GHz.

[in] Range: The reader range string. Use one of the strings listed below:

- "SHORT": Short range.
- "MID": Mid range.
- "LONG": Long range.

[in] Antennas: The reader antennas string. Use one of the values listed below:

- "SINGLE": Single antenna.
- "DUAL": Dual antennas.
- "QUAD": Quad antennas.

[in] Major: The firmware version major number.

[in] Minor: The firmware version minor number.

[in] Variant: The firmware variant. One char which identifies the firmware variant.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_InvalidParams.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_SetDevice (BLUEBOX_Handle *Handle, char
*Type, char *Frequency, char *Range, char *Antennas,
int Major, int Minor, char Variant);
```

### 2.5.8 BLUEBOX\_GetDevice

**Name:**

BLUEBOX\_GetDevice

**Reader:**

All readers.

**Description:**

This function gets the reader type, frequency, range, and

other parameters needed to communicate correctly with the reader

## Parameters:

[in] Handle: The handle that identifies the reader.

[out] Type: The reader type string. One of the strings listed below:

- "DESKTOP": Desktop reader.
- "INDUSTRIAL": Industrial reader.
- "TINYOEM": OEM reader like OEM HF or OEM LF.
- "EASYOEM": OEM reader like OEM HF E.
- "BB OEM": OEM reader like OEM UHF.
- "PORTAL": PORTAL reader.
- "BB2 DESKTOP": Gen2 desktop reader.
- "BB2 INDUSTRIAL": Gen2 industrial reader.
- "BB2 BASIC": Gen2 basic reader.
- "CX": CX readers.

[out] Frequency: The reader frequency string. One of the strings listed below:

- "LF": Low Frequency (125 kHz).
- "HF": High Frequency (13.56 MHz).
- "UHF": Ultra High Frequency (860 – 960 MHz).
- "MICROWAVE": Microwave 2.4 GHz.

[out] Range: The reader range string. One of the strings listed below:

- "SHORT": Short range.
- "MID": Mid range.
- "LONG": Long range.

[out] Antennas: The reader antennas string. One of the values listed below:

- "SINGLE": Single antenna.
- "DUAL": Dual antennas.
- "QUAD": Quad antennas.

[out] Major: The firmware version major number.

[out] Minor: The firmware version minor number.

[out] Variant: The firmware variant. One char which identifies the firmware variant.

## Return:

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetDevice (BLUEBOX\_Handle \*Handle, char  
\*Type, char \*Frequency, char \*Range, char \*Antennas,  
int \*Major, int \*Minor, char \*Variant);

### 2.5.9 BLUEBOX\_SetChannel

**Name:** BLUEBOX\_SetChannel

**Reader:** All readers.

**Description:** This function sets the notification channel to use with the reader.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Channel: Channel to use string. Use one of the strings listed below:

- "RS232": RS232.
- "RS485": RS485.
- "TCP": TCP.

[in] Settings: Channel settings string. Use one of the form listed below and depending on channel to use:

- RS232: "<port name>,<baud rate>,<data bits>,<parity>,<stop bits>,<retx>,<timeout>" (e.g. "COM1,19200,8,n,1,5,60000").
- RS485: "<port name>,<baud rate>,<data bits>,<parity>,<stop bits>,<retx>,<timeout>" (e.g. "COM1,19200,8,n,1,5,60000").
- TCP: "<ip>:<port>,<timeout>" (e.g. "192.168.4.200:3000,60000").

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_SetChannel (BLUEBOX\_Handle \*Handle, char  
\*Channel, char \*Settings);

**Remarks** The <timeout> field is expressed in ms. RS232 is also used with USB Virtual Com interfaces.

### 2.5.10 BLUEBOX\_GetFwRelease

**Name:** BLUEBOX\_GetFwRelease

<b>Reader:</b>	All readers.
<b>Description:</b>	This function gets the firmware release of the reader.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Reader: The reader to read the version firmware. Use one of the values listed below and defined in BLUEBOX_Reader in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_PRIMARY_READER: The primary reader.</li> <li>• BLUEBOX_AUXILIARY_1_READER: The 1<sup>st</sup> auxiliary reader.</li> <li>• BLUEBOX_AUXILIARY_2_READER: The 2<sup>nd</sup> auxiliary reader.</li> </ul> <p>[out] FwRel: The firmware release of the reader.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_GetFwRelease (BLUEBOX_Handle *Handle, BLUEBOX_Reader Reader, char *FwRel);</pre>
<b>Remarks</b>	This function must be called after opening the connection with the reader.

#### 2.5.11 BLUEBOX\_Reset

<b>Name:</b>	BLUEBOX_Reset
<b>Reader:</b>	All readers.
<b>Description:</b>	This function resets the reader.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader.
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_TimeoutError.</li> </ul>

- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetReset (BLUEBOX\_Handle \*Handle);

## 2.5.12 BLUEBOX\_GetDateTime

**Name:** BLUEBOX\_GetDateTime  
**Reader:** All BLUEBOX GEN2 INDUSTRIAL readers.  
**Description:** This function gets the date / time set in the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] DateTime: The date / time string in BCD format (YYYYMMDDHHMMSS). It is a null terminating string.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetDateTime (BLUEBOX\_Handle \*Handle, char \*DateTime);

## 2.5.13 BLUEBOX\_SetDateTime

**Name:** BLUEBOX\_SetDateTime  
**Reader:** All BLUEBOX GEN2 INDUSTRIAL readers.  
**Description:** This function sets the date / time set in the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] DateTime: The date / time string in BCD format (YYYYMMDDHHMMSS). It is a null terminating string.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.

- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_SetDateTime (BLUEBOX\_Handle \*Handle, char  
\*DateTime);

#### 2.5.14 BLUEBOX\_ReadParameters

**Name:** BLUEBOX\_ReadParameters  
**Reader:** All readers.  
**Description:** This function reads the general parameters of the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Parameters: General parameters set in the reader.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadParameters (BLUEBOX\_Handle \*Handle,  
unsigned char \*Parameters);

**Remarks** This function must be called after opening the connection with the reader and after reading the firmware version. See the reader technical manual for the Parameters format.

#### 2.5.15 BLUEBOX\_WriteParameters

**Name:** BLUEBOX\_WriteParameters  
**Reader:** All readers.  
**Description:** This function writes the general parameters of the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Parameters: General parameters to be set in the reader.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:



- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_WriteParameters (BLUEBOX\_Handle \*Handle,  
unsigned char \*Parameters);

**Remarks** See the reader technical manual for the Parameters format.

#### 2.5.16 BLUEBOX\_DefaultParameters

**Name:** BLUEBOX\_DefaultParameters

**Reader:** All readers.

**Description:** This function resets the parameters to the factory default values.

**Parameters:** [in] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_DefaultParameters (BLUEBOX\_Handle  
\*Handle);

#### 2.5.17 BLUEBOX\_ReadSerialNumber

**Name:** BLUEBOX\_ReadSerialNumber

**Reader:** All BLUEBOX GEN2 INDUSTRIAL readers, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

**Description:** This function reads the serial number of the reader.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Serial: Serial number set in the reader. 6-bytes length in BCD format.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadSerialNumber (BLUEBOX\_Handle \*Handle,  
unsigned char \*Serial);

#### 2.5.18 BLUEBOX\_ReadMACAddress

**Name:** BLUEBOX\_ReadMACAddress

**Reader:** All BLUEBOX GEN2 INDUSTRIAL readers, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

**Description:** This function reads the MAC address of the reader.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] MAC: MAC address of the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadMACAddress (BLUEBOX\_Handle \*Handle,  
unsigned char \*MAC);

### 2.5.19 BLUEBOX\_ReadConfiguration

<b>Name:</b>	BLUEBOX_ReadConfiguration
<b>Reader:</b>	All readers.
<b>Description:</b>	This function reads a configuration page of the reader.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Page: The configuration page number to read. [out] Config: Configuration set in the reader.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_ReadConfiguration (BLUEBOX_Handle *Handle, int Page, unsigned char * Config);</pre>
<b>Remarks</b>	The configuration buffer size in bytes depends on the page and it is 7 bytes length for pages between 00h and 7Fh, and 14 bytes length for pages between 80h and FFh.

### 2.5.20 BLUEBOX\_WriteConfiguration

<b>Name:</b>	BLUEBOX_WriteConfiguration
<b>Reader:</b>	All readers.
<b>Description:</b>	This function writes a configuration page of the reader.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Page: The configuration page number to write. [in] Config: The configuration to be set in the reader.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> </ul>

- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_WriteConfiguration (BLUEBOX\_Handle \*Handle,  
int Page, unsigned char \*Config);

**Remarks** The configuration buffer size in bytes depends on the page and it is 7 bytes length for pages between 00h and 7Fh, and 14 bytes length for pages between 80h and FFh.

#### 2.5.21 BLUEBOX\_DefaultConfiguration

**Name:** BLUEBOX\_DefaultConfiguration

**Reader:** All readers.

**Description:** This function resets the configuration to the factory default values.

**Parameters:** [in] Handle: The handle that identifies the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_DefaultConfiguration (BLUEBOX\_Handle \*Handle);

#### 2.5.22 BLUEBOX\_DataRequest

**Name:** BLUEBOX\_DataRequest

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function reads data from buffer. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Tags: The array containing the tags read.  
[out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in

BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_MemoryError.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_DataRequest (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

#### 2.5.23 BLUEBOX\_QueueRequest

**Name:** BLUEBOX\_QueueRequest

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function reads data from queue. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Tags: The array containing the tags read.  
[out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_MemoryError.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_QueueRequest (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

#### 2.5.24 BLUEBOX\_FreeTagsMemory

**Name:** BLUEBOX\_FreeTagsMemory

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function frees the memory allocated to store tags by BLUEBOX\_DataRequest or BLUEBOX\_QueueRequest or inventory commands.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Tags: The array containing the tags read.  
[in] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_FreeTagsMemory (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Tag \*\*Tags, int TagsNo);

#### 2.5.25 BLUEBOX\_AllocateNotifyChannel

**Name:** BLUEBOX\_AllocateNotifyChannel

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function allocates a notify channel in order to start a tag notification

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Address: The address of the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_AllocationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_AllocateNotifyChannel (BLUEBOX\_Handle  
\*Handle, unsigned char Address);

#### 2.5.26 BLUEBOX\_DeallocateNotifyChannel

**Name:** BLUEBOX\_DeallocateNotifyChannel

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function deallocates a notify channel in order to stop a tag notification

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Address: The address of the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidChannel.
- BLUEBOX\_AllocationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_DeallocateNotifyChannel (BLUEBOX\_Handle  
\*Handle, unsigned char Address);

#### 2.5.27 BLUEBOX\_GetNotification

**Name:** BLUEBOX\_GetNotification

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function reads data from notification buffer. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Tags: The array containing the tags read.  
[out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_MemoryError.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_AllocationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetNotification (BLUEBOX\_Handle \*Handle,

BLUEBOX\_Notify \*\*Tags, int \*TagsNo);

#### 2.5.28 BLUEBOX\_FreeNotifyMemory

<b>Name:</b>	BLUEBOX_FreeNotifyMemory
<b>Reader:</b>	All readers except of BLUEBOX PORTAL UHF.
<b>Description:</b>	This function frees the memory allocated to store tags by BLUEBOX_GetNotification commands.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Tags: The array containing the tags read. [in] TagsNo: The number of tags in the array.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> </ul>
<b>Syntax:</b>	BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_FreeNotifyMemory (BLUEBOX_Handle *Handle, BLUEBOX_Notify **Tags, int TagsNo);

#### 2.5.29 BLUEBOX\_SetOutput

<b>Name:</b>	BLUEBOX_SetOutput
<b>Reader:</b>	BLUEBOX OEM, BLUEBOX INDUSTRIAL, BLUEBOX GEN2 OEM, INDUSTRIAL and BASIC readers, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.
<b>Description:</b>	This function sets an output behavior.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Output: The value that identifies the output. Use one of the values defined in BLUEBOX_Output in BLUEBOXLib.h. [in] Period: Activation period from 1 (0x01) to 99 (0x63) seconds. Use 0x80 to continuously deactivate the output and 0x81 to continuously activate the output.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> </ul>



- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_SetOutput (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Output Output, unsigned char Period);

### 2.5.30 BLUEBOX\_GetReaderStatus

**Name:** BLUEBOX\_GetReaderStatus  
**Reader:** All readers.  
**Description:** This function reads the status of the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Status: The status of the reader.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetReaderStatus (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_ReaderStatus \*Status);

**Remarks** See the reader technical manual for the Status format.

### 2.5.31 BLUEBOX\_GetTemperature

**Name:** BLUEBOX\_GetTemperature  
**Reader:** All BLUEBOX GEN2 INDUSTRIAL readers, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.  
**Description:** This function reads the internal temperature of the reader.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Temperature: The internal temperature of the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetTemperature (BLUEBOX\_Handle \*Handle,  
double \*Temperature);

### 2.5.32 BLUEBOX\_RfOnOff

**Name:** BLUEBOX\_RfOnOff

**Reader:** All readers except of BLUEBOX PORTAL UHF.

**Description:** This function sets RF ON/OFF.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] OnOff: Flag to activate/deactivate the RF. Use one of the values listed below:

- = 0: To deactivate the RF.
- != 0: To activate the RF.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_RfOnOff (BLUEBOX\_Handle \*Handle, short  
OnOff);

### 2.5.33 BLUEBOX\_SelectiveRfOnOff

**Name:** BLUEBOX\_SelectiveRfOnOff

<b>Reader:</b>	BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.
<b>Description:</b>	This function sets selectively RF ON/OFF.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Antenna: The antenna to switch ON / OFF. [out] OnOff: Flag to activate/deactivate the RF. Use one of the values listed below: <ul style="list-style-type: none"> <li>• = 0: To deactivate the RF.</li> <li>• != 0: To activate the RF.</li> </ul>
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> </ul>
<b>Syntax:</b>	BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_SelectiveRfOnOff (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, short OnOff);

#### 2.5.34 BLUEBOX\_ReadID\_EM4305

<b>Name:</b>	BLUEBOX_ReadID_EM4305
<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to read the ID of an EM4305 tag.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h. <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> </ul>

- BLUEBOX\_ANT\_2: Antenna 2.

[out] TagType: One of the values listed below and defined in BLUEBOX\_TagType enum in BLUEBOXLib.h:

- BLUEBOX\_EM4305: EM4305.

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadID_EM4305 (BLUEBOX_Handle *Handle,
BLUEBOX_Antenna Antenna, BLUEBOX_TagType
*TagType, void *Data);
```

**2.5.35 BLUEBOX\_Write\_EM4305**

**Name:**

BLUEBOX\_Write\_EM4305

**Reader:**

BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to writes a EM4305 tag with one of the codes defined below:

- BLUEBOX SHORT: 5 bytes, UNIQUE equivalent.
- BLUEBOX MEDIUM: 10 bytes.
- BLUEBOX LONG: 20 bytes.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined

in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Data: The data to write in the tag's memory.

[in] Length: The number of bytes to write. Allowed values are 5, 10 and 20.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:**

BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Write\_EM4305 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Data, int Length);

### 2.5.36 BLUEBOX\_ReadID\_T5557

**Name:**

BLUEBOX\_ReadID\_T5557

**Reader:**

BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read the ID of a T5557 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] TagType: One of the values listed below and defined in BLUEBOX\_TagType enum in BLUEBOXLib.h:

- BLUEBOX\_T5557: T5557.

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadID_T5557 (BLUEBOX_Handle *Handle,
BLUEBOX_Antenna Antenna, BLUEBOX_TagType
*TagType, void *Data);
```

### 2.5.37 BLUEBOX\_Write\_T5557

**Name:**

BLUEBOX\_Write\_T5557

**Reader:**

BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to writes a T5557 tag with one of the codes defined below:

- BLUEBOX SHORT: 5 bytes, UNIQUE equivalent.
- BLUEBOX MEDIUM: 10 bytes.
- BLUEBOX LONG: 20 bytes.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Data: The data to write in the tag's memory.

[in] Length: The number of bytes to write. Allowed values are 5, 10 and 20.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Write\_T5557 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Data, int Length);

### 2.5.38 BLUEBOX\_ReadID\_Q5

**Name:** BLUEBOX\_ReadID\_Q5

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read the ID of a Q5 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] TagType: One of the values listed below and defined in BLUEBOX\_TagType enum in BLUEBOXLib.h:

- BLUEBOX\_Q5: Q5.

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadID\_Q5 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, BLUEBOX\_TagType  
\*TagType, void \*Data);

#### 2.5.39 BLUEBOX\_Write\_Q5

**Name:** BLUEBOX\_Write\_Q5

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to writes a Q5 tag with one of the codes defined below:

- BLUEBOX SHORT: 5 bytes, UNIQUE equivalent.
- BLUEBOX MEDIUM: 10 bytes.
- BLUEBOX LONG: 20 bytes.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Data: The data to write in the tag's memory.

[in] Length: The number of bytes to write. Allowed values are 5, 10 and 20.

**Return:** An error code about the execution of the function. One of the values listed below and defined in



BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Write\_Q5 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Data, int Length);

#### 2.5.40 BLUEBOX\_ReadID\_HITAG1

**Name:** BLUEBOX\_ReadID\_HITAG1

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read the ID of a HITAG 1 tag (UID).

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] TagType: One of the values listed below and defined in BLUEBOX\_TagType enum in BLUEBOXLib.h:

- BLUEBOX\_HITAG\_1: HITAG 1.

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.

- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadID\_HITAG1 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, BLUEBOX\_TagType  
\*TagType, void \*Data);

#### 2.5.41 BLUEBOX\_ReadPage\_HITAG1

**Name:** BLUEBOX\_ReadPage\_HITAG1

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read a page of a HITAG 1 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Page: The page of the tag's memory to read (16 – 63).

[in] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.

- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadPage\_HITAG1 (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, void \*Id, int Page,  
void \*Data);

#### 2.5.42 BLUEBOX\_WritePage\_HITAG1

**Name:** BLUEBOX\_WritePage\_HITAG1

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to write a page of a HITAG 1 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Page: The page of the tag's memory to write (16 – 63).

[in] Data: The data written to the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_WritePage\_HITAG1 (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, void \*Id, int Page,  
void \*Data);

#### 2.5.43 BLUEBOX\_ReadID\_HITAGS

**Name:** BLUEBOX\_ReadID\_HITAGS

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read the ID of a HITAG S tag (UID).

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] TagType: One of the values listed below and defined in BLUEBOX\_TagType enum in BLUEBOXLib.h:

- BLUEBOX\_HITAG\_S256: HITAG S 256.
- BLUEBOX\_HITAG\_S2048: HITAG S 2048.

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadID\_HITAGS (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, BLUEBOX\_TagType  
\*TagType, void \*Data);

#### 2.5.44 BLUEBOX\_Write\_HITAGS

**Name:** BLUEBOX\_Write\_HITAGS

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to writes a HITAG S tag with one of the codes defined below:

- BLUEBOX SHORT: 5 bytes, UNIQUE equivalent.
- BLUEBOX MEDIUM: 10 bytes.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Data: The data to write in the tag's memory.

[in] Length: The number of bytes to write. Allowed values are 5 and 10.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Write\_HITAGS (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Data, int Length);

#### 2.5.45 BLUEBOX\_ReadPage\_HITAGS

**Name:** BLUEBOX\_ReadPage\_HITAGS

**Reader:** BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read a page of a HITAG S tag.

**Parameters:**

- [in] Handle: The handle that identifies the reader.
- [in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.
  - BLUEBOX\_ANT\_1: Antenna 1.
  - BLUEBOX\_ANT\_2: Antenna 2.
- [in] Id: The ID of the tag to read.
- [in] Page: The page of the tag's memory to read (0 – 63).
- [in] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadPage\_HITAGS (BLUEBOX\_Handle \*Handle, BLUEBOX\_Antenna Antenna, void \*Id, int Page, void \*Data);

## 2.5.46 BLUEBOX\_WritePage\_HITAGS

<b>Name:</b>	BLUEBOX_WritePage_HITAGS
<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to write a page of a HITAG S tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> <li>• BLUEBOX_ANT_2: Antenna 2.</li> </ul> <p>[in] Id: The ID of the tag to write.</p> <p>[in] Page: The page of the tag's memory to write (0 – 63).</p> <p>[in] Data: The data written to the tag's memory.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_WritePage_HITAGS (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Id, int Page, void *Data);</pre>

## 2.5.47 BLUEBOX\_ReadID\_TITAN

<b>Name:</b>	BLUEBOX_ReadID_TITAN
<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to read the ID of a TITAN tag (ID + SN).
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to read the tag's ID. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> <li>• BLUEBOX_ANT_2: Antenna 2.</li> </ul> <p>[out] TagType: One of the values listed below and defined in BLUEBOX_TagType enum in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_TITAN: TITAN.</li> </ul> <p>[out] Data: The data read from the tag's memory.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_ReadID_TITAN (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, BLUEBOX_TagType *TagType, void *Data);</pre>

#### 2.5.48 BLUEBOX\_Reset\_TITAN

**Name:** BLUEBOX\_Reset\_TITAN



<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to reset a TITAN tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to reset the tag. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> <li>• BLUEBOX_ANT_2: Antenna 2.</li> </ul>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_Reset_TITAN (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna);</pre>

#### 2.5.49 BLUEBOX\_Login\_TITAN

<b>Name:</b>	BLUEBOX_Login_TITAN
<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

<b>Description:</b>	This function allows to login a TITAN tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to login the tag. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> <li>• BLUEBOX_ANT_2: Antenna 2.</li> </ul> <p>[in] Password: The password to use to login the tag.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_Login_TITAN (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Password);</pre>

#### 2.5.50 BLUEBOX\_WritePassword\_TITAN

<b>Name:</b>	BLUEBOX_WritePassword_TITAN
<b>Reader:</b>	BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to write the password of a TITAN tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to write the tag's password. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> </ul>

- BLUEBOX\_ANT\_2: Antenna 2.

[in] OldPwd: The old password of the tag.

[in] NewPwd: The new password of the tag.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WritePassword_TITAN (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *OldPwd, void
*NewPwd);
```

**2.5.51 BLUEBOX\_SelectiveRead\_TITAN**

**Name:**

BLUEBOX\_SelectiveRead\_TITAN

**Reader:**

BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to selective read of a TITAN tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Address: The address of the memory to read.

[in] Length: The number of words (4 bytes length) to read from the tag's memory.

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_SelectiveRead_TITAN (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, int Address, int
Length, void *Data);
```

## 2.5.52 BLUEBOX\_SelectiveWrite\_TITAN

**Name:**

BLUEBOX\_SelectiveWrite\_TITAN

**Reader:**

BLUEBOX DESKTOP LF, BLUEBOX OEM LF, BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to selective write of a TITAN tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Address: The address of the memory to write.

[in] Length: The number of words (4 bytes length) to write to the tag's memory.

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of

the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_SelectiveWrite\_TITAN (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, int Address, int  
Length, void \*Data);

### 2.5.53 BLUEBOX\_Inventory\_ISO15693

**Name:** BLUEBOX\_Inventory\_ISO15693

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAD CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function sends an inventory command with anticollision to read all the ISO 15693 tags. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to inventory the tag's. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.
- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] UseAFI: Flag to use the AFI in inventory procedure. Use one of the values listed below:

- == 0: To not use AFI.
- != 0: To use AFI.

[in] AFI: The AFI field to use in inventory procedure. It's an optional parameter needed when UseAFI != 0.

[out] Tags: The array containing the tags read.

[out] TagsNo: The number of tags in the array.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_Inventory_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, short UseAFI,
unsigned char AFI, BLUEBOX_Tag **Tags, int *TagsNo);
```

**2.5.54 BLUEBOX\_ReadPage\_ISO15693**

**Name:**

BLUEBOX\_ReadPage\_ISO15693

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAD CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read a page of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined

in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.
- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Page: The page of the tag's memory to read (0 – 255).

[in] Size: The size of the page to read in bytes (4, 8).

[out] Data: The data read from the tag's memory.

#### Return:

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

#### Syntax:

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadPage_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Page,
int Size, void *Data);
```

### 2.5.55 BLUEBOX\_ReadMultiPage\_ISO15693

**Name:** BLUEBOX\_ReadMultiPage\_ISO15693

**Reader:** BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL,  
BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL.

**Description:** This function allows to read multi pages of a ISO 15693 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Page: The page of the tag's memory to read (0 – 255).

[in] Size: The size of the page to read in bytes (4, 8).

[in] Count: The number of pages to read.

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadMultiPage_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Page,
int Size, int Count, void *Data);
```

**2.5.56 BLUEBOX\_WritePage\_ISO15693**

**Name:**

BLUEBOX\_WritePage\_ISO15693

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAD CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to write a page of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's



memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.
- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Page: The page of the tag's memory to write (0 – 255).

[in] Size: The size of the page to write in bytes (4, 8).

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WritePage_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Page,
int Size, void *Data);
```

**2.5.57 BLUEBOX\_WriteMultiPage\_ISO15693**

**Name:**

BLUEBOX\_WriteMultiPage\_ISO15693

**Reader:**

BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL,  
BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE  
CHANNEL.

**Description:**

This function allows to write a page of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Page: The page of the tag's memory to write (0 – 255).

[in] Size: The size of the page to write in bytes (4, 8).

[in] Count: The number of pages to write.

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteMultiPage_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Page,
int Size, int Count, void *Data);
```

**2.5.58 BLUEBOX\_LockPage\_ISO15693**

**Name:**

BLUEBOX\_LockPage\_ISO15693

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAD CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to lock a page of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to lock the tag's

memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_NOANT: No antenna info. Use only with quad channel readers.
- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to lock.

[in] Page: The page of the tag's memory to lock (0 – 255).

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_LockPage_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int
Page);
```

**2.5.59 BLUEBOX\_Write\_AFI\_ISO15693**

**Name:**

BLUEBOX\_Write\_AFI\_ISO15693

**Reader:**

BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL,  
BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE  
CHANNEL.

**Description:**

This function allows to write the AFI of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag to write.

[in] Afi: The tag's AFI to be written.

**Return:**

An error code about the execution of the function. One of

the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_Write_AFI_ISO15693 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, unsigned
char Afi);
```

**2.5.60 BLUEBOX\_Lock\_AFI\_ISO15693**

**Name:**

BLUEBOX\_Lock\_AFI\_ISO15693

**Reader:**

BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL,  
BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE  
CHANNEL.

**Description:**

This function allows to lock the AFI of a ISO 15693 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to lock the tag's  
memory. Use one of the values listed below and defined  
in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag to lock.

**Return:**

An error code about the execution of the function. One of  
the values listed below and defined in  
BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Lock\_AFI\_ISO15693 (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, void \*Id);

#### 2.5.61 BLUEBOX\_GetRandomNumber\_ICODE\_SLI\_S

**Name:** BLUEBOX\_GetRandomNumber\_ICODE\_SLI\_S

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to get a random number from an ICODE SLI-S tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.  
[out] Random: The random number received from the tag.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GetRandomNumber\_ICODE\_SLI\_S  
(BLUEBOX\_Handle \*Handle, BLUEBOX\_Antenna Antenna,  
void \*Id, void \*Random);

## 2.5.62 BLUEBOX\_SetPassword\_ICODE\_SLI\_S

<b>Name:</b>	BLUEBOX_SetPassword_ICODE_SLI_S
<b>Reader:</b>	BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to transmit a password to an ICODE SLI-S tag to get access to the different protected functionalities of the tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> </ul> <p>[in] Id: The ID of the tag.</p> <p>[in] PwdId: The password identifier which identifies the type of the password to transmit.</p> <p>[in] Password: The password to transmit to the tag.</p> <p>[in] Random: The random number previously received from the tag.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_SetPassword_ICODE_SLI_S (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Id, BLUEBOX_ICODE_SLI_S_PasswordIdentifier PwdId, void *Password, void *Random);</pre>

### 2.5.63 BLUEBOX\_WritePassword\_ICODE\_SLI\_S

<b>Name:</b>	BLUEBOX_WritePassword_ICODE_SLI_S
<b>Reader:</b>	BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to write a new password to an ICODE SLI-S tag if the related old password has already been transmitted with a Set Password command before and the addressed password is not locked.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> </ul> <p>[in] Id: The ID of the tag.</p> <p>[in] PwdId: The password identifier which identifies the type of the password to write.</p> <p>[in] Password: The new password to write to the tag.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_WritePassword_ICODE_SLI_S (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Id, BLUEBOX_ICODE_SLI_S_PasswordIdentifier PwdId, void *Password);</pre>

### 2.5.64 BLUEBOX\_LockPassword\_ICODE\_SLI\_S

<b>Name:</b>	BLUEBOX_LockPassword_ICODE_SLI_S
<b>Reader:</b>	BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to lock a password of an ICODE SLI-S tag if the related old password has already been transmitted with a Set Password command before and the addressed password is not locked.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> </ul> <p>[in] Id: The ID of the tag.</p> <p>[in] PwdId: The password identifier which identifies the type of the password to lock.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_LockPassword_ICODE_SLI_S (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Id, BLUEBOX_ICODE_SLI_S_PasswordIdentifier PwdId);</pre>

2.5.65	BLUEBOX_64BitPasswordProtection_ICODE_SLI_S
--------	---

<b>Name:</b>	BLUEBOX_64BitPasswordProtection_ICODE_SLI_S
<b>Reader:</b>	BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF



SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to activate the 64-bit password protection. This mode can be enabled if both the Read and Write password have already been transmitted with a Set Password command before.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_64BitPasswordProtection\_ICODE\_SLI\_S  
(BLUEBOX\_Handle \*Handle, BLUEBOX\_Antenna Antenna,  
void \*Id);

#### 2.5.66 BLUEBOX\_ProtectPage\_ICODE\_SLI\_S

**Name:** BLUEBOX\_ProtectPage\_ICODE\_SLI\_S

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to change the protection condition of a page of an ICODE SLI-S if the related passwords have

already been transmitted with a Set Password command before and the addressed page is not locked.

**Parameters:**

[in] Handle: The handle that identifies the reader.  
 [in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

[in] PageNo: The number of the page.

[in] Status: The protection status. Use one of the values defined in BLUEBOX\_ICODE\_SLI\_S\_ProtectionStatus enum.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ProtectPage_ICODE_SLI_S (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int
PageNo, BLUEBOX_ICODE_SLI_S_ProtectionStatus
Status);
```

2.5.67	BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S
--------	---

**Name:**

BLUEBOX\_LockPageProtectionCondition\_ICODE\_SLI\_S

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to lock the page protection condition of a page of an ICOE SLI-S if the related passwords have already been transmitted with a Set Password command

before.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

[in] PageNo: The number of the page.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S
(BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna,
void *Id, int PageNo);
```

**2.5.68**

**BLUEBOX\_GetMultipleBlockProtectionStatus\_ICODE\_SLI\_S**

**Name:**

BLUEBOX\_GetMultipleBlockProtectionStatus\_ICODE\_SLI\_S

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to get the block protection status of the requested blocks of an ICODE SLI-S.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

[in] BlockNo: The number of the first block.

[in] Length: The number of blocks.

[out] Status: The block protection status of the requested blocks array.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S
(BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna,
void *Id, int BlockNo, int Length,
BLUEBOX_ICODE_SLI_S_BlockProtectionStatus *Status);
```

2.5.69      BLUEBOX_Destroy_SLI_S_ICODE_SLI_S
---

**Name:**

BLUEBOX\_Destroy\_SLI\_S\_ICODE\_SLI\_S

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to destroy an ICODE SLI-S tag. It can be destroyed if the Destroy SLI-S password has been transmitted before. This command is irreversible.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in

BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Destroy\_SLI\_S\_ICODE\_SLI\_S  
(BLUEBOX\_Handle \*Handle, BLUEBOX\_Antenna Antenna,  
void \*Id);

#### 2.5.70 BLUEBOX\_EnablePrivacy\_ICODE\_SLI\_S

**Name:** BLUEBOX\_EnablePrivacy\_ICODE\_SLI\_S

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to set an ICODE SLI-S into Privacy mode if the Privacy password has already been transmitted before.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

[in] Id: The ID of the tag.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.

- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
 BLUEBOX\_EnablePrivacy\_ICODE\_SLI\_S  
 (BLUEBOX\_Handle \*Handle, BLUEBOX\_Antenna Antenna,  
 void \*Id);

#### 2.5.71 BLUEBOX\_Inventory\_ISO14443A

**Name:** BLUEBOX\_Inventory\_ISO14443A

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function sends an inventory command with anticollision to read all ISO 14443A tags. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
 [in] Antenna: The antenna to use to inventory tags. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] Tags: The array containing the tags read.  
 [out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Inventory\_ISO14443A (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

**Remarks** Read only one ISO 14443A tag.

## 2.5.72 BLUEBOX\_ReadBlock\_MIFARE\_1k

**Name:** BLUEBOX\_ReadBlock\_MIFARE\_1k

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read a block of memory of a MIFARE 1k tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] KeyType: The key type to use. Use one of the values listed below and defined in BLUEBOX\_MifareKey enum in BLUEBOXLib.h.

- BLUEBOX\_MIFARE\_KEY\_A: Key A.
- BLUEBOX\_MIFARE\_KEY\_B: Key B.

[in] Key: The key to use to read the tag's memory.

[in] Block: The page of the tag's memory to read (0 – 63).

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.

- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadBlock\_MIFARE\_1k (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, void \*Id,  
BLUEBOX\_MifareKey KeyType, void \*Key, int Block, void  
\*Data);

#### 2.5.73 BLUEBOX\_WriteBlock\_MIFARE\_1k

**Name:** BLUEBOX\_WriteBlock\_MIFARE\_1k

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to write a block of memory of a MIFARE 1k tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] KeyType: The key type to use. Use one of the values listed below and defined in BLUEBOX\_MifareKey enum in BLUEBOXLib.h.

- BLUEBOX\_MIFARE\_KEY\_A: Key A.
- BLUEBOX\_MIFARE\_KEY\_B: Key B.

[in] Key: The key to use to write the tag's memory.

[in] Block: The page of the tag's memory to write (0 – 63).

[in] Data: The data to write to the tag's memory.

**Return:** An error code about the execution of the function. One of



the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteBlock_MIFARE_1k (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id,
BLUEBOX_MifareKey KeyType, void *Key, int Block, void
*Data);
```

2.5.74 <a href="#">BLUEBOX_ReadBlock_MIFARE_4k</a>
--

**Name:**

BLUEBOX\_ReadBlock\_MIFARE\_4k

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read a block of memory of a MIFARE 4k tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] KeyType: The key type to use. Use one of the values listed below and defined in BLUEBOX\_MifareKey enum in BLUEBOXLib.h.

- BLUEBOX\_MIFARE\_KEY\_A: Key A.

- BLUEBOX\_MIFARE\_KEY\_B: Key B.

[in] Key: The key to use to read the tag's memory.

[in] Block: The page of the tag's memory to read (0 – 255).

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadBlock_MIFARE_4k (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id,
BLUEBOX_MifareKey KeyType, void *Key, int Block, void
*Data);
```

**2.5.75 BLUEBOX\_WriteBlock\_MIFARE\_4k**

**Name:**

BLUEBOX\_WriteBlock\_MIFARE\_4k

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to write a block of memory of a MIFARE 4k tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] KeyType: The key type to use. Use one of the values listed below and defined in BLUEBOX\_MifareKey enum in BLUEBOXLib.h.

- BLUEBOX\_MIFARE\_KEY\_A: Key A.
- BLUEBOX\_MIFARE\_KEY\_B: Key B.

[in] Key: The key to use to write the tag's memory.

[in] Block: The page of the tag's memory to write (0 – 255).

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteBlock_MIFARE_4k (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id,
BLUEBOX_MifareKey KeyType, void *Key, int Block, void
*Data);
```

2.5.76	BLUEBOX_ReadBlock_MIFARE_Ultralight
--------	-------------------------------------

**Name:**

BLUEBOX\_ReadBlock\_MIFARE\_Ultralight

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read a block of memory of a MIFARE Ultralight tag.

<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX_Antenna enum in BLUEBOXLib.h.</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ANT_1: Antenna 1.</li> <li>• BLUEBOX_ANT_2: Antenna 2.</li> </ul> <p>[in] Id: The ID of the tag to read.</p> <p>[in] Block: The page of the tag's memory to read (0 – 15).</p> <p>[out] Data: The data read from the tag's memory.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_ReadBlock_MIFARE_Ultralight (BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna, void *Id, int Block, void *Data);</pre>

## 2.5.77 BLUEBOX\_WriteBlock\_MIFARE\_Ultralight

<b>Name:</b>	BLUEBOX_WriteBlock_MIFARE_Ultralight
<b>Reader:</b>	BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.
<b>Description:</b>	This function allows to write a block of memory of a MIFARE Ultralight tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Antenna: The antenna to use to write the tag's</p>

memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Block: The page of the tag's memory to write (0 – 15).

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteBlock_MIFARE_Ultralight
(BLUEBOX_Handle *Handle, BLUEBOX_Antenna Antenna,
void *Id, int Block, void *Data);
```

<b>2.5.78      BLUEBOX_ReadBlock_NTAG213</b>
--

**Name:**

BLUEBOX\_ReadBlock\_NTAG213

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read a block of memory of a NTAG213 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Block: The page of the tag's memory to read (0 – 44).

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadBlock_NTAG213 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Block,
void *Data);
```

**2.5.79 BLUEBOX\_WriteBlock\_NTAG213**

**Name:**

BLUEBOX\_WriteBlock\_NTAG213

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to write a block of memory of a NTAG213 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.

- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Block: The page of the tag's memory to write (0 – 44).

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteBlock_NTAG213 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Block,
void *Data);
```

**2.5.80 BLUEBOX\_ReadBlock\_NTAG215**

**Name:** BLUEBOX\_ReadBlock\_NTAG215

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read a block of memory of a NTAG215 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Block: The page of the tag's memory to read (0 – 134).

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ReadBlock_NTAG215 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Block,
void *Data);
```

2.5.81	BLUEBOX_WriteBlock_NTAG215
--------	----------------------------

**Name:**

BLUEBOX\_WriteBlock\_NTAG215

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to write a block of memory of a NTAG215 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.

[in] Block: The page of the tag's memory to write (0 –



134).

[in] Data: The data to write to the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_WriteBlock_NTAG215 (BLUEBOX_Handle
*Handle, BLUEBOX_Antenna Antenna, void *Id, int Block,
void *Data);
```

**2.5.82 BLUEBOX\_ReadBlock\_NTAG216**

**Name:**

BLUEBOX\_ReadBlock\_NTAG216

**Reader:**

BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:**

This function allows to read a block of memory of a NTAG216 tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Block: The page of the tag's memory to read (0 – 230).

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadBlock\_NTAG216 (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Antenna Antenna, void \*Id, int Block,  
void \*Data);

2.5.83      BLUEBOX_WriteBlock_NTAG216
--

**Name:** BLUEBOX\_WriteBlock\_NTAG216

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to write a block of memory of a NTAG216 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.  
[in] Block: The page of the tag's memory to write (0 – 230).  
[in] Data: The data to write to the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in

BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
 BLUEBOX\_WriteBlock\_NTAG216 (BLUEBOX\_Handle  
 \*Handle, BLUEBOX\_Antenna Antenna, void \*Id, int Block,  
 void \*Data);

#### 2.5.84 BLUEBOX\_Inventory\_ISO14443B

**Name:** BLUEBOX\_Inventory\_ISO14443B

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function sends an inventory command with anticollision to read all ISO 14443B tags. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
 [in] Antenna: The antenna to use to inventory tags. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] Tags: The array containing the tags read.  
 [out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Inventory\_ISO14443B (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

**Remarks** Read only one ISO 14443B tag.

#### 2.5.85 BLUEBOX\_ReadBlock\_SR176

**Name:** BLUEBOX\_ReadBlock\_SR176

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to read a block of memory of a SR176 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to read the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to read.

[in] Block: The page of the tag's memory to read (0 – 63).

[out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.

- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadBlock\_SR176 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Id, int Block, void  
\*Data);

#### 2.5.86 BLUEBOX\_WriteBlock\_SR176

**Name:** BLUEBOX\_WriteBlock\_SR176

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function allows to write a block of memory of a SR176 tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to write the tag's memory. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[in] Id: The ID of the tag to write.  
[in] Block: The page of the tag's memory to write (0 – 63).  
[in] Data: The data to write to the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.

- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_WriteBlock\_SR176 (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_Antenna Antenna, void \*Id, int Block, void  
\*Data);

#### 2.5.87 BLUEBOX\_Inventory\_PICOPASS

**Name:** BLUEBOX\_Inventory\_PICOPASS

**Reader:** BLUEBOX DESKTOP HF, BLUEBOX OEM HF, BLUEBOX OEM HF E, BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL.

**Description:** This function sends an inventory command with anticollision to read all Picopass tags. The tags array contain all the tag ID and other information related to every tag such as antenna and ID length.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Antenna: The antenna to use to inventory tags. Use one of the values listed below and defined in BLUEBOX\_Antenna enum in BLUEBOXLib.h.

- BLUEBOX\_ANT\_1: Antenna 1.
- BLUEBOX\_ANT\_2: Antenna 2.

[out] Tags: The array containing the tags read.  
[out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.

- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Inventory\_PICOPASS (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

#### 2.5.88 BLUEBOX\_ReadRfParameters

**Name:** BLUEBOX\_ReadRfParameters

**Reader:** BLUEBOX INDUSTRIAL HF MID/LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAND CHANNEL, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL ACTIVE, BLUEBOX PORTAL UHF, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 UHF MID RANGE SINGLE CHANNEL.

**Description:** This function reads the RF parameters of the reader.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Parameters: RF parameters set in the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadRfParameters (BLUEBOX\_Handle \*Handle,  
unsigned char \*Parameters);

**Remarks** See the reader technical manual for the Parameters format.  
This functions could be replaced with BLUEBOX\_ReadConfiguration (2.5.15).

## 2.5.89 BLUEBOX\_WriteRfParameters

<b>Name:</b>	BLUEBOX_WriteRfParameters
<b>Reader:</b>	BLUEBOX INDUSTRIAL HF MID/LONG RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL HF LONG RANGE QUAND CHANNEL, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL ACTIVE, BLUEBOX PORTAL UHF, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 UHF MID RANGE SINGLE CHANNEL.
<b>Description:</b>	This function writes the RF parameters of the reader.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Parameters: RF parameters to be set in the reader.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> </ul>
<b>Syntax:</b>	BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_WriteRfParameters (BLUEBOX_Handle *Handle, unsigned char *Parameters);
<b>Remarks</b>	See the reader technical manual for the Parameters format. This functions could be replaced with BLUEBOX_ReadConfiguration (2.5.16).

## 2.5.90 BLUEBOX\_Inventory\_ISO18K6B

<b>Name:</b>	BLUEBOX_Inventory_ISO18K6B
<b>Reader:</b>	BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL.
<b>Description:</b>	This function sends an inventory command with anticollision to read all the ISO 18000-6B tags.



**Parameters:** [in] Handle: The handle that identifies the reader.  
 [out] Tags: The array containing the tags read.  
 [out] TagsNo: The number of tags in the array.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
 BLUEBOX\_Inventory\_ISO18K6B (BLUEBOX\_Handle  
 \*Handle, BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

#### 2.5.91 BLUEBOX\_Read\_ISO18K6B

**Name:** BLUEBOX\_Read\_ISO18K6B

**Reader:** BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL.

**Description:** This function allows to read the memory of a ISO 18000-6B tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
 [in] Uid: The UID of the tag to be read.  
 [in] Address: The starting address of the tag's memory to be read.  
 [in] Nblocks: The number of 8-bytes blocks to be read (1 ... 8).  
 [out] Data: The data read from the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Read\_ISO18K6B (BLUEBOX\_Handle \*Handle,  
void \*Uid, void \*Pwd, BLUEBOX\_ISO18K6C\_Bank Bank,  
int Address, int Length, void \*Data);

#### 2.5.92 BLUEBOX\_Write\_ISO18K6B

**Name:** BLUEBOX\_Write\_ISO18K6B

**Reader:** BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL.

**Description:** This function allows to write the memory of a ISO 18000-6B tag.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Uid: The UID of the tag to be written.  
[in] Address: The starting address of the tag's memory to be written.  
[in] Length: The number of bytes to be written (1 ... 32).  
[in] Data: The data to be written into the tag's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Write\_ISO18K6B (BLUEBOX\_Handle \*Handle,  
void \*Uid, int Address, int Length, void \*Data);

#### 2.5.93 BLUEBOX\_Inventory\_ISO18K6C

**Name:** BLUEBOX\_Inventory\_ISO18K6C

**Reader:** BLUEBOX OEM UHF, BLUEBOX INDUSTRIAL UHF MID

RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE DUAL CHANNEL, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

- Description:** This function sends an inventory command with anticollision to read all the ISO 18000-6C tags.
- Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Tags: The array containing the tags read.  
[out] TagsNo: The number of tags in the array.
- Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:
- BLUEBOX\_StatusOk.
  - BLUEBOX\_InvalidHandle.
  - BLUEBOX\_ConnectionError.
  - BLUEBOX\_InvalidCommand.
  - BLUEBOX\_TimeoutError.
  - BLUEBOX\_CommunicationError.
  - BLUEBOX\_InvalidParams.
  - BLUEBOX\_TagNotFound.
- Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_Inventory\_ISO18K6C (BLUEBOX\_Handle  
\*Handle, BLUEBOX\_Tag \*\*Tags, int \*TagsNo);

#### 2.5.94 BLUEBOX\_ProgramEPC\_ISO18K6C

- Name:** BLUEBOX\_ProgramEPC\_ISO18K6C
- Reader:** BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.
- Description:** This function allows to write the EPC of a ISO 18000-6C tag.
- Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Uid: The UID of the tag to be written.  
[in] Pwd: The access password to write the tag. Set to 0

if no password is required.

[in] Length: The number of 2-bits words to be written.

[in] Data: The data to be written into the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_ProgramEPC_ISO18K6C (BLUEBOX_Handle
*Handle, void *Uid, void *Pwd, int Length, void *Data);
```

### 2.5.95 BLUEBOX\_Read\_ISO18K6C

**Name:**

BLUEBOX\_Read\_ISO18K6C

**Reader:**

BLUEBOX OEM UHF, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE DUAL CHANNEL, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

**Description:**

This function allows to read the memory of a ISO 18000-6C tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Uid: The UID of the tag to be read.

[in] Pwd: The access password to read the tag. Set to 0 if no password is required.

[in] Bank: The memory bank to be read. One of the values listed below and defined in

BLUEBOX\_ISO18K6C\_Bank in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_BANK\_RESERVED: Reserved.
- BLUEBOX\_ISO18K6C\_BANK\_EPC: EPC.
- BLUEBOX\_ISO18K6C\_BANK\_TID: TID.
- BLUEBOX\_ISO18K6C\_BANK\_USER: User.

[in] Address: The starting address of the tag's memory to be read.

[in] Length: The number of 16-bits words to be read.

[out] Data: The data read from the tag's memory.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_Read_ISO18K6C (BLUEBOX_Handle *Handle,
void *Uid, void *Pwd, BLUEBOX_ISO18K6C_Bank Bank,
int Address, int Length, void *Data);
```

**2.5.96 BLUEBOX\_Write\_ISO18K6C**

**Name:**

BLUEBOX\_Write\_ISO18K6C

**Reader:**

BLUEBOX OEM UHF, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE DUAL CHANNEL, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

<b>Description:</b>	This function allows to write the memory of a ISO 18000-6C tag.
<b>Parameters:</b>	<p>[in] Handle: The handle that identifies the reader.</p> <p>[in] Uid: The UID of the tag to be written.</p> <p>[in] Pwd: The access password to write the tag. Set to 0 if no password is required.</p> <p>[in] Bank: The memory bank to be written. One of the values listed below and defined in BLUEBOX_ISO18K6C_Bank in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ISO18K6C_BANK_RESERVED: Reserved.</li> <li>• BLUEBOX_ISO18K6C_BANK_EPC: EPC.</li> <li>• BLUEBOX_ISO18K6C_BANK_TID: TID.</li> <li>• BLUEBOX_ISO18K6C_BANK_USER: User.</li> </ul> <p>[in] Address: The starting address of the tag's memory to be written.</p> <p>[in] Length: The number of 2-bits words to be written.</p> <p>[in] Data: The data to be written into the tag's memory.</p>
<b>Return:</b>	<p>An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> <li>• BLUEBOX_InvalidParams.</li> <li>• BLUEBOX_TagNotFound.</li> <li>• BLUEBOX_TagError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_Write_ISO18K6C (BLUEBOX_Handle *Handle, void *Uid, void *Pwd, BLUEBOX_ISO18K6C_Bank Bank, int Address, int Length, void *Data);</pre>

## 2.5.97 BLUEBOX\_BlockWrite\_ISO18K6C

<b>Name:</b>	BLUEBOX_BlockWrite_ISO18K6C
<b>Reader:</b>	BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.
<b>Description:</b>	This function allows to write the memory of a ISO 18000-

6C tag using the BlockWrite command of the EPC C1G2 (Class-1 Generation-2) standard.

**Parameters:**

- [in] Handle: The handle that identifies the reader.
- [in] Uid: The UID of the tag to be written.
- [in] Pwd: The access password to write the tag. Set to 0 if no password is required.
- [in] Bank: The memory bank to be written. One of the values listed below and defined in BLUEBOX\_ISO18K6C\_Bank in BLUEBOXLib.h:
  - BLUEBOX\_ISO18K6C\_BANK\_RESERVED: Reserved.
  - BLUEBOX\_ISO18K6C\_BANK\_EPC: EPC.
  - BLUEBOX\_ISO18K6C\_BANK\_TID: TID.
  - BLUEBOX\_ISO18K6C\_BANK\_USER: User.
- [in] Address: The starting address of the tag's memory to be written.
- [in] Length: The number of 2-bits words to be written.
- [in] Data: The data to be written into the tag's memory.

**Return:**

- An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:
- BLUEBOX\_StatusOk.
  - BLUEBOX\_InvalidHandle.
  - BLUEBOX\_ConnectionError.
  - BLUEBOX\_InvalidCommand.
  - BLUEBOX\_TimeoutError.
  - BLUEBOX\_CommunicationError.
  - BLUEBOX\_InvalidParams.
  - BLUEBOX\_TagNotFound.
  - BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_BlockWrite_ISO18K6C (BLUEBOX_Handle
*Handle, void *Uid, void *Pwd,
BLUEBOX_ISO18K6C_Bank Bank, int Address, int Length,
void *Data);
```

2.5.98      [BLUEBOX\\_Lock\\_ISO18K6C](#)

**Name:**

BLUEBOX\_Lock\_ISO18K6C

**Reader:**

BLUEBOX OEM UHF, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL



UHF LONG RANGE DUAL CHANNEL, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

**Description:**

This function allows to lock the password and memory of a ISO 18000-6C tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Uid: The UID of the tag to be written.

[in] Pwd: The access password to write the tag. Must be not 0.

[in] KillPwd: To lock the kill password. One of the values listed below and defined in BLUEBOX\_ISO18K6C\_PasswordPermission in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ACCESSIBLE: Accessible from both opened and secured states.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_ACCESSIBLE: Permanently accessible from both opened and secured states and may never be locked.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_SECURED\_ACCESSIBLE: Accessible only from secured state.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_NOT\_ACCESSIBLE: Not accessible from either opened or secured states.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_NO\_CHANGE: No change.

[in] AccessPwd: To lock the access password. One of the values listed below and defined in BLUEBOX\_ISO18K6C\_PasswordPermission in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ACCESSIBLE: Accessible from both opened and secured states.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_ACCESSIBLE: Permanently accessible from both opened and secured states and may never be locked.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_SECURED\_A



CCESSIBLE: Accessible only from secured state.

- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_ALWAYS\_NOT\_ACCESSIBLE: Not accessible from either opened or secured states.
- BLUEBOX\_ISO18K6C\_TAG\_PWD\_PERM\_NO\_CHANGE: No change.

[in] EPCMemory: To lock the EPC memory. One of the values listed below and defined in BLUEBXO\_ISO18K6C\_MemoryPermission in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_WRITABLE: Writable from both opened and secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_WRITABLE: Permanently writable from both opened and secured states and may never be locked.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_SECURED\_WRITABLE: Writable only from secured state.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_NOT\_WRITABLE: Not writable from either opened or secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_NO\_CHANGE: No change.

[in] TIDMemory: To lock the TID memory. One of the values listed below and defined in BLUEBXO\_ISO18K6C\_MemoryPermission in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_WRITABLE: Writable from both opened and secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_WRITABLE: Permanently writable from both opened and secured states and may never be locked.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_SECURED\_WRITABLE: Writable only from secured state.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_NOT\_WRITABLE: Not writable from either opened or secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_NO\_CHANGE: No change.

[in] UserMemory: To lock the user memory. One of the values listed below and defined in BLUEBXO\_ISO18K6C\_MemoryPermission in BLUEBOXLib.h:

- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_WRITABLE: Writable from both opened and secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_W  
RITABLE: Permanently writable from both opened and secured states and may never be locked.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_SECURED\_  
WRITABLE: Writable only from secured state.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_ALWAYS\_NO  
T\_WRITABLE: Not writable from either opened or secured states.
- BLUEBOX\_ISO18K6C\_TAG\_MEM\_PERM\_NO\_CHANG  
E: No change.

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_Lock_ISO18K6C (BLUEBOX_Handle *Handle,
void *Uid, void *Pwd,
BLUEBOX_ISO18K6C_PasswordPermission KillPwd,
BLUEBOX_ISO18K6C_PasswordPermission AccessPwd,
BLUEBOX_ISO18K6C_MemoryPermission EPCMemory,
BLUEBOX_ISO18K6C_MemoryPermission TIDMemory,
BLUEBOX_ISO18K6C_MemoryPermission UserMemory);
```

**2.5.99 BLUEBOX\_Kill\_ISO18K6C**

**Name:**

BLUEBOX\_Kill\_ISO18K6C

**Reader:**

BLUEBOX OEM UHF, BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL, BLUEBOX INDUSTRIAL UHF LONG RANGE DUAL CHANNEL, BLUEBOX GEN2 OEM UHF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2

INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF MID RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL, BLUEBOX CX UHF LONG RANGE DUAL CHANNEL.

**Description:**

This function allows to kill a ISO 18000-6C tag.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] Uid: The UID of the tag to be killed.

[in] Pwd: The kill password to kill the tag

**Return:**

An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_TagNotFound.
- BLUEBOX\_TagError.

**Syntax:**

```
BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall
BLUEBOX_Kill_ISO18K6C (BLUEBOX_Handle *Handle,
void *Uid, void *Pwd);
```

**2.5.100 BLUEBOX\_FwUpgrade**

**Name:**

BLUEBOX\_FwUpgrade

**Reader:**

All readers.

**Description:**

This function allows to upgrade the BLUEBOX readers firmware.

**Parameters:**

[in] Handle: The handle that identifies the reader.

[in] FileName: The binary file name with the firmware to send to the reader.

[in] Reader: The reader to upgrade. Use one of the values listed below and defined in BLUEBOX\_Reader in BLUEBOXLib.h.

- BLUEBOX\_PRIMARY\_READER: To upgrade the primary reader.

- BLUEBOX\_AUXILIARY\_1\_READER: To upgrade the 1<sup>st</sup> auxiliary reader.
- BLUEBOX\_AUXILIARY\_2\_READER: To upgrade the 2<sup>nd</sup> auxiliary reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.
- BLUEBOX\_GenericError.
- BLUEBOX\_InvalidParams.
- BLUEBOX\_FileError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_FwUpgrade (BLUEBOX\_Handle \*Handle,  
BLUEBOX\_UpgReader Reader, char \*FileName);

#### 2.5.101 BLUEBOX\_ReadNumberOfRegistrations

**Name:** BLUEBOX\_ReadNumberOfRegistrations

**Reader:** BLUEBOX PORTAL UHF.

**Description:** This function reads the number of registrations saved in the reader's memory.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[out] Registrations: The number of registrations saved in the reader's memory.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadNumberOfRegistrations (BLUEBOX\_Handle  
\*Handle, int \*Registrations);

### 2.5.102 BLUEBOX\_ReadOlderRegistration

<b>Name:</b>	BLUEBOX_ReadOlderRegistration
<b>Reader:</b>	BLUEBOX PORTAL UHF.
<b>Description:</b>	This function reads the older registration saved in the reader's memory.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [out] Index: The index of the registration read from reader's memory. [out] Registration: The registration read from the reader's memory.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> <li>• BLUEBOX_TimeoutError.</li> <li>• BLUEBOX_CommunicationError.</li> </ul>
<b>Syntax:</b>	<pre>BLUEBOXLib_API BLUEBOX_ErrorCodes __stdcall BLUEBOX_ReadOlderRegistration (BLUEBOX_Handle *Handle, int *Index, BLUEBOX_Registration *Registration);</pre>

### 2.5.103 BLUEBOX\_CancelOlderRegistration

<b>Name:</b>	BLUEBOX_CancelOlderRegistration
<b>Reader:</b>	BLUEBOX PORTAL UHF.
<b>Description:</b>	This function cancels the older registration saved in the reader's memory.
<b>Parameters:</b>	[in] Handle: The handle that identifies the reader. [in] Index: The index of the registration to cancel from reader's memory.
<b>Return:</b>	An error code about the execution of the function. One of the values listed below and defined in BLUEBOX_ErrorCodes in BLUEBOXLib.h: <ul style="list-style-type: none"> <li>• BLUEBOX_StatusOk.</li> <li>• BLUEBOX_InvalidHandle.</li> <li>• BLUEBOX_ConnectionError.</li> <li>• BLUEBOX_InvalidCommand.</li> </ul>

- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_CancelOlderRegistration (BLUEBOX\_Handle  
\*Handle, int Index);

#### 2.5.104 BLUEBOX\_CancelAllRegistrations

**Name:** BLUEBOX\_CancelAllRegistrations  
**Reader:** BLUEBOX PORTAL UHF.  
**Description:** This function cancels all the registrations saved in the reader's memory.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_CancelAllRegistrations (BLUEBOX\_Handle  
\*Handle);

#### 2.5.105 BLUEBOX\_ReadPreviousRegistration

**Name:** BLUEBOX\_ReadPreviousRegistration  
**Reader:** BLUEBOX PORTAL UHF.  
**Description:** This function reads a previous registration saved in the reader's memory.  
**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Index: The index of the registration to be read from reader's memory.  
[out] Registration: The registrations read from the reader's memory.  
**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.

- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_ReadPreviousRegistration (BLUEBOX\_Handle  
\*Handle, int Index, BLUEBOX\_Registration  
\*Registration);

#### 2.5.106 BLUEBOX\_GenericCommand

**Name:** BLUEBOX\_GenericCommand

**Reader:** All readers.

**Description:** This function sends a generic command using the BLUEBOX protocol.

**Parameters:** [in] Handle: The handle that identifies the reader.  
[in] Command: The command to send to the reader.  
[out] Reply: The reply got from the reader.

**Return:** An error code about the execution of the function. One of the values listed below and defined in BLUEBOX\_ErrorCodes in BLUEBOXLib.h:

- BLUEBOX\_StatusOk.
- BLUEBOX\_InvalidHandle.
- BLUEBOX\_ConnectionError.
- BLUEBOX\_InvalidCommand.
- BLUEBOX\_TimeoutError.
- BLUEBOX\_CommunicationError.

**Syntax:** BLUEBOXLib\_API BLUEBOX\_ErrorCodes \_\_stdcall  
BLUEBOX\_GenericCommand (BLUEBOX\_Handle \*Handle,  
char \*Command, char \*Reply);

### 3 BlueBox Gen1 Functions Table

	BLUEBOX OEM LF	BLUEBOX OEM HF	BLUEBOX OEM HF E	BLUEBOX OEM UHF	BLUEBOX DESKTOP LF	BLUEBOX DESKTOP HF	BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL LF SHORT RANGE DUAL CHANNEL	BLUEBOX INDUSTRIAL LF LONG RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL HF SHORT RANGE DUAL CHANNEL	BLUEBOX INDUSTRIAL HF MID RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL HF LONG RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL HF LONG RANGE QUAD CHANNEL	BLUEBOX INDUSTRIAL UHF MID RANGE SINGLE CHANNEL	BLUEBOX INDUSTRIAL UHF LONG RANGE QUAD CHANNEL	BLUEBOX INDUSTRIAL UHF LONG RANGE DUAL CHANNEL	BLUEBOX INDUSTRIAL ACTIVE	BLUEBOX PORTAL UHF
BLUEBOX_GetSwRelease	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Init	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_End	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Open	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Close	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetAddress	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetDevice	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetDevice	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetFwRelease	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Reset	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetDateTime																			
BLUEBOX_SetDateTime																			
BLUEBOX_ReadParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_WriteParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DefaultParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_ReadSerialNumber																			
BLUEBOX_ReadMACAddress																			
BLUEBOX_ReadConfiguration				✓								✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_WriteConfiguration				✓								✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DefaultConfiguration				✓								✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DataRequest	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_QueueRequest	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_FreeTagsMemory	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_AllocateNotifyChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DeallocateNotifyChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetNotification	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_FreeNotifyMemory	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetOutput	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetReaderStatus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetTemperature																			
BLUEBOX_RfOnOff	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SelectiveRfOnOff																			
BLUEBOX_ReadID_EM4305	✓				✓		✓	✓											
BLUEBOX_Write_EM4305	✓				✓		✓	✓											
BLUEBOX_ReadID_T5557	✓				✓		✓	✓											
BLUEBOX_Write_T5557	✓				✓		✓	✓											
BLUEBOX_ReadID_Q5	✓				✓		✓	✓											
BLUEBOX_WriteQ5	✓				✓		✓	✓											
BLUEBOX_ReadID_HITAG1	✓				✓		✓	✓											
BLUEBOX_ReadPage_HITAG1	✓				✓		✓	✓											
BLUEBOX_WritePage_HITAG1	✓				✓		✓	✓											
BLUEBOX_ReadID_HITAGS	✓				✓		✓	✓											
BLUEBOX_Write_HITAGS	✓				✓		✓	✓											
BLUEBOX_ReadPage_HITAGS	✓				✓		✓	✓											
BLUEBOX_WritePage_HITAGS	✓				✓		✓	✓											
BLUEBOX_ReadID_TITAN	✓				✓		✓	✓											
BLUEBOX_Reset_TITAN	✓				✓		✓	✓											
BLUEBOX_Login_TITAN																			
BLUEBOX_WritePassword_TITAN	✓				✓		✓	✓											
BLUEBOX_SelectiveRead_TITAN					✓		✓	✓											
BLUEBOX_SelectiveWrite_TITAN	✓				✓		✓	✓											



BLUEBOX_Inventory_ISO15693		✓			✓				✓	✓	✓	✓	✓				
BLUEBOX_ReadPage_ISO15693		✓			✓				✓	✓	✓	✓	✓				
BLUEBOX_ReadMultiPage_ISO15693										✓	✓	✓	✓				
BLUEBOX_WritePage_ISO15693		✓			✓				✓	✓	✓	✓	✓				
BLUEBOX_WriteMultiPage_ISO15693											✓	✓	✓				
BLUEBOX_LockPage_ISO15693		✓			✓				✓	✓	✓	✓	✓				
BLUEBOX_Write_AFI_ISO15693											✓	✓	✓				
BLUEBOX_Lock_AFI_ISO15693											✓	✓	✓				
BLUEBOX_GetRandomNumber_ICODE_SLI_S		✓			✓												
BLUEBOX_SetPassword_ICODE_SLI_S		✓			✓												
BLUEBOX_WritePassword_ICODE_SLI_S		✓			✓												
BLUEBOX_LockPassword_ICODE_SLI_S		✓			✓												
BLUEBOX_64BitPasswordProtection_ICODE_SLI_S		✓			✓												
BLUEBOX_ProtectPage_ICODE_SLI_S		✓			✓												
BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S		✓			✓												
BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S		✓			✓												
BLUEBOX_Destroy_SLI_S_ICODE_SLI_S		✓			✓												
BLUEBOX_EnablePrivacy_ICODE_SLI_S		✓			✓												
BLUEBOX_Inevntory_ISO14443A		✓	✓		✓				✓	✓							
BLUEBOX_ReadBlock_MIFARE_1k		✓	✓		✓				✓	✓							
BLUEBOX_WriteBlock_MIFARE_1k		✓	✓		✓				✓	✓							
BLUEBOX_ReadBlock_MIFARE_4k		✓	✓		✓				✓	✓							
BLUEBOX_WriteBlock_MIFARE_4k		✓	✓		✓				✓	✓							
BLUEBOX_ReadBlock_MIFARE_Ultralight		✓	✓		✓				✓	✓							
BLUEBOX_WriteBlock_MIFARE_Ultralight		✓	✓		✓				✓	✓							
BLUEBOX_ReadBlock_NTAG213		✓			✓												
BLUEBOX_WriteBlock_NTAG213		✓			✓												
BLUEBOX_ReadBlock_NTAG215		✓			✓												
BLUEBOX_WriteBlock_NTAG215		✓			✓												
BLUEBOX_ReadBlock_NTAG216		✓			✓												
BLUEBOX_WriteBlock_NTAG216		✓			✓												
BLUEBOX_Inventory_ISO14443B		✓			✓				✓	✓							
BLUEBOX_ReadBlock_SR176		✓			✓				✓	✓							
BLUEBOX_WriteBlock_SR176		✓			✓				✓	✓							
BLUEBOX_ReadRfParameters											✓	✓	✓	✓	✓	✓	✓
BLUEBOX_WriteRfParameters											✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Inventory_ISO18K6B														✓	✓	✓	
BLUEBOX_Read_ISO18K6B														✓	✓	✓	
BLUEBOX_Write_ISO18K6B														✓	✓	✓	
BLUEBOX_Inventory_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_ProgramEPC_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_Read_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_Write_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_BlockWrite_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_Lock_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_Kill_ISO18K6C				✓										✓	✓	✓	
BLUEBOX_FwUpgrade	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_ReadNumberOfRegistrations																	✓
BLUEBOX_ReadOlderRegistration																	✓
BLUEBOX_CancelOlderRegistration																	✓
BLUEBOX_CancelAllRegistrations																	✓
BLUEBOX_ReadPreviousRegistration																	✓
BLUEBOX_GenericCommand	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

## 4 BlueBox Gen2 Functions Table

	BLUEBOX GEN2 DESKTOP LF	BLUEBOX GEN2 DESKTOP HF	BLUEBOX GEN2 DESKTOP UHF	BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL	BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL	BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL	BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL	BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL	BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL	BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL	BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL	BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL	BLUEBOX GEN2 BASIC UHF MID RANGE SINGLE CHANNEL
BLUEBOX_GetSwRelease	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Init	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_End	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Open	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Close	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetAddress	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetDevice	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetDevice	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetFwRelease	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_Reset	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetDateTime				✓	✓	✓	✓	✓	✓				
BLUEBOX_SetDateTime				✓	✓	✓	✓	✓	✓				
BLUEBOX_ReadParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_WriteParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DefaultParameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_ReadSerialNumber				✓	✓	✓	✓	✓	✓				
BLUEBOX_ReadMACAddress				✓	✓	✓	✓	✓	✓				
BLUEBOX_ReadConfiguration			✓	✓	✓	✓	✓	✓	✓				
BLUEBOX_WriteConfiguration			✓	✓	✓	✓	✓	✓	✓				
BLUEBOX_DefaultConfiguration				✓	✓	✓	✓	✓	✓				
BLUEBOX_DataRequest	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_QueueRequest	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_FreeTagsMemory	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_AllocateNotifyChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_DeallocateNotifyChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetNotification	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_FreeNotifyMemory	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SetOutput				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetReaderStatus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_GetTemperature				✓	✓	✓	✓	✓	✓				
BLUEBOX_RfOnOff	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_SelectiveRfOnOff													
BLUEBOX_ReadID_EM4305	✓			✓	✓					✓			
BLUEBOX_Write_EM4305	✓			✓	✓					✓			
BLUEBOX_ReadID_T5557	✓			✓	✓					✓			
BLUEBOX_Write_T5557	✓			✓	✓					✓			
BLUEBOX_ReadID_Q5	✓			✓	✓					✓			
BLUEBOX_WriteQ5	✓			✓	✓					✓			
BLUEBOX_ReadID_HITAG1	✓			✓	✓					✓			
BLUEBOX_ReadPage_HITAG1	✓			✓	✓					✓			
BLUEBOX_WritePage_HITAG1	✓			✓	✓					✓			
BLUEBOX_ReadID_HITAGS	✓			✓	✓					✓			
BLUEBOX_Write_HITAGS	✓			✓	✓					✓			
BLUEBOX_ReadPage_HITAGS	✓			✓	✓					✓			
BLUEBOX_WritePage_HITAGS	✓			✓	✓					✓			
BLUEBOX_ReadID_TITAN	✓			✓	✓					✓			
BLUEBOX_Reset_TITAN	✓			✓	✓					✓			

BLUEBOX_Login_TITAN	✓			✓	✓					✓			
BLUEBOX_WritePassword_TITAN	✓			✓	✓					✓			
BLUEBOX_SelectiveRead_TITAN	✓			✓	✓					✓			
BLUEBOX_SelectiveWrite_TITAN	✓			✓	✓					✓			
BLUEBOX_Inventory_ISO15693		✓				✓	✓	✓			✓		
BLUEBOX_ReadPage_ISO15693		✓				✓	✓	✓			✓		
BLUEBOX_ReadMultiPage_ISO15693													
BLUEBOX_WritePage_ISO15693		✓				✓	✓	✓			✓		
BLUEBOX_WriteMultiPage_ISO15693													
BLUEBOX_LockPage_ISO15693		✓				✓	✓	✓			✓		
BLUEBOX_Write_AFI_ISO15693								✓					
BLUEBOX_Lock_AFI_ISO15693								✓					
BLUEBOX_GetRandomNumber_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_SetPassword_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_WritePassword_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_LockPassword_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_64BitPasswordProtection_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_ProtectPage_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_Destroy_SLI_S_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_EnablePrivacy_ICODE_SLI_S		✓				✓	✓				✓		
BLUEBOX_Inevntory_ISO14443A		✓				✓	✓				✓		
BLUEBOX_ReadBlock_MIFARE_1k		✓				✓	✓				✓		
BLUEBOX_WriteBlock_MIFARE_1k		✓				✓	✓				✓		
BLUEBOX_ReadBlock_MIFARE_4k		✓				✓	✓				✓		
BLUEBOX_WriteBlock_MIFARE_4k		✓				✓	✓				✓		
BLUEBOX_ReadBlock_MIFARE_Ultralight		✓				✓	✓				✓		
BLUEBOX_WriteBlock_MIFARE_Ultralight		✓				✓	✓				✓		
BLUEBOX_ReadBlock_NTAG213		✓				✓	✓				✓		
BLUEBOX_WriteBlock_NTAG213		✓				✓	✓				✓		
BLUEBOX_ReadBlock_NTAG215		✓				✓	✓				✓		
BLUEBOX_WriteBlock_NTAG215		✓				✓	✓				✓		
BLUEBOX_ReadBlock_NTAG216		✓				✓	✓				✓		
BLUEBOX_WriteBlock_NTAG216		✓				✓	✓				✓		
BLUEBOX_Inventory_ISO14443B		✓				✓	✓				✓		
BLUEBOX_ReadBlock_SR176		✓				✓	✓				✓		
BLUEBOX_WriteBlock_SR176		✓				✓	✓				✓		
BLUEBOX_ReadRfParameters			✓					✓	✓			✓	✓
BLUEBOX_WriteRfParameters			✓					✓	✓			✓	✓
BLUEBOX_Inventory_ISO18K6B													
BLUEBOX_Read_ISO18K6B													
BLUEBOX_Write_ISO18K6B													
BLUEBOX_Inventory_ISO18K6C			✓					✓			✓	✓	
BLUEBOX_ProgramEPC_ISO18K6C													
BLUEBOX_Read_ISO18K6C			✓					✓			✓	✓	
BLUEBOX_Write_ISO18K6C			✓					✓			✓	✓	
BLUEBOX_BlockWrite_ISO18K6C													
BLUEBOX_Lock_ISO18K6C			✓					✓			✓	✓	
BLUEBOX_Kill_ISO18K6C			✓					✓			✓	✓	
BLUEBOX_FwUpgrade	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLUEBOX_ReadNumberOfRegistrations													
BLUEBOX_ReadOlderRegistration													
BLUEBOX_CancelOlderRegistration													
BLUEBOX_CancelAllRegistrations													
BLUEBOX_ReadPreviousRegistration													
BLUEBOX_GenericCommand	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

## 5 BlueBox CX Functions Table

	BLUEBOX CX UHF MID RANGE SINGLE CHANNEL	BLUEBOX CX UHF LONG RANGE SINGLE CHANNEL	BLUEBOX CX UHF LONG RANGE DUAL CHANNEL
BLUEBOX_GetSwRelease	✓	✓	✓
BLUEBOX_Init	✓	✓	✓
BLUEBOX_End	✓	✓	✓
BLUEBOX_Open	✓	✓	✓
BLUEBOX_Close	✓	✓	✓
BLUEBOX_SetAddress	✓	✓	✓
BLUEBOX_SetDevice	✓	✓	✓
BLUEBOX_GetDevice	✓	✓	✓
BLUEBOX_SetChannel	✓	✓	✓
BLUEBOX_GetFwRelease	✓	✓	✓
BLUEBOX_Reset	✓	✓	✓
BLUEBOX_GetDateTime	✓	✓	✓
BLUEBOX_SetDateTime	✓	✓	✓
BLUEBOX_ReadParameters	✓	✓	✓
BLUEBOX_WriteParameters	✓	✓	✓
BLUEBOX_DefaultParameters	✓	✓	✓
BLUEBOX_ReadSerialNumber	✓	✓	✓
BLUEBOX_ReadMACAddress	✓	✓	✓
BLUEBOX_ReadConfiguration	✓	✓	✓
BLUEBOX_WriteConfiguration	✓	✓	✓
BLUEBOX_DefaultConfiguration	✓	✓	✓
BLUEBOX_DataRequest	✓	✓	✓
BLUEBOX_QueueRequest	✓	✓	✓
BLUEBOX_FreeTagsMemory	✓	✓	✓
BLUEBOX_AllocateNotifyChannel	✓	✓	✓
BLUEBOX_DeallocateNotifyChannel	✓	✓	✓
BLUEBOX_GetNotification	✓	✓	✓
BLUEBOX_FreeNotifyMemory	✓	✓	✓
BLUEBOX_SetOutput	✓	✓	✓
BLUEBOX_GetReaderStatus	✓	✓	✓
BLUEBOX_GetTemperature	✓	✓	✓
BLUEBOX_RfOnOff	✓	✓	✓
BLUEBOX_SelectiveRfOnOff	✓	✓	✓
BLUEBOX_ReadID_EM4305			
BLUEBOX_Write_EM4305			
BLUEBOX_ReadID_T5557			
BLUEBOX_Write_T5557			
BLUEBOX_ReadID_Q5			
BLUEBOX_WriteQ5			
BLUEBOX_ReadID_HITAG1			
BLUEBOX_ReadPage_HITAG1			
BLUEBOX_WritePage_HITAG1			
BLUEBOX_ReadID_HITAGS			
BLUEBOX_Write_HITAGS			
BLUEBOX_ReadPage_HITAGS			
BLUEBOX_WritePage_HITAGS			
BLUEBOX_ReadID_TITAN			
BLUEBOX_Reset_TITAN			

BLUEBOX_Login_TITAN			
BLUEBOX_WritePassword_TITAN			
BLUEBOX_SelectiveRead_TITAN			
BLUEBOX_SelectiveWrite_TITAN			
BLUEBOX_Inventory_ISO15693			
BLUEBOX_ReadPage_ISO15693			
BLUEBOX_ReadMultiPage_ISO15693			
BLUEBOX_WritePage_ISO15693			
BLUEBOX_WriteMultiPage_ISO15693			
BLUEBOX_LockPage_ISO15693			
BLUEBOX_Write_AFI_ISO15693			
BLUEBOX_Lock_AFI_ISO15693			
BLUEBOX_GetRandomNumber_ICODE_SLI_S			
BLUEBOX_SetPassword_ICODE_SLI_S			
BLUEBOX_WritePassword_ICODE_SLI_S			
BLUEBOX_LockPassword_ICODE_SLI_S			
BLUEBOX_64BitPasswordProtection_ICODE_SLI_S			
BLUEBOX_ProtectPage_ICODE_SLI_S			
BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S			
BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S			
BLUEBOX_Destroy_SLI_S_ICODE_SLI_S			
BLUEBOX_EnablePrivacy_ICODE_SLI_S			
BLUEBOX_Inventory_ISO14443A			
BLUEBOX_ReadBlock_MIFARE_1k			
BLUEBOX_WriteBlock_MIFARE_1k			
BLUEBOX_ReadBlock_MIFARE_4k			
BLUEBOX_WriteBlock_MIFARE_4k			
BLUEBOX_ReadBlock_MIFARE_Ultralight			
BLUEBOX_WriteBlock_MIFARE_Ultralight			
BLUEBOX_Inventory_ISO14443B			
BLUEBOX_ReadBlock_SR176			
BLUEBOX_WriteBlock_SR176			
BLUEBOX_ReadRfParameters	✓	✓	✓
BLUEBOX_WriteRfParameters	✓	✓	✓
BLUEBOX_Inventory_ISO18K6B			
BLUEBOX_Read_ISO18K6B			
BLUEBOX_Write_ISO18K6B			
BLUEBOX_Inventory_ISO18K6C	✓	✓	✓
BLUEBOX_ProgramEPC_ISO18K6C	✓	✓	✓
BLUEBOX_Read_ISO18K6C	✓	✓	✓
BLUEBOX_Write_ISO18K6C	✓	✓	✓
BLUEBOX_BlockWrite_ISO18K6C	✓	✓	✓
BLUEBOX_Lock_ISO18K6C	✓	✓	✓
BLUEBOX_Kill_ISO18K6C	✓	✓	✓
BLUEBOX_FwUpgrade	✓	✓	✓
BLUEBOX_ReadNumberOfRegistrations			
BLUEBOX_ReadOlderRegistration			
BLUEBOX_CancelOlderRegistration			
BLUEBOX_CancelAllRegistrations			
BLUEBOX_ReadPreviousRegistration			
BLUEBOX_GenericCommand	✓	✓	✓

## 6 Document Revision History

Revision	Date	Description
1.00	30/04/10	First release.
1.01	05/05/10	<p>Changes in supported readers list to add the new readers managed from the library release 2.0.0.</p> <p>Added the document revision history section.</p> <p>Added definitions, functions, enums and structs to manage the new readers.</p> <p>Changes in BLUEBOX_SetChannel function parameters.</p> <p>Changes in nibble coding and gain enums definitions (ref. LF readers).</p> <p>Changes in BLUEBOX_GeneralParameters struct definition.</p> <p>Changes in BLUEBOX_TagType enum definition.</p> <p>Changes in MIFARE key enum definitions (ref. HF readers).</p> <p>Changes in BLUEBOX_ReaderStatus struct definition (deleted the Line flag in BLUEBOX INDUSTRIAL readers definition).</p>
1.02	18/06/10	<p>Changes in supported readers list to add the new readers managed from the library release 3.0.0.</p> <p>Added the 'spontaneous' message notifications by adding the functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_AllocateNotifyChannel</li> <li>• BLUEBOX_DeallocateNotifyChannel</li> <li>• BLUEBOX_GetNotification</li> </ul> <p>and the error codes:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_AllocationError</li> </ul> <p>and the structs:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Notify</li> </ul> <p>Added functions to manage the device type:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_SetDevice</li> <li>• BLUEBOX_GetDevice</li> </ul> <p>Added the ISO 15693 AFI (Application Family Identifier) management in BLUEBOX_Inventory_ISO15693 function.</p> <p>Added functions to manage ISO 15693 AFI (Application</p>

Revision	Date	Description
		<p>Family Identifier):</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Write_AFI_ISO15693</li> <li>• BLUEBOX_Lock_AFI_ISO15693</li> </ul> <p>Improved BLUEBOX_Close and BLUEBOX_End functions. The BLUEBOX_End function implicitly calls the BLUEBOX_Close function.</p> <p>Changes in BLUEBOX_SetChannel function parameters strings, added the retransmission numbers at the end of Settings string with RS232/RS485 interface.</p> <p>Added the BLUEBOX_Tag struct definition.</p> <p>Changes in BLUEBOX_RfParameters struct definition.</p> <p>Added the upgrade firmware by adding the functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_FwUpgrade</li> <li>• BLUEBOX_GetUpgradeStatus</li> </ul> <p>and the error codes:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_FileError</li> </ul> <p>and the enums:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_UpgReader</li> </ul> <p>Changes in definitions to uniform the code writing rules.</p> <p>Added the library release information in document revision history table.</p>
1.03	08/07/10	<p>Corrections in the document revision history in description of the 1.02 document release.</p> <p>Changes in supported readers list to add the new readers managed from the library release 4.0.0.</p> <p>Changes in BLUEBOX_SetChannel function parameters strings, added the communication timeout of Settings string with all interfaces.</p> <p>Changes in BLUEBOX_GetFwRelease to allow the auxiliary reader's fw version reading.</p> <p>Changes in configuration functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadParameters;</li> <li>• BLUEBOX_WriteParameters;</li> <li>• BLUEBOX_ReadRfParameters;</li> <li>• BLUEBOX_WriteRfParameters;</li> </ul> <p>parameters to make them more flexible. Also deleted all the enumerations and structures related to them.</p> <p>Changes in BLUEBOX_GetReaderStatus function parameters to make it more flexible. Also deleted all the enumerations and structures related to it.</p>

Revision	Date	Description
		<p>Changed the BLUEBOX_UpgReader enumeration to BLUEBOX_Reader and also changed its items names.</p> <p>Added the BLUEBOX_TagError error code in the following functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Inventory_ISO15693;</li> <li>• BLUEBOX_Inventory_ISO14443A;</li> <li>• BLUEBOX_Inventory_ISO14443B.</li> </ul>
1.04	02/08/10	<p>Changed the BLUEBOX_GetDevice and BLUEBOX_SetDevice function parameters by adding the firmware version major and minor numbers to manage different features in different firmware versions.</p>
1.05	05/08/10	<p>Added the EM4305 and T5557 tags management in BLUEBOX INDUSTRIAL LF SHORT RANGE SINGLE/DUAL CHANNEL, BLUEBOX OEM LF and BLUEBOX DESKTOP LF by adding, in BLUEBOX_TagType the enumerators:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_EM4305;</li> <li>• BLUEBOX_T5557;</li> </ul> <p>and definitions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_EM4305_ID_SIZE;</li> <li>• BLUEBOX_T5557_ID_SIZE;</li> </ul> <p>and functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadID_EM4305;</li> <li>• BLUEBOX_Write_EM4305;</li> <li>• BLUEBOX_ReadID_T5557;</li> <li>• BLUEBOX_Write_T5557.</li> </ul>
1.06	05/10/10	<p>Corrections in the document revision history in description of the 1.04 document release.</p> <p>Deleted the Antenna parameter from BLUEBOX_RfOnOff function.</p> <p>Deleted the BLUEBOX_GetUpgradeStatus function.</p> <p>Added the BLUEBOX PORTAL UHF reader management. The affected functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_SetDevice;</li> <li>• BLUEBOX_GetDevice;</li> <li>• BLUEBOX_GetFwRelease;</li> <li>• BLUEBOX_ReadParameters;</li> <li>• BLUEBOX_WriteParameters;</li> <li>• BLUEBOX_DefaultParameters;</li> <li>• BLUEBOX_GetReaderStatus;</li> </ul>



Revision	Date	Description
		<ul style="list-style-type: none"> <li>• BLUEBOX_ReadRfParameters;</li> <li>• BLUEBOX_WriteRfParameters;</li> </ul> <p>the functions added are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadNumberOfRegistrations;</li> <li>• BLUEBOX_ReadOlderRegistration;</li> <li>• BLUEBOX_CancelOlderRegistration;</li> <li>• BLUEBOX_CancelAllRegistrations;</li> <li>• BLUEBOX_ReadPreviousRegistration;</li> </ul> <p>the affected definitions / enumerations / structures are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ErrorCodes;</li> </ul> <p>the definitions / enumerations / structures added are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Input;</li> <li>• BLUEBOX_Registration.</li> </ul>
1.07	17/01/11	<p>Added the ISO 18000-6C (EPC C1G2) with variable UID size tags management. The affected definitions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ISO18K6C_UID_SIZE.</li> </ul> <p>Added the section remarks in functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_GetFwRelease;</li> <li>• BLUEBOX_ReadParameters.</li> </ul>
1.08	09/09/11	<p>Added the firmware release related to this technical manual in the first page.</p> <p>Added the x64 architecture support in section 1.</p> <p>Deleted the BLUEBOX INDUSTRIAL UHF SHORT RANGE SINGLE CHANNEL reader management (replaced with the MID RANGE one).</p> <p>Added the customization management through the variant management. The affected functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_SetDevice;</li> <li>• BLUEBOX_GetDevice.</li> </ul> <p>Changed the BLUEBOX_AllocateNotifyChannel parameters and function prototype.</p> <p>Changes in section 'Document Revision History' (this section).</p>
1.09	24/10/11	<p>Extended the BLUEBOX_Input enumeration to support the 'no input' case.</p> <p>Increased the maximum tag's ID length supported (BLUEBOX_MAX_ID_LENGTH definition).</p> <p>Changes to BLUEBOX_Tag and BLUEBOX_Notify structures.</p>

Revision	Date	Description
1.10	19/01/12	<p>Deleted the maximum tag's ID length definitions (BLUEBOX_MAX_ID_LENGTH).</p> <p>Changed the tag's ID management from static array to dynamic array. The affected structures are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Tags;</li> <li>• BLUEBOX_Notify;</li> <li>• BLUEBOX_Registration.</li> </ul> <p>The affected functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_FreeTagsMemory;</li> <li>• BLUEBOX_FreeNotifyMemory.</li> </ul>
1.11	22/02/12	<p>Added the ICODE SLI-S tag management in BLUEBOX DESKTOP HF and BLUEBOX OEM HF by adding, in BLUEBOX_TagType the enumerators:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ICODE_SLI_S;</li> </ul> <p>and enumerations:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ICODE_SLI_S_PasswordIdentifier;</li> <li>• BLUEBOX_ICODE_SLI_S_ProtectionStatus;</li> </ul> <p>and definitions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ICODE_SLI_S_RND_SIZE;</li> <li>• BLUEBOX_ICODE_SLI_S_PWD_SIZE;</li> </ul> <p>and structures:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ICODE_SLI_S_BlockProtectionStatus;</li> </ul> <p>and functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_GetRandomNumber_ICODE_SLI_S;</li> <li>• BLUEBOX_SetPassword_ICODE_SLI_S;</li> <li>• BLUEBOX_WritePassword_ICODE_SLI_S;</li> <li>• BLUEBOX_LockPassword_ICODE_SLI_S;</li> <li>• BLUEBOX_64BitPasswordProtection_ICODE_SLI_S;</li> <li>• BLUEBOX_ProtectPage_ICODE_SLI_S;</li> <li>• BLUEBOX_LockPageProtectionCondition_ICODE_SLI_S;</li> <li>• BLUEBOX_GetMultipleBlockProtectionStatus_ICODE_SLI_S;</li> <li>• BLUEBOX_Destroy_SLI_S_ICODE_SLI_S;</li> <li>• BLUEBOX_EnablePrivacy_ICODE_SLI_S;</li> </ul>
1.12	10/10/12	<p>Added the BLUEBOX Gen2 readers support to the library (BLUEBOX GEN2 DESKTOP LF, BLUEBOX GEN2 DESKTOP HF, BLUEBOX GEN2 DESKTOP UHF, BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE SINGLE CHANNEL,</p>

Revision	Date	Description
		<p>BLUEBOX GEN2 INDUSTRIAL LF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF SHORT RANGE DUAL CHANNEL, BLUEBOX GEN2 INDUSTRIAL HF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 INDUSTRIAL UHF MID RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC LF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC HF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC UHF SHORT RANGE SINGLE CHANNEL, BLUEBOX GEN2 BASIC MID RANGE SINGLE CHANNEL.</p> <p>Deleted the list of the supported readers in section 1 and added sections 4 and 5 with the readers supported functions tables.</p> <p>Added the management of a second auxiliary reader. The affected enumerations are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_Reader.</li> </ul> <p>The affected functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_GetFwRelease;</li> <li>• BLUEBOX_FwUpgrade.</li> </ul> <p>Added the BLUEBOX_Reset function.</p> <p>Added the management of the configuration pages of the readers. The added functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadConfiguration;</li> <li>• BLUEBOX_WriteConfiguration;</li> <li>• BLUEBOX_DefaultConfiguration.</li> </ul> <p>Removed remarks in BLUEBOX_SetDevice and BLUEBOX_GetDevice functions.</p> <p>Added the BLUEBOX_GenericCommand function.</p>
1.13	29/04/13	<p>Added the BLUEBOX_GetTemperature function.</p> <p>Added the management of the HITAG 1 transponders. The affected enumerations are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_TagType (added BLUEBOX_HITAG_1).</li> </ul> <p>The added definitions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_HITAG1_ID_SIZE;</li> <li>• BLUEBOX_HITAG1_PAGE_SIZE.</li> </ul> <p>The added functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadID_HITAG1;</li> <li>• BLUEBOX_ReadPage_HITAG1;</li> <li>• BLUEBOX_WritePage_HITAG1.</li> </ul>

Revision	Date	Description
1.14	20/05/13	<p>Changed the BLUEBOX_ISO18K6C_UID_SIZE to 66 bytes.</p> <p>Added the read/write multi page of ISO 15693 tags. The added functions are:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_ReadMultiPage_ISO15693;</li> <li>• BLUEBOX_WriteMultiPage_ISO15693.</li> </ul> <p>Added the management of BLUEBOX GEN2 OEM UHF readers.</p>
1.15	08/10/13	<p>Added the BLUEBOX OEM UHF reader support to the library.</p> <p>Added the enumeration items</p> <ul style="list-style-type: none"> <li>• BLUEBOX_EM4305;</li> <li>• BLUEBOX_T5557;</li> <li>• BLUEBOX_ICODE_SLI_S;</li> <li>• BLUEBOX_HITAG_1;</li> <li>• BLUEBOX_MIFARE_MINI;</li> <li>• BLUEBOX_MIFARE_DESFIRE;</li> <li>• BLUEBOX_MIFARE_7BUID_2k;</li> <li>• BLUEBOX_MIFARE_7BUID_4k;</li> <li>• BLUEBOX_MIFARE_PLUS_2k;</li> <li>• BLUEBOX_MIFARE_PLUS_4k;</li> <li>• BLUEBOX_SRI512;</li> <li>• BLUEBOX_JCOS;</li> <li>• BLUEBOX_PICOPASS;</li> </ul> <p>to BLUEBOX_TagType enumeration.</p> <p>Added the definitions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_MIFARE_DESFIRE_UID_SIZE;</li> <li>• BLUEBOX_MIFARE_7BUID_2k_UID_SIZE;</li> <li>• BLUEBOX_MIFARE_7BUID_4k_UID_SIZE;</li> <li>• BLUEBOX_MIFARE_PLUS_2k_UID_SIZE;</li> <li>• BLUEBOX_MIFARE_PLUS_4k_UID_SIZE;</li> <li>• BLUEBOX_SRI512_UID_SIZE;</li> <li>• BLUEBOX_JCOS_UID_SIZE;</li> <li>• BLUEBOX_PICOPASS_UID_SIZE.</li> </ul> <p>Added the functions:</p> <ul style="list-style-type: none"> <li>• BLUEBOX_GetDateTime;</li> <li>• BLUEBOX_SetDateTime;</li> <li>• BLUEBOX_ReadSerialNumber;</li> </ul>

Revision	Date	Description
		<ul style="list-style-type: none"> <li>• BLUEBOX_ReadMACAddress;</li> <li>• BLUEBOX_SelectiveRfOnOff.</li> </ul> Updated the supported commands table.
1.16	10/02/15	Added the BLUEBOX CX UHF reader support to the library. Added the functions: <ul style="list-style-type: none"> <li>• BLUEBOX_ProgramEPC_ISO18K6C;</li> <li>• BLUEBOX_BlockWrite_ISO18K6C.</li> </ul> Updated the supported commands tables.
1.17	29/09/16	Added the management of the NTAG213/215216/ transponders. The added definitions are: <ul style="list-style-type: none"> <li>• BLUEBOX_NTAG21x_UID_SIZE;</li> <li>• BLUEBOX_NTAG21x_BLOCK_SIZE.</li> </ul> The added functions are: <ul style="list-style-type: none"> <li>• BLUEBOX_ReadBlock_NTAG213;</li> <li>• BLUEBOX_WriteBlock_NTAG213;</li> <li>• BLUEBOX_ReadBlock_NTAG215;</li> <li>• BLUEBOX_WriteBlock_NTAG215;</li> <li>• BLUEBOX_ReadBlock_NTAG216;</li> <li>• BLUEBOX_WriteBlock_NTAG216.</li> </ul>