

### Developing an application allowing PR101-USB and MR101-USB FEIG readers to communicate with M24LRXX transponders

#### Introduction

This application note explains how to develop a Visual Basic or C/C++ application code to drive ISO 15693 FEIG readers from a host computer. FEIG readers are contactless readers which can communicate with transponders based on the STMicroelectronics M24LRXX Dual interface EEPROM.

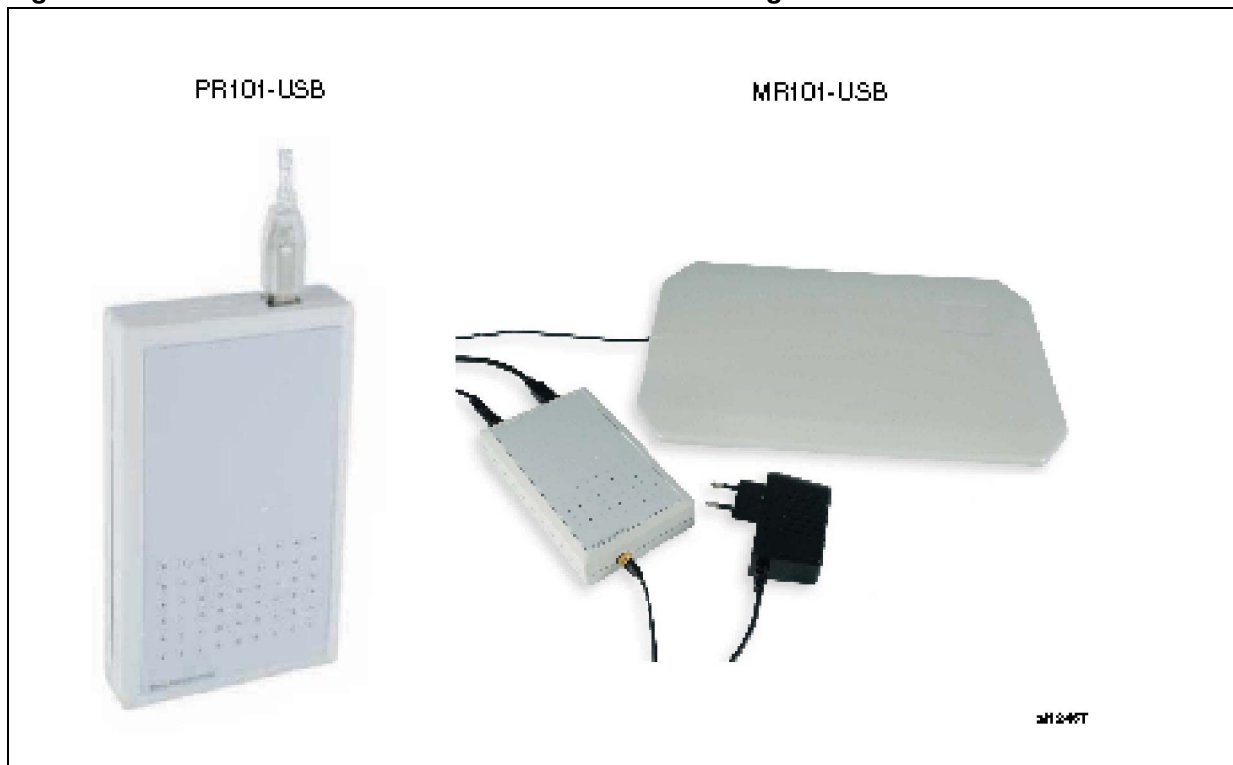
The readers are delivered within ST demonstration kit DEMOKIT-M24LR-A and development kit DEVKIT-M24LR-A:

- PR101-USB FEIG reader is delivered within the DEMOKIT-M24LR-A (see [Figure 1](#))
- MR101-USB FEIG reader is delivered within the DEVKIT-M24LR-A (see [Figure 1](#)).

This application note helps software engineers using and including the software delivered within the DEMOKIT-M24LR-A and the DEVKIT-M24LR-A in their own application. Code examples are also provided to illustrate how to send Visual Basic and C/C++ RF commands.

- Note:*
- 1 For details on FEIG readers go to <http://www.feig.de>.
  - 2 This application note complements FEIG development tool documentation and examples.

**Figure 1. PR101-USB and MR101-USB RF reader with integrated RF antenna**



# Contents

- 1 Description . . . . . 7**
  - 1.1 M24LRXX dual interface EEPROM . . . . . 7
  - 1.2 PR101-USB and MR101-USB FEIG contactless readers . . . . . 7
    - 1.2.1 PR101-USB FEIG reader . . . . . 7
    - 1.2.2 MR101-USB FEIG reader . . . . . 8
  - 1.3 FEIG development toolchain . . . . . 8
- 2 RF ISO 15693 commands . . . . . 9**
- 3 Installation requirements . . . . . 10**
  - 3.1 Installing the .dll files . . . . . 10
- 4 PR101-USB and MR101-USB reader detection . . . . . 12**
  - 4.1 Overview of reader detection and connection functions . . . . . 12
    - 4.1.1 FEUSB\_ClearScanList . . . . . 13
    - 4.1.2 FEUSB\_Scan . . . . . 14
    - 4.1.3 FEUSB\_GetScanListSize . . . . . 16
    - 4.1.4 FEUSB\_GetScanListPara . . . . . 17
    - 4.1.5 FEUSB\_OpenDevice . . . . . 18
    - 4.1.6 FEISC\_NewReader . . . . . 19
  - 4.2 Detection and connection sequence . . . . . 20
  - 4.3 Visual Basic source code example . . . . . 20
  - 4.4 C/C++ source code example . . . . . 23
- 5 RF ISO 15693 High-level commands . . . . . 25**
  - 5.1 FEISC\_0xB0\_ISOCmd general description . . . . . 26
  - 5.2 RF ISO 15693 High-level Inventory command . . . . . 28
    - 5.2.1 Example of High-level Inventory command  
Visual Basic source code . . . . . 28
    - 5.2.2 Example of High-level Inventory command  
C/C++ source code . . . . . 29
  - 5.3 RF ISO 15693 High-level RESET TO READY command . . . . . 31
    - 5.3.1 RESET TO READY command  
Visual Basic source code example . . . . . 31

<b>6</b>	<b>RF ISO 15693 Transparent commands</b> .....	<b>32</b>
6.1	FEISC_0xBF_ISOTranspCmd general description .....	33
6.2	Issuing a Read single block request with a transparent command .....	35
6.2.1	Transparent Read single block command Visual Basic source code example .....	36
6.2.2	Transparent Read single block command C/C++ source code example .....	37
6.3	Issuing a Write single block request with a transparent command .....	39
6.3.1	Transparent Write single block command Visual Basic source code example .....	40
6.3.2	Transparent Write single block command C/C++ source code example .....	41
<b>Appendix A FEIG reference documents</b> .....		<b>43</b>
<b>Appendix B Useful source code zip files</b> .....		<b>44</b>
<b>Appendix C List of error codes</b> .....		<b>45</b>
C.1	FEUSB error codes .....	45
C.1.1	Common errors .....	45
C.1.2	Scanning errors .....	45
C.1.3	Handle errors .....	45
C.1.4	Communication errors .....	45
C.1.5	Open/close device errors .....	45
C.1.6	Parameter errors .....	46
C.1.7	Identification errors .....	46
C.2	FEISC error codes .....	46
C.2.1	Common errors .....	46
C.2.2	Query errors .....	46
C.2.3	Handle errors .....	46
C.2.4	Communication errors .....	47
C.2.5	Parameter errors .....	47
C.2.6	Plug-in errors .....	47
C.2.7	Communication data flow errors .....	47
C.2.8	Task errors .....	47

---

**Revision history** ..... **48**

## List of tables

Table 1.	FEUSB_ClearScanList Visual Basic function . . . . .	13
Table 2.	FEUSB_ClearScanList C/C++ function . . . . .	13
Table 3.	FEUSB_Scan Visual Basic function . . . . .	14
Table 4.	FEUSB_Scan C/C++ function. . . . .	15
Table 5.	FEUSB_GetScanListSize Visual Basic function . . . . .	16
Table 6.	FEUSB_GetScanListSize C/C++ function . . . . .	16
Table 7.	FEUSB_GetScanListPara Visual Basic function . . . . .	17
Table 8.	FEUSB_GetScanListPara C/C++ function . . . . .	17
Table 9.	FEUSB_OpenDevice Visual Basic function . . . . .	18
Table 10.	FEUSB_OpenDevice C/C++ function . . . . .	18
Table 11.	FEISC_NewReader Visual Basic function . . . . .	19
Table 12.	FEISC_NewReader C/C++ function . . . . .	19
Table 13.	FEISC_0xB0_ISOCmd Visual Basic prototype. . . . .	26
Table 14.	FEISC_0xB0_ISOCmd C/C++ prototype . . . . .	27
Table 15.	FEISC_0xBF_ISOTranspCmd Visual Basic prototype . . . . .	33
Table 16.	FEISC_0xBF_ISOTranspCmd C/C++ prototype . . . . .	34
Table 17.	Example of Read single block command in Visual Basic . . . . .	35
Table 18.	Example of Read single block command in C/C++ . . . . .	36
Table 19.	Example of Write single block command in Visual Basic . . . . .	39
Table 20.	Example of Write single block command in C/C++ . . . . .	40
Table 21.	Document revision history . . . . .	48

---

## List of figures

Figure 1.	PR101-USB and MR101-USB RF reader with integrated RF antenna . . . . .	1
Figure 2.	DEMOKIT-M24LR-A application schematics . . . . .	7
Figure 3.	DEVKIT-M24LR-A application schematics . . . . .	8

# 1 Description

## 1.1 M24LRXX dual interface EEPROM

The M24LRXX is a dual interface EEPROM which can be accessed either through an I<sup>2</sup>C serial bus or a contactless interface using the ISO 15693 RFID protocol.

To easily access the M24LRXX content through its RF interface, ST offers several evaluation kits, among which are the DEMOKIT-M24LR-A and the DEVKIT-M24LR-A.

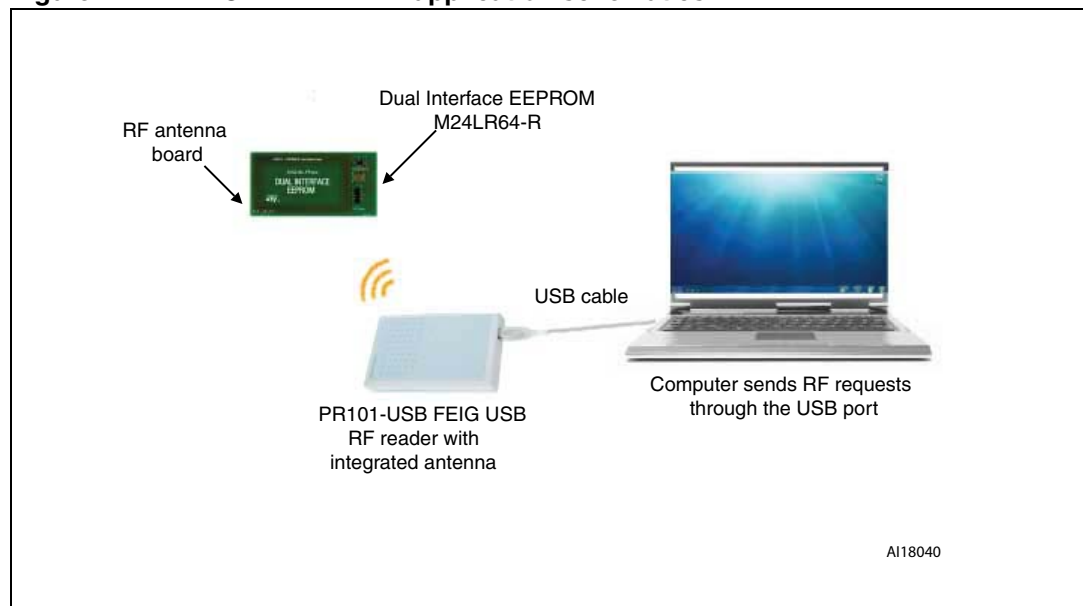
Refer to the product datasheet and to application note AN3163 “Configuring your ISO 15693 reader to support the M24LRXX” for more in-depth information on the M24LRXX and for explanations on the RF and I<sup>2</sup>C communication protocols. Both documents are available from <http://www.st.com>.

## 1.2 PR101-USB and MR101-USB FEIG contactless readers

### 1.2.1 PR101-USB FEIG reader

The PR101-USB RF reader is delivered within ST DEMOKIT-M24LR-A. It supports the ISO 15693 protocol with high datarate transfers and one subcarrier to communicate with M24LRXX-based transponders by sending RF ISO 15693 commands. The reader is connected to the host-computer USB port (see [Figure 2](#)).

**Figure 2. DEMOKIT-M24LR-A application schematics**



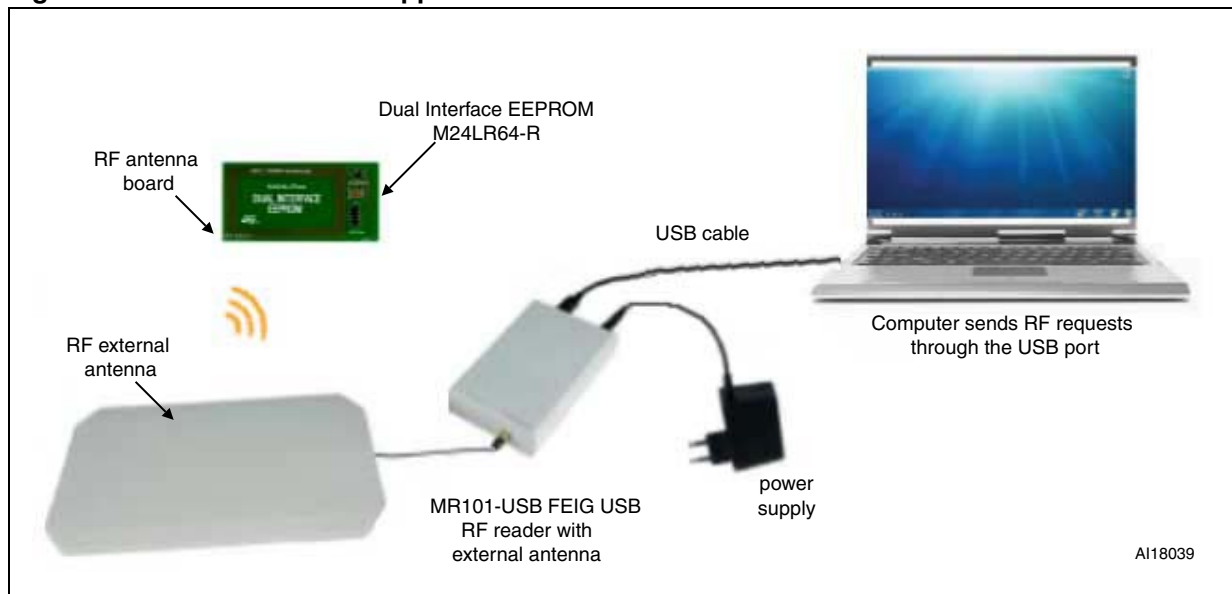
### 1.2.2 MR101-USB FEIG reader

The MR101-USB is connected to the host-computer USB port. It features an external antenna and is powered from an external power supply to increase the RF read range (see [Figure 3](#)).

The reader supports the ISO 15693 protocol with high datarate transfers and one subcarrier.

The MR101-USB RF reader is delivered within the DEVKIT-M24LR-A. It communicates with M24LRXX-based transponders using RF ISO 15693 commands.

**Figure 3. DEVKIT-M24LR-A application schematics**



### 1.3 FEIG development toolchain

The PR101-USB and MR101-USB FEIG readers are driven by the same software. USB driver libraries, software development kits (free Windows SDK and other platform SDKs with charge) and OBID drivers can be downloaded from <http://www.feig.de>. To download FEIG software, go to <http://www.feig.de> and click **Download**. You can use STMicroelectronics customers' account:

User: stm\_customer

Password: 01032010france

Refer to for [Appendix A](#) the list of FEIG reference document.



## 2 RF ISO 15693 commands

RF ISO 15693 commands are sent by the host computer to the M24LRXX transponders via the PR101-USB or MR101-USB reader RF interface. Two types of commands are available:

- High-level commands (see [Section 5](#))

The host sends an already formatted request sent to the transponders via the reader RF interface. As an example, the High-level Inventory request launches an anticollision sequence to identify all the transponders present in the RF field and sends back the UID information to the host. Refer to <http://www.feig.de> for a description of the full set of high-level commands supported by the readers.

- Transparent commands (see [Section 6](#)).

RF Transparent commands are sent by the host to the transponders via the reader RF interface. The Transparent commands transmit single or multiple frames compliant with the ISO 15693 protocol. Refer to the MR101-USB datasheet for a detailed description of the available Dual interface EEPROM command.

The transponders answers are sent back to the computer.

Examples are included to allow you using, modifying the RF requests sent to the transponders, and decoding RF answers from the transponders. Both Visual Basic and C/C++ examples are provided.

## 3 Installation requirements

FEIG Windows SDK includes several .dll files which allow to drive all the FEIG USB readers in Visual Basic and C/C++ languages. They support all RF ISO 15693 commands.

The .dll files required to drive FEIG drivers are the following:

- Low level functions to drive USB readers
  - *FEUSB.dll* contains all the functions required for the PC to communicate through a USB interface.
  - *FEUSB.bas*: declaration file for Visual Basic project (VB6)
  - *FEUSB.h*: header file for C/C++ project
  - *FEUSB.lib*: declaration file for C/C++ project
- Medium level functions to driver any readers
  - *FEISC.dll* contains the functions required to perform basic communications through any communication interfaces
  - *FEISC.bas*: declaration file for Visual Basic project (VB6)
  - *FEISC.h*: header file for C/C++ project
  - *FEISC.lib*: declaration file for C/C++ project

Refer to [Section 4.1: Overview of reader detection and connection functions](#) for a detailed description of FEUSB.dll and FEISC.dll function.

Dll documentation and programming examples are available on the FEIG website <http://www.feig.de>.

### 3.1 Installing the .dll files

Follow the steps below to install the .dll files on your computer:

1. Copy the .dll files in C:/Windows/System32/
  - If you have installed the *M24LRxx\_Application\_Software*, then the *dll* files are already present in your Windows system folder.
2. Add the library files and header files to your software project:
  - Before using the functions included in the .dll files, reference the .dll files in your software project. .h, .lib and .bas files are available for Visual Basic and C/C++ software development.
  - a) Visual Basic project requirements:
    - Insert FEUSB.bas and FEISC.bas in your Visual Basic project. These files contain all the High-level or Transparent commands with their declaration and their description. Below is are examples of the command that needs to be declared in your Visual Basic header file.

#### **USB management command for FEIG reader detection**

```
Public Declare Function FEUSB_OpenDevice Lib  
"FEUSB.DLL" (ByVal dwDeviceID As Long) As Long
```

**High-level command declaration**

```
Public Declare Function FEISC_0xB0_ISOCmd Lib "FEISC.DLL"
  (ByVal iReaderHnd As Long, ByVal cBusAdr As Byte, ByVal
  cReqData As String, ByVal iReqLen As Long, ByVal cRspData As
  String, iRspLen As Long, ByVal iDataType As Long) As Long
```

**Transparent mode command declaration**

```
Public Declare Function FEISC_0xBF_ISOTranspCmd Lib
  "FEISC.DLL" (ByVal iReaderHnd As Long, ByVal cBusAdr As Byte,
  ByVal iMode As Long, ByVal iRspLength As Long, ByVal cReqData
  As String, ByVal iReqLen As Long, ByVal cRspData As String,
  iRspLen As Long, ByVal iDataType As Long) As Long ByVal
  dwDeviceID As Long) As Long
```

## b) C/C++ project requirements

When working in C/C++, insert FEUSB.h, FEUSB.lib, FEISC.h, FEISC.lib in your C/C++ project, and declare them in your source code header file:

```
#include "FEUSB.h"
#include "FEISC.h"
```

Below are examples of command declaration in C/C++.

**USB management command for FEIG reader detection**

```
int DLL_EXT_FUNC FEUSB_OpenDevice( long nDeviceID );
```

**High-level command declaration**

```
int DLL_EXT_FUNC FEISC_0xB0_ISOCmd(
  int iReaderHnd,
  unsigned char cBusAdr,
  unsigned char* cReqData,
  int iReqLen,
  unsigned char* cRspData,
  int* iRspLen,
  int iDataFormat );
```

**Transparent mode command declaration**

```
int DLL_EXT_FUNC FEISC_0xBF_ISOTranspCmd(
  int iReaderHnd,
  unsigned char cBusAdr,
  int iMode,
  int iRspLength,
  unsigned char* cReqData,
  int iReqLen,
  unsigned char* cRspData,
  int* iRspLen,
  int iDataFormat );
```

## 4 PR101-USB and MR101-USB reader detection

Prior to sending any RF command to a PR101-USB or an MR101-USB RF readers, the host computer must detect the reader. Once detected, a handle is randomly assigned to the reader.

This section presents all the available functions for performing reader detection. Visual Basic and C/C++ source code examples are also provided.

The readers are driven by the FEUSB and FEISC functions delivered in the *FEUSB.dll* and *FEISC.dll* files (see [Section 3: Installation requirements](#)).

Refer to Appendix [C.1](#) for the full list of error codes returned by the FEUSB functions.

### 4.1 Overview of reader detection and connection functions

*FEUSB.dll* and *FEISC.dll* include several functions allowing to detect or connect to a reader:

- **FEUSB\_ClearScanList**  
This function re-initializes the USB detection process by clearing the list of scanned readers. Refer to [Section 4.1.1](#) for a detailed description of the function.
- **FEUSB\_Scan**  
This function searches for all the USB FEIG readers connected to the USB ports of the host computer. Refer to [Section 4.1.2](#) for a detailed description of the function.
- **FEUSB\_GetScanListSize**  
This function retrieves the number of detected readers. Refer to [Section 4.1.3](#) for a detailed description of the function.
- **FEUSB\_GetScanListPara**  
This function gives access to all the detected reader information. Refer to [Section 4.1.4](#) for a detailed description of the function.
- **FEUSB\_OpenDevice**  
This function opens a communication channel between a USB FEIG reader and the **FEUSB dll**, and assigns a handle to this channel. Refer to [Section 4.1.5](#) for a detailed description of the function.
- **FEISC\_NewReader**  
This function opens a communication channel between a USB FEIG reader and the **FEISC dll**. The reader must have been previously detected by the `FEUSB_OpenDevice` function. Refer to [Section 4.1.6](#) for a detailed description of the function.

### 4.1.1 FEUSB\_ClearScanList

#### FEUSB\_ClearScanList Visual Basic prototype

[Table 1](#) illustrates the FEUSB\_ClearScanList Visual Basic function.

**Table 1. FEUSB\_ClearScanList Visual Basic function**

Function description	
Declaration	Public Declare Sub FEUSB_ClearScanList Lib "FEUSB.DLL" ()
Prototype	Call FEUSB_ClearScanList
Parameters	None
Returned value	None

#### FEUSB\_ClearScanList C/C++ prototype

[Table 2](#) illustrates the FEUSB\_ClearScanList C/C++ function.

**Table 2. FEUSB\_ClearScanList C/C++ function**

Function description	
Declaration	void DLL_EXT_FUNC FEUSB_ClearScanList();
Prototype	FEUSB_ClearScanList();
Parameters	None
Returned value	None

### 4.1.2 FEUSB\_Scan

#### FEUSB\_Scan Visual Basic prototype

Table 3 illustrates the FEUSB\_Scan Visual Basic function.

**Table 3. FEUSB\_Scan Visual Basic function**

Function description	
Declaration	Public Declare Function FEUSB_Scan Lib "FEUSB.DLL" (ByVal iScanOpt As Long, oSearchOpt As FEUSB_SCANSEARCH) As Long
Prototype	Dim SearchOpt As FEUSB_SCANSEARCH lngScanValue = FEUSB_Scan(FEUSB_SCAN_ALL, SearchOpt)
Parameters	<p><b>iScanOpt:</b>                      FEUSB_SCAN_FIRST: performs a search for the first device registered by the PC                      FEUSB_SCAN_NEXT: performs a search for the next device registered by the PC                      FEUSB_SCAN_NEW : performs a search for a new connected device                      FEUSB_SCAN_ALL: performs a search for all devices connected to the USB port.                      FEUSB_SCAN_SEARCH: this parameter can be added to FEUSB_SCAN_ALL to perform a search for a specific device.</p> <p><b>oSearchOpt:</b> This parameter contains the parameters of all the USB devices detected on the PC USB port.</p>
Returned value	0: for FEUSB_SCAN_FIRST or FEUSB_SCAN_ALL if at least one USB device have been detected Index of the detected device if any: for FEUSB_SCAN_NEXT and FEUSB_SCAN_NEW Otherwise FEUSB error code
Example	lngScanValue = FEUSB_Scan (FEUSB_SCAN_ALL, FEUSB_SCAN_SEARCH, oSearchOpt) with oSearchOpt containing the identifier of a specific device to be searched for.

## FEUSB\_Scan C/C++ prototype

Table 4 illustrates the FEUSB\_Scan C/C++ function.

**Table 4. FEUSB\_Scan C/C++ function**

Function description	
Declaration	Public Declare Function FEUSB_Scan Lib "FEUSB.DLL" (ByVal iScanOpt As Long, SearchOpt As FEUSB_SCANSEARCH) As Long
Prototype	FEUSB_SCANSEARCH SearchOpt; iScanValue = FEUSB_Scan(FEUSB_SCAN_ ALL, &SearchOpt);
Parameters	<p><b>iScanOpt:</b>  FEUSB_SCAN_FIRST: performs a search for the first device registered by the PC  FEUSB_SCAN_NEXT: performs a search for the next device registered by the PC  FEUSB_SCAN_NEW: performs a search for a new connected device  FEUSB_SCAN_ALL: performs a search for all devices connected to the USB port.  FEUSB_SCAN_SEARCH: this parameter can be added to FEUSB_SCAN_ALL to perform a search for a specific device.</p> <p><b>SearchOpt:</b> This parameter contains the parameters of all the USB devices detected on the PC USB port.</p>
Returned value	0: for FEUSB_SCAN_FIRST or FEUSB_SCAN_ALL if at least one USB device have been detected Index of the detected device if any: for FEUSB_SCAN_NEXT and FEUSB_SCAN_NEW Otherwise FEUSB error code

### 4.1.3 FEUSB\_GetScanListSize

#### FEUSB\_GetScanListSize Visual Basic prototype

Table 5 illustrates the FEUSB\_GetScanListSize Visual Basic function.

**Table 5. FEUSB\_GetScanListSize Visual Basic function**

Function description	
Declaration	Public Declare Function FEUSB_GetScanListSize Lib "FEUSB.DLL" () As Long
Prototype	lngReaderNumber = FEUSB_GetScanListSize
Parameters	None
Returned value	Number of readers connected to the computer USB port and detected by the FEUSB_Scan function Otherwise FEUSB error code.

#### FEUSB\_GetScanListSize C/C++ prototype

Table 6 illustrates the FEUSB\_GetScanListSize C/C++ function.

**Table 6. FEUSB\_GetScanListSize C/C++ function**

Function description	
Declaration	int DLL_EXT_FUNC FEUSB_GetScanListSize();
Prototype	intReaderNumber = FEUSB_GetScanListSize();
Parameters	None
Returned value	Number of readers connected to the computer USB port and detected by the FEUSB_Scan function Otherwise FEUSB error code.



#### 4.1.4 FEUSB\_GetScanListPara

##### FEUSB\_GetScanListPara Visual Basic prototype

[Table 7](#) illustrates the FEUSB\_GetScanListPara Visual Basic function.

**Table 7. FEUSB\_GetScanListPara Visual Basic function**

Function description	
Declaration	Public Declare Function FEUSB_GetScanListPara Lib "FEUSB.DLL" (ByVal iIndex As Long, ByVal cPara As String, ByVal cValue As String) As Long
Prototype	error = FEUSB_GetScanListPara(iIndex, cPara, cValue)
Parameters	<p><b>iIndex:</b> Device index number when several USB readers are connected to the computer USB port.</p> <p><b>cPara:</b> Character string containing</p> <ul style="list-style-type: none"> <li>Device-ID: USB device serial number</li> <li>DeviceHnd: USB channel device handle</li> <li>FamilyName: Name of the device family corresponding to the device connected to the USB channel</li> <li>DeviceName: Name of the device connected to the USB channel</li> <li>Present: USB device if connected (cValue=1) or disconnected (cValue=0)</li> </ul> <p><b>cValue:</b> Value returned by the reader depending on cPara.</p>
Returned value	0: no error Otherwise FEUSB error code

##### FEUSB\_GetScanListPara C/C++ prototype

[Table 8](#) illustrates the FEUSB\_GetScanListPara C/C++ function.

**Table 8. FEUSB\_GetScanListPara C/C++ function**

Function description	
Declaration	int DLL_EXT_FUNC FEUSB_GetScanListPara( int iIndex, char* cParaID, char* cValue );
Prototype	error = FEUSB_GetScanListPara(iIndex, cPara, cValue);
Parameters	<p><b>iIndex:</b> Device index number when several USB readers are connected to the computer USB port.</p> <p><b>cPara:</b> character string containing</p> <ul style="list-style-type: none"> <li>Device-ID: USB device serial number</li> <li>DeviceHnd: USB channel device handle</li> <li>FamilyName: Name of the device family corresponding to the device connected to the USB channel</li> <li>DeviceName: Name of the device connected to the USB channel</li> <li>Present: USB device if connected (cValue=1) or disconnected (cValue=0)</li> </ul> <p><b>cValue:</b> Value returned by the reader depending on cPara.</p>
Returned value	0: no error Otherwise FEUSB error code

### 4.1.5 FEUSB\_OpenDevice

#### FEUSB\_OpenDevice Visual Basic prototype

Table 9 illustrates the FEUSB\_OpenDevice Visual Basic function.

**Table 9. FEUSB\_OpenDevice Visual Basic function**

Function description	
Declaration	Public Declare Function FEUSB_OpenDevice Lib "FEUSB.DLL" (ByVal dwDeviceID As Long) As Long
Prototype	iHandle = FEUSB_OpenDevice(lngDeviceId(i))
Parameters	<b>dwDeviceID:</b> The reader is detected by calling FEUSB_Scan. The FEUSB_GetScanListPara allows retrieving the value of the strDeviceHnd parameter that is converted in Long to obtain dwDeviceID.
Returned value	<b>iHandle:</b> USB reader handle for FEUSB functions Otherwise FEUSB Error code

#### FEUSB\_OpenDevice C/C++ prototype

Table 10 illustrates the FEUSB\_OpenDevice C/C++ function.

**Table 10. FEUSB\_OpenDevice C/C++ function**

Function description	
Declaration	int DLL_EXT_FUNC FEUSB_OpenDevice( long nDeviceID );
Prototype	iDeviceHandle = FEUSB_OpenDevice(dwDeviceSerialNumber);
Parameters	<b>dwDeviceID:</b> The reader is detected by calling FEUSB_Scan. The FEUSB_GetScanListPara allows retrieving the value of the strDeviceHnd parameter that is converted in Long to obtain dwDeviceID.
Returned value	<b>iHandle:</b> USB reader handle for FEUSB functions Otherwise FEUSB Error code

#### 4.1.6 FEISC\_NewReader

##### FEISC\_NewReader Visual Basic prototype

[Table 11](#) illustrates the FEISC\_NewReader Visual Basic function.

**Table 11. FEISC\_NewReader Visual Basic function**

Function description	
Declaration	Public Declare Function FEISC_NewReader Lib "FEISC.DLL" (ByVal iPortHnd As Long) As Long
Prototype	lngAttachedDeviceHandle(1) = FEISC_NewReader(iHandle)
Parameters	<b>iHandle</b> : USB reader handle which has been filled in after the reader detection process for FEUSB functions (FEUSB_OpenDevice)
Returned value	<b>lngAttachedDeviceHandle(1)</b> : USB reader handle for FEISC functions Otherwise FEISC error code

##### FEISC\_NewReader C/C++ prototype

[Table 12](#) illustrates the FEISC\_NewReader C/C++ function.

**Table 12. FEISC\_NewReader C/C++ function**

Function description	
Declaration	int DLL_EXT_FUNC FEISC_NewReader( int iPortHnd );
Prototype	iFeiscHandle = FEISC_NewReader(iDeviceHandle)
Parameters	<b>iHandle</b> : USB reader handle which has been filled in after the reader detection process for FEUSB functions (FEUSB_OpenDevice)
Returned value	<b>lngAttachedDeviceHandle(1)</b> : USB reader handle for FEISC functions Otherwise FEISC error code

## 4.2 Detection and connection sequence

The sequence required to perform reader detection and connection is the following:

1. Call the **FEUSB\_ClearScanList** function to re-initialize the USB detection process.
2. Call the **FEUSB\_Scan** function to create the list of USB readers connected to the host ports. The list is stored in the oSearch variable. An error code is sent back if no reader have been detected.
3. Call the **FEUSB\_GetScanListSize** function to obtain the number of detected readers.
4. Call the **FEUSB\_GetScanListPara** function to retrieve the data related to the detected readers, together with the identifiers (Device-ID) assigned to each detected reader.
5. Call the **FEUSB\_OpenDevice** function to open a communication channel between an USB reader, identified by Device-ID, and the **FEUSB dll**. The function sends back the **USB handle** that will be used to manage further FEUSB communication with the reader.
6. Call the **FEISC\_NewReader** function to open a communication channel between an USB reader, identified by its **USB handle**, and the **FEISC dll**. The function sends back the **attached device handle** (lngAttachedDeviceHandle variable) that will be used to manage further FEISC communication with the reader.  
 The `detect_FEIG_USB_reader()` function returns TRUE if a FEIG USB reader is detected and FALSE otherwise.
7. When data transmissions and receptions have completed, call the **FEUSB\_CloseDevice** function to close USB communication.

Hereafter are source code examples of reader detection and connection examples in Visual Basic and C/C++.

## 4.3 Visual Basic source code example

```
'--- global variable to manage FEUSB handle
Public lngFEUSBhandle as long
'--- global variable to manage FEISC handle
Public lngAttachedDeviceHandle() as long
'--- Detect FEIG USB readers (detected=true, non-detected=false) ---
Private Function detect_USB_FEIG_reader() As Boolean
    Dim Back As Long
    Dim i As Long
    Dim Cnt As Integer
    Dim Char As String
    Dim CharCnt As Integer
    Dim oSearch As FEUSB_SCANSEARCH
    Dim lngReaderNumber As Long
    Dim error As Long
    Dim DeviceHnd As Long

    Dim lngDeviceId(0 To 15) As Long
    Dim strFamilyName(0 To 15) As String
    Dim strDeviceName(0 To 15) As String
    Dim lngPresent(0 To 15) As Long
```

```

Dim strDeviceHnd(0 To 15) As String * 8
Dim strPresent(0 To 15) As String * 25
Dim strDeviceID(0 To 15) As String * 8

Dim dwHandle As Long
Dim iHandle As Long

'init display
'txtDetectResult.Text = ""
oSearch.iMask = 0

'--- Clear Scan List (dll function)
FEUSB_ClearScanList ' should be called before every scan process
'--- scan and open in one process (dll function)
DeviceHnd = FEUSB_Scan(FEUSB_SCAN_ALL, oSearch)

If (DeviceHnd = FEUSB_ERR_NO_DEVICE_FOUND) Then
'--- txtScannedReaders.Text = "NO READER DETECTED"
Else
'---Number of USB device connected
lngReaderNumber = FEUSB_GetScanListSize
For i = 1 To lngReaderNumber
'--- Get System parameters of all detected USB devices
error = FEUSB_GetScanListPara(0, "DeviceHnd", strDeviceHnd(i))
error = FEUSB_GetScanListPara(0, "FamilyName",
strFamilyName(i))
error = FEUSB_GetScanListPara(0, "DeviceName",
strDeviceName(i))
error = FEUSB_GetScanListPara(0, "Present", strPresent(i))
error = FEUSB_GetScanListPara(0, "Device-ID", strDeviceID(i))
'--- decode lngDeviceHnd(i) value
CharCnt = 0
lngDeviceHnd(i) = 0
For Cnt = 1 To 8
Char = UCase(Mid(strDeviceHnd(i), 9 - Cnt, 1))
If ((Char >= "0") And (Char <= "9")) Then
lngDeviceHnd(i) = lngDeviceHnd(i) + (Asc(Char) - 48) * _
(16 ^ CharCnt)
CharCnt = CharCnt + 1
End If
If ((Char >= "A") And (Char <= "F")) Then
lngDeviceHnd(i) = lngDeviceHnd(i) + (Asc(Char) - 55) * _
(16 ^ CharCnt)
CharCnt = CharCnt + 1
End If
Next Cnt
'--- decode strFamilyName(i) value
strFamilyName(i) = ""
For Cnt = 1 To 25
Char = UCase(Mid(strFamilyName(i), Cnt, 1))
If (((Char >= "0") And (Char <= "9")) Or _
((Char >= "A") And (Char <= "Z"))) Or _

```

```

        (Char = "-" ) Or (Char = "." ) Or (Char = " ") ) Then
            strFamilyName(i) = strFamilyName(i) & Char
        End If
    Next Cnt
    '--- decode strDeviceName(i) value
    strDeviceName(i) = ""
    For Cnt = 1 To 25
        Char = UCase(Mid(strDeviceName(i), Cnt, 1))
        If (((Char >= "0") And (Char <= "9")) Or ((Char >= "A") _
            And (Char <= "Z")) Or (Char = "-") Or (Char = ".") Or _
            (Char = " ")) Then
            strDeviceName(i) = strDeviceName(i) & Char
        End If
    Next Cnt
    '-- decode strDeviceID value
    CharCnt = 0
    lngDeviceId(i) = 0
    For Cnt = 1 To 8
        Char = UCase(Mid(strDeviceID(i), 9 - Cnt, 1))
        If ((Char >= "0") And (Char <= "9")) Then
            lngDeviceId(i) = lngDeviceId(i) + (Asc(Char) - 48) * _
                (16 ^ CharCnt)
            CharCnt = CharCnt + 1
        End If
        If ((Char >= "A") And (Char <= "F")) Then
            lngDeviceId(i) = lngDeviceId(i) + (Asc(Char) - 55) * _
                (16 ^ CharCnt)
            CharCnt = CharCnt + 1
        End If
    Next Cnt
    '-- decode strPresent value
    CharCnt = 0
    lngPresent(i) = 0
    For Cnt = 1 To 8
        Char = UCase(Mid(strPresent(i), 9 - Cnt, 1))
        If ((Char >= "0") And (Char <= "9")) Then
            lngPresent(i) = lngPresent(i) + (Asc(Char) - 48) * _
                (16 ^ CharCnt)
            CharCnt = CharCnt + 1
        End If
        If ((Char >= "A") And (Char <= "F")) Then
            lngPresent(i) = lngPresent(i) + (Asc(Char) - 55) * _
                (16 ^ CharCnt)
            CharCnt = CharCnt + 1
        End If
    Next Cnt

    '--- summary of Detected Device parameters for One USB device
    '- txtDeviceHandle.Text = "DeviceHnd : " & lngDeviceHnd(i)
    '- txtFamilyName.Text = "FamilyName : " & strFamilyName(i)
    '- txtDeviceName.Text = "DeviceName : " & strDeviceName(i)
    '- txtPresent.Text = "Present : " & CLng("&h" & lngPresent(i))

```

```

        '- txtDeviceID.Text = "Device-ID : " & lngDeviceId(i)
    Next i
End If
'--- analysis if FEIG USB readers are connected and how much
If (lngReaderNumber = 0) Then
    '--- NO FEIG USB device detected
    `txtDetectResult.Text = "No FEIG USB Reader detected"
    detect_USB_FEIG_reader = False
ElseIf (lngReaderNumber > 1) Then
    '--- TOO MUCH DETECTED FEIG USB READERS
    `txtDetectResult.Text = "Too much FEIG USB Reader detected"
    detect_USB_FEIG_reader = False
ElseIf (lngReaderNumber = 1) Then
    '--- only One FEIG USB device detected OK
    glngFEUSBhandle = FEUSB_OpenDevice(lngDeviceId(1))
    If (glngFEUSBhandle < 0) Then
        error = FEUSB_CloseDevice(glngFEUSBhandle)
        detect_USB_FEIG_reader = False
    Else
        lngAttachedDeviceHandle(0) = FEISC_NewReader(glngFEUSBhandle)
        detect_USB_FEIG_reader = True
    End If
End If
End Function

```

## 4.4 C/C++ source code example

```

/* global variables */
int iDeviceHandle;
int iFeiscHandle;
int main ()
{
    char  sDeviceHandle[32];
    char  sDeviceName[32];
    char  sDeviceSerialNumber[32];
    char  sDeviceFamilyName[32];
    char  sDevicePresence[2];
    DWORD dwDeviceSerialNumber = 0;

    FEUSB_ClearScanList();/* check USB device connected */

    while (FEUSB_Scan(FEUSB_SCAN_FIRST,NULL) < 0)
    {
        /* time out to be added toexit while loop If no reader connected
        */
        /*return 0;
        */
    }
    /* GetScanListPara : Get Detected FEIG USB device parameters */
    FEUSB_GetScanListPara( 0, "DeviceName", sDeviceName ) ;
    FEUSB_GetScanListPara( 0, "Device-ID", sDeviceSerialNumber ) ;
    FEUSB_GetScanListPara( 0, "DeviceHnd", sDeviceHandle ) ;

```

```
FEUSB_GetScanListPara( 0, "FamilyName", sDeviceFamilyName ) ;
FEUSB_GetScanListPara( 0, "Present", sDevicePresence ) ;

/* convert receive data in hexa format */
sscanf((const char*)sDeviceSerialNumber, "%lx",
&dwDeviceSerialNumber);

/* Open Communciation with one FEIG USB device */
/* and get USB hanle for FEUSB functions */
iDeviceHandle = FEUSB_OpenDevice(dwDeviceSerialNumber);

if (iDeviceHandle < 0)
{
    /* Error : USB connection problem */
    /* Close USB connection */
    iErr = FEUSB_CloseDevice(iDeviceHandle);
    return 0;
}
/* Link FEIG USB device with FEISC functions */
/* and get USB hanle */
iFeiscHandle = FEISC_NewReader( iDeviceHandle );

if (iFeiscHandle < 0)
{
    /* Error : USB connection problem */
    /* Close USB connection */
    iErr = FEISC_DeleteReader( iFeiscHandle );
    return 0;
}
Else
{
    /* USB connection OK */
    /* iDeviceHandle is handle for FEUSB functions */
    /* iFeiscHandle is handle for FEISC functions */
    return 1;
}
}
```



## 5 RF ISO 15693 High-level commands

The `FEISC_0xB0_ISOCmd` function is part of the `FEISC.dll`. It allows sending predefined RF ISO 15693 commands to RF transponders via the FEIG readers, and retrieving the transponder answers. The reader is identified by its attached device handle contained in the `IngAttachedDeviceHandle(1)` variable which must have been previously filled in with the correct FEIG USB reader handle by calling the `FEISC_NewReader` function (see [Section 4.2: Detection and connection sequence](#)).

Among the ISO 15693 commands that can be used by FEIG readers to communicate with M24LRXX transponders are:

- Inventory: command code is '0x01 00'
- RESET TO READY: command code is '0x26 00'

Refer to the M24LRXX datasheet and to user manuals available from <http://www.feig.de> for the full list of RF ISO 15693 requests supported by the FEIG readers and ST M24LRXX Dual interface EEPROM.

Refer to [Section 5.1](#) for a general description of the `FEISC_0xB0_ISOCmd` function, and to [Section 5.2](#) and [Section 5.3](#) for details on the High-level Inventory and READY TO READY functions.

Refer to Appendix [C.2](#) for the full list of error codes returned by the FEISC functions.

## 5.1 FEISC\_0xB0\_ISOCmd general description

[Table 13](#) and [Table 14](#) give the description of the FEISC\_0xB0\_ISOCmd function in Visual Basic and C/C++.

**Table 13. FEISC\_0xB0\_ISOCmd Visual Basic prototype**

Function description	
Prototype	<pre>IngStatus = FEISC_0xB0_ISOCmd (lngAttachedDeviceHandle(1), &amp;hFF, strReqData, lngReqDataLen, strRespData, lngRespDataLen, N)</pre>
Parameters	<p><b>lngAttachedDeviceHandle(1)</b>: USB reader handle which has been filled in after the reader detection process (<code>FEISC_NewReader</code> function).</p> <p><b>&amp;hFF</b>: Communication address.</p> <p><b>strReqData</b>: High-level command to be sent to the reader.</p> <p><b>lngReqDataLen</b>: High-level command length.</p> <p><b>strRespData</b>: Transponder answer (if any). This variable is filled after the USB request is issued. Before sending any USB request, <b>strRespData</b> must be formatted as follows: Dim strRespData As String * 512</p> <p><b>lngRespDataLen</b>: size of the RF answer (length of <b>strRespData</b>).</p> <p><b>N</b>: Format of all the parameters passed to the function</p> <p>0: ASCII. Example: '2356' corresponds to the string '#V' composed of 2 ASCII codes.</p> <p>1: Characters. Example: '2356' corresponds to 4 characters (2, 3, 5, 6)</p> <p>2: Hexadecimal. Example: '2356' corresponds to 2 bytes (&amp;h23 and &amp;h56)</p>
Returned value	<p><b>IngStatus</b>: error code</p> <p>0: USB request transmission successful</p> <p>1: USB request transmission failed</p>

**Table 14. FEISC\_0xB0\_ISOCmd C/C++ prototype**

Function description	
Prototype	<code>result = FEISC_0xB0_ISOCmd(iFeiscHandle, 0xFF, sReqData, iReqLen, sRspData, &amp;iRspLen, N);</code>
Parameters	<p><b>iFeiscHandle:</b> USB reader handle which has been filled in after the reader detection process (<code>FEISC_NewReader</code> function).</p> <p><b>0xFF:</b> Communication address.</p> <p><b>sReqData:</b> High-level command to be sent to the reader.</p> <p><b>iReqLen:</b> High-level command length.</p> <p><b>sRspData:</b> Transponder answer (if any). This variable is filled after the USB request is issued. Before sending any USB request, <b>strRespData</b> must be formatted as follows:  <code>char sRspData[64]={0}</code></p> <p><b>&amp;iRspLen:</b> size of the RF answer (length of <b>sRspData</b>).</p> <p><b>N:</b> Format of all the parameters passed to the function</p> <p>0: ASCII.                      Example: '2356' corresponds to the string '#V' composed of 2 ASCII codes.</p> <p>1: Characters.                      Example: '2356' corresponds to 4 characters (2, 3, 5, 6)</p> <p>2: Hexadecimal.                      Example: '2356' corresponds to 2 bytes (0x23 and 0x56)</p>
Returned value	<p><b>result:</b> error code</p> <p>0: USB request transmission successful</p> <p>1: USB request transmission failed</p>

## 5.2 RF ISO 15693 High-level Inventory command

To issue an RF ISO 15693 High-level Inventory command, send the `FEISC_0xB0_ISOCmd` function with the `strReqData` set to '0100' (Inventory request).

At the end of an inventory request, all the transponders are put in Quiet mode (ISO 15693 "STAY QUIET"). To be able to communicate with the detected transponder. A "RESET TO READY" RF request must be issued.

Below are code examples in Visual Basic and C/C++.

### 5.2.1 Example of High-level Inventory command Visual Basic source code

```
Private Function Cmd_Inventory_FEIG() As Boolean
Dim strReqData As String
Dim lngReqDataLen As Long
Dim lngRspLength As Long
Dim strRespData As String * 512
Dim lngRespDataLen As Long
Dim lngStatus As Long
Dim i As Long
Dim lngTranspNumber As Long
Dim strtransponder As String
'init display
'txtInventoryRF_answer.Text = ""
'For i = 0 To 2
'    txtTransponderUID(i).Text = ""
'    txtTransponderDSFID(i).Text = ""
'Next i

' Inventory request Host mode : 0x0100
strReqData = "0100"
lngReqDataLen = Len(strReqData)
' FEIG USB INVENTORY request in Host mode
lngStatus = FEISC_0xB0_ISOCmd(lngAttachedDeviceHandle(0), &HFF, _
strReqData, lngReqDataLen, strRespData, _
lngRespDataLen, 1)

' RF INVENTORY REQUEST RESULT
' if(lngStatus = 0) then PASS else FAIL
' if (lngRespDataLen = 0) then No transponder answer
' else strRespData contains the transponder(s) answer(s)

If (lngRespDataLen = 0) Then

    Cmd_Inventory_FEIG = False
    'txtInventoryRF_answer.Text = "No Tag answer detected"
Else
    lngTranspNumber = CLng("&h" & Mid(strRespData, 1, 2))
    If (lngRespDataLen > 1 And lngTranspNumber > 3) Then
        Cmd_Inventory_FEIG = False
        'Too much Tags detected :
```

```

        Else
            Cmd_Inventory_FEIG = True
            `lngTranspNumber & " Tags Detected"
For i = 0 To lngTranspNumber - 1
            strtransponder = Mid(strRespData, 3 + (20 * (i)), 20)
            Next i
        End If
    End If
End If

' after INVENTORY request, all transponders are in QUIET mode
' RESET TO READY request is sent to Wake Up transponders
Call cmd_ResetToReadyRF_FEIG
End Function

```

## 5.2.2 Example of High-level Inventory command C/C++ source code

```

int Cmd_Inventory_FEIG (void)
{

    int entry3;
    int i;
    UCHAR sReqData[64]={0};
    UCHAR sRspData[64]={0};
    int iReqLen,iRspLen;
    int iResult, iResult2;
    int iRspLength=56;

    /* Inventory request HOST MODE command : B0 + 0100 */
    sReqData[0] = (UCHAR)0x01;
    sReqData[1] = (UCHAR)0x00;
    iReqLen = 2; /* (number of bytes :param=2 in request) */

    printf("\n\n\n\n");
    printf("\n>>> INVENTORY request in ISO mode : ");
    printf("\n --> request : ");
    for (i=0; i<iReqLen; i++) printf("%.2x",sReqData[i]);

    /* FEIG USB INVENTORY request in HOST MODE command */
    iResult = FEISC_0xB0_ISOCmd(iFeiscHandle,0xFF,
        sReqData, iReqLen,/* request */
        &sRspData[0], &iRspLen,/* answer */
        2);/* length format 2 : Number of Bytes */

    printf("\n --> answer : ");
    if (iRspLen == 0)
        printf("No tag answer received");
    else
        for (i=0; i<iRspLen; i++) printf("%c",sRspData[i]);

    printf("\n\n\n\n");
    printf("\npress any key to continue");
}

```

```
printf("\n");
scanf("%x", &entry3);

/* RF INVENTORY REQUEST RESULT */
/* if(iResult == 0) PASS else FAIL */
/* if (iRspLen == 0) No transponder answer */
/* else sRspData contains the transponder(s) answer(s) */
if (iResult != 0)
{
    /* No Tag detected in the Antenna Field */
}
else
{
    /* 1 or more transponders are in Antenna Field */
}

/* after INVENTORY request, all transponders are in QUIET mode */
/* RESET TO READY request is sent to Wake Up transponders */
/* RESET TO READY [0xB0] request */
sReqData[0] = (UCHAR)0x26;
sReqData[1] = (UCHAR)0x00;
iReqLen = 2; /* (number of bytes :param=2 in request) */

iResult2 = FEISC_0xB0_ISOCmd(iFeiscHandle, 0xFF,
    sReqData, iReqLen, /* request */
    &sRspData[0], &iRspLen, /* answer */
    2); /* length format 2 */

if (iResult2 != 0)
{
    /* Reset to ready request problem */
}
else
{
    /* Reset to ready request OK */
}

return iResult;
}
```

## 5.3 RF ISO 15693 High-level RESET TO READY command

To issue an RF ISO 15693 High-level RESET TO READY command, send the FEISC\_0xB0\_ISOCmd function with the strReqData set to '0026' (RESET TO READY request).

Below is an example of code in Visual Basic and C/C++.

### 5.3.1 RESET TO READY command Visual Basic source code example

```
Private Function cmd_ResetToReadyRF_FEIG() As Boolean

Dim strReqData As String
Dim lngReqDataLen As Long
Dim lngRspLength As Long
Dim strRespData As String * 512
Dim lngRespDataLen As Long
Dim lngStatus As Long

' Reset to Ready request Host mode : 0x2600
strReqData = "2600"

lngReqDataLen = Len(strReqData)

' FEIG USB INVENTORY request in Host Mode
lngStatus = FEISC_0xB0_ISOCmd(lngAttachedDeviceHandle(0), &HFF, _
strReqData, lngReqDataLen, _
strRespData, lngRespDataLen, _
1)
' RF INVENTORY REQUEST RESULT
' if(lngStatus = 0) then PASS else FAIL
' if (lngRespDataLen = 0) then No transponder answer
' else strRespData contains the transponder(s) answer(s)

If (lngRespDataLen = 0) Then
cmd_ResetToReadyRF_FEIG = False
Else
cmd_ResetToReadyRF_FEIG = True
End If
End Function
```

## 6 RF ISO 15693 Transparent commands

The `FEISC_0xBF_ISOTranspCmd` function is part of the `FEISC.dll`. It allows sending any RF ISO 15693 request to RF transponders via the FEIG readers.

The reader is identified by its attached device handle contained in the `IngAttachedDeviceHandle(1)` variable which must have been previously filled in with the correct FEIG USB reader handle by calling the `FEISC_NewReader` function (see [Section 4.2: Detection and connection sequence](#)).

Several parameters must be passed to the `FEISC_0xBF_ISOTranspCmd` function to indicate to the reader the type of RF request, and the type of transponder answer expected.

Refer to [Section 6.1](#) for a general description of the `FEISC_0xB0_ISOCmd` function. All the requests described in the M24LRXX datasheet can be issued by using this method. [Section 6.2](#) and [Section 6.3](#) illustrate two examples of requests, the Read single block and Write single block request, which allow to read and write a single block of Dual interface memory.

For more informations about how to use Transparent commands, please refers to FEIG documentation available from <http://www.feig.de>.



## 6.1 FEISC\_0xBF\_ISOTranspCmd general description

[Table 15](#) and [Table 16](#) give the description of the FEISC\_0xBF\_ISOTranspCmd function in Visual Basic and C/C++.

**Table 15. FEISC\_0xBF\_ISOTranspCmd Visual Basic prototype**

Function description	
Prototype	<pre>IngStatus = FEISC_0xBF_ISOTranspCmd (lngAttachedDeviceHandle(1) , &amp;hFF, M, lngRspLength, strReqData, lngReqDataLen, strRespData, lngRespDataLen, N)</pre>
Parameters	<p><b>lngAttachedDeviceHandle(1):</b> USB reader handle which has been filled in after the reader detection process (FEISC_NewReader function).</p> <p><b>&amp;hFF:</b> Communication address.</p> <p><b>M:</b> Mode. The method used by the Transparent command to detect the transponder answer depends on the mode.</p> <ul style="list-style-type: none"> <li>1: answer detected after reception of a Read request</li> <li>2: answer detected after reception of a Write request with Option_flag = 0.</li> <li>3: answer detected after reception of a Write request with Option_flag = 1.</li> <li>4: answer detected after reception of an Inventory request</li> </ul> <p><b>lngRspLength:</b> Expected RF answer size (bit number)</p> <p><b>strReqData:</b> RF request frame to be sent to the reader</p> <p><b>lngReqDataLen:</b> RF request frame length (strReqData)</p> <p><b>strRespData:</b> Transponder answer (if any). This variable is filled after the USB request is issued. Before sending any USB request, <b>strRespData</b> must be formatted as follows:</p> <p>Dim strRespData As String * 512</p> <p><b>lngRespDataLen:</b> strRespData length (0 if no answer)</p> <p><b>N:</b> Format of all the parameters passed to the function</p> <ul style="list-style-type: none"> <li>0: ASCII. Example: '2356' corresponds to the string '#V' composed of 2 ASCII codes.</li> <li>1: Characters. Example: '2356' corresponds to 4 characters (2, 3, 5, 6)</li> <li>2: Hexadecimal. Example: '2356' corresponds to 2 bytes (&amp;h23 and &amp;h56)</li> </ul>
Returned value	<p><b>IngStatus:</b> error code</p> <ul style="list-style-type: none"> <li>0: USB request transmission successful</li> <li>1: USB request transmission failed</li> </ul>

**Table 16. FEISC\_0xBF\_ISOTranspCmd C/C++ prototype**

Function description	
Prototype	<pre>iResult = FEISC_0xBF_ISOTranspCmd(iFeiscHandle, 0xFF, M, iRspLength, &amp;sReqData[0], iReqLen, &amp;sRspData[0],&amp;iRspLen, N);</pre>
Parameters	<p><b>iFeiscHandle:</b> USB reader handle which has been filled in after the reader detection process (FEISC_NewReader function).</p> <p><b>0xFF:</b> Communication address.</p> <p><b>M:</b> Mode. The method used by the Transparent command to detect the transponder answer depends on the mode.</p> <ul style="list-style-type: none"> <li>1: answer detected after reception of a Read request</li> <li>2: answer detected after reception of a Write request with Option_flag = 0.</li> <li>3: answer detected after reception of a Write request with Option_flag = 1.</li> <li>4: answer detected after reception of an Inventory request</li> </ul> <p><b>&amp;iRspLength:</b> Expected RF answer size (bit number)</p> <p><b>&amp;sReqData:</b> RF request frame to be sent to the reader.</p> <p><b>iReqLen:</b> RF request frame length (sReqData)</p> <p><b>&amp;sRspData:</b> Transponder answer (if any). This variable is filled after the USB request is issued. Before sending any USB request, <b>sRspData</b> must be formatted as follows:  char sRspData[64]={0}</p> <p><b>&amp;isRspLen:</b> sRspData length (0 if no answer)</p> <p><b>N:</b> Format of all the parameters passed to the function</p> <ul style="list-style-type: none"> <li>0: ASCII.  Example: '2356' corresponds to the string '#V' composed of 2 ASCII codes.</li> <li>1: Characters.  Example: '2356' corresponds to 4 characters (2, 3, 5, 6)</li> <li>2: Hexadecimal.  Example: '2356' corresponds to 2 bytes (0x23 and 0x56)</li> </ul>
Returned value	<p><b>iResult:</b> error code</p> <ul style="list-style-type: none"> <li>0: USB request transmission successful</li> <li>1: USB request transmission failed</li> </ul>

## 6.2 Issuing a Read single block request with a transparent command

[Table 17](#) and [Table 18](#) give an example of parameters to be passed to the `FEISC_0xBF_ISOTranspCmd` function to issue a Transparent Read single block command. [Section 6.2.1](#) and [Section 6.2.2](#) describe code examples in Visual Basic and C/C++.

**Table 17. Example of Read single block command in Visual Basic**

FEISC_0xBF_ISOTranspCmd parameters	
Parameters	<b>lngAttachedDeviceHandle(0)</b> : USB Handle <b>&amp;hFF</b> : Communication address <b>M</b> : 1: Read mode <b>lngRespLength</b> : 0x38 <b>strReqData</b> : 021F: Reader parameter 0A: RF protocol flag request 20: RF ISO 15693 Read single block command FA01: address &h01FA <b>lngReqDataLen</b> : 12 <b>strRespData</b> : answer from Transponder (if any) <b>lngRespDataLen</b> : size of strRespData <b>N</b> :1 (data expressed in characters)
Returned value	Error code
Example	<pre>FEISC_0xBF_ISOTranspCmd(lngAttachedDeviceHandle(0), &amp;hFF, 1, &amp;h38, '021F0A20FA01', lngReqDataLen, strRespData, lngRespDataLen, 1)</pre>

Table 18. Example of Read single block command in C/C++

FEISC_0xBF_ISOTranspCmd parameters	
Parameters	<b>IFeiscHandle:</b> USB Handle <b>0xFF:</b> Communication address <b>M:</b> 1: Read mode <b>iRspLength:</b> 0x38 <b>sRqData:</b> 021F: Reader parameter 0A: RF protocol flag request 20: RF ISO 15693 Read single block command FA01: address 0x01FA <b>iReqLen:</b> 6 <b>sRspData:</b> answer from Transponder (if any) <b>IRspLen:</b> size of sRspData <b>N:</b> 2 (data expressed in hexadecimal bytes)
Returned value	Error code
Example	<pre>iResult = FEISC_0xBF_ISOTranspCmd(iFeiscHandle, 0xFF, 1, iRspLength, '021F0A20FA01', iReqLen, &amp;sRspData[0], &amp;iRspLen, 2);</pre>

## 6.2.1 Transparent Read single block command Visual Basic source code example

```
Private Function RFReadsingleBlock() As Boolean
Dim strReqData As String
Dim lngReqDataLen As Long
Dim lngRspLength As Long
Dim strRespData As String * 512 ' has to be formatted
Dim lngRespDataLen As Long
Dim lngStatus As Long
'init display
'txtReadRF_answer.Text = ""
'txtReadData.Text = ""
lngRspLength = &H38 'response length (Feig USB Reader)

' RF READ SINGLE BLOCK request
' FEIG reader parameters 0X021F
' Flag 0x0A
' RF Read command 0X20
' Address 0x01FA : send FA01
strReqData = "021F0A20FA01"
lngReqDataLen = Len(strReqData)
' SEND COMMAND IN transparent mode
lngStatus = _
    FEISC_0xBF_ISOTranspCmd(lngAttachedDeviceHandle(0), &HFF, _
        &H1, lngRspLength, _
        strReqData, lngReqDataLen, _
        strRespData, lngRespDataLen, _
```

```

        1)
' RF READ REQUEST RESULT
' if(lngStatus = 0) then PASS else FAIL
' if (lngRespDataLen = 0) then No transponder answer
' else strRespData contains the transponder answer

If (lngRespDataLen = 0) Then
    RFReadsingleBlock = False
    `txtReadRF_answer.Text = "No detected Tag answer"
Else
    If (Mid(strRespData, 1, 2) = "00") Then
        RFReadsingleBlock = True
        `txtReadRF_answer.Text = strRespData & " = read single block
OK"
    `txtReadData.Text = Mid(strRespData, 3, 8)
    Else
        RFReadsingleBlock = False
        `txtReadRF_answer.Text = strRespData & " = Error code"
    End If
End If
End Function

```

## 6.2.2 Transparent Read single block command C/C++ source code example

```

int RFReadsingleBlock (void)
{
    int entry3;
    int i;
    int iRspLength=56;
    int iReqLen,iRspLen,iResult;
    unsigned char sReqData[32]={0};
    unsigned char sRspData[32]={0};
    /* RF READ SINGLE BLOCK request format */
    /* FEIG reader parameters : 021F*/
    /* Flag : 0A */
    /* RF Read single block command : 20*/
    /* Address :01FA (note : send FA01 = LSB BYTE first)*/
    /* request = 021F + 0A20FA01*/
    sReqData[0] = (UCHAR)0x02;
    sReqData[1] = (UCHAR)0x1F;
    sReqData[2] = (UCHAR)0x0A;
    sReqData[3] = (UCHAR)0x20;
    sReqData[4] = 0xFA
    sReqData[5] = 0x01
    iReqLen = 6; /* (number of characters :param=2 in request) */
    printf("\n\n\n\n");
    printf("\n>>> RF Read at adress %.2x%.2x : ",Address_parameter[0],
Address_parameter[1]);
    printf("\n --> request : {021F} ");
    for (i=2; i<iReqLen; i++) printf("%.2X",sReqData[i]);
    /* FEIG USB request in transparent mode */

```

```
    iResult = FEISC_0xBF_ISOTranspCmd (iFeiscHandle, 0xFF,
    1, /* MODE 1 : read answer */
    iRspLength, sReqData, /* request */
    iReqLen, /* USB request length */
    &sRspData[0], &iRspLen, /* answer */
    2 ); /* length format 2 : Number of Bytes */
printf("\n --> answer : ");
if (iRspLen == 0)
    printf("No tag answer received");
else
    for (i=0; i<iRspLen; i++) printf("%c",sRspData[i]);
printf("\n\n\n\n");
printf("\npress any key to continue");
printf("\n");
scanf("%x", &entry3);
/* RF REQUEST RESULT */
/* if(iResult == 0) PASS else FAIL */
/* if (iRspLen == 0) No transponder answer */
/* else sRspData contains the transponder answer */
return iResult;
}
```

## 6.3 Issuing a Write single block request with a transparent command

[Table 19](#) and [Table 20](#) gives an example of parameters to be passed to the `FEISC_0xBF_ISOTranspCmd` function to issue a Transparent Write single block command. [Section 6.3.1](#) and [Section 6.3.2](#) describe code examples in Visual Basic and C/C++.

**Table 19. Example of Write single block command in Visual Basic**

FEISC_0xBF_ISOTranspCmd parameters	
Parameters	<b>lngAttachedDeviceHandle(0)</b> : USB Handle <b>&amp;hFF</b> : Communication address= 0xFF <b>M</b> : 2: Write mode <b>lngRespLength</b> : &h18 <b>strReqData</b> : 021F: Reader parameter 0A: RF protocol flag request 21: RF ISO 15693 Write single block command FA01: address 0x01FA 01020304: data to be written <b>lngReqDataLen</b> : 20 <b>strRespData</b> : answer from Transponder (if any) <b>lngRespDataLen</b> : size of strRespData <b>N</b> :1 (data expressed in characters)
Returned value	Error code
Example	<pre>FEISC_0xBF_ISOTranspCmd(lngAttachedDeviceHandle(0), &amp;hFF, 2, 1, &amp;h18, '021F0A21FA0101020304', lngReqDataLen, strRespData, lngRespDataLen, 1)</pre>

Table 20. Example of Write single block command in C/C++

FEISC_0xBF_ISOTranspCmd parameters	
Parameters	<b>IFeiscHandle:</b> USB Handle <b>0xFF:</b> Communication address <b>M: 2</b> (Write mode) <b>IRspLength:</b> 0x18 <b>strReqData:</b> 021F: Reader parameter 0A: RF protocol flag request 21: RF ISO 15693 Write single block command FA01: address 0x01FA 01020304: data to be written <b>IReqLen:</b> 10 <b>&amp;sRspData:</b> answer from Transponder (if any) <b>&amp;IRspLen:</b> size of sRspData <b>N:</b> 2 (data expressed in hexadecimal bytes)
Returned value	Error code
Example	<pre>iResult = FEISC_0xBF_ISOTranspCmd(iFeiscHandle, 0xFF, 2, iRspLength, '021F0A20FA01', iReqLen, &amp;sRspData[0], &amp;iRspLen, 2);</pre>

### 6.3.1 Transparent Write single block command Visual Basic source code example

```
Private Function WriteSingleBlockRF() As Boolean
Dim strReqData As String
Dim lngReqDataLen As Long
Dim lngRspLength As Long
Dim strRespData As String * 512 ' has to be formatted
Dim lngRespDataLen As Long
Dim lngStatus As Long
'init display
'txtWriteRF_answer.Text = ""
lngRspLength = &H18 'FEIG response length

' RF WRITE SINGLE BLOCK request
' FEIG reader parameters 0X021F
' Flag 0x0A
' RF Write Single Block command 0X21
' Address 0x01FA : send FA01
' Data 0x01020304
strReqData = "021F0A21FA0101020304"
lngReqDataLen = Len(strReqData)
lngStatus = _
    FEISC_0xBF_ISOTranspCmd(lngAttachedDeviceHandle(0), &HFF, _
&H2, lngRspLength, _
    strReqData, lngReqDataLen, _
    strRespData, lngRespDataLen, _
1)
```



```

' RF WRITE REQUEST RESULT
' if(lngStatus = 0) then PASS else FAIL
' if (lngRespDataLen = 0) then No transponder answer
' else strRespData contains the transponder answer
If (lngRespDataLen = 0) Then
  WriteSingleBlockRF = False
  `txtWriteRF_answer.Text = "No detected Tag answer"
Else
  If (Mid(strRespData, 1, 2) = "00") Then
    WriteSingleBlockRF = True
    `txtWriteRF_answer.Text = strRespData & " = write single
block OK"
  Else
    WriteSingleBlockRF = False
    `txtWriteRF_answer.Text = strRespData & " = Error code"
  End If
End If
End Function

```

### 6.3.2 Transparent Write single block command C/C++ source code example

```

int WriteSingleBlockRF (void)
{
  int entry3;
  int i;
  int iRspLength=0x18;
  int iReqLen,iRspLen,iResult;
  unsigned char sReqData[32]={0};
  unsigned char sRspData[32]={0};
  /* RF WRITE SINGLE BLOCK request format*/
  /* FEIG reader parameters : 021F*/
  /* Flag : 0A */
  /* RF Write command : 21*/
  /* address : 01FA (note: send FA01 = LSB BYTE first) */
  /* Data : 01020304*/
  /* request = 021F + 0A21FA0101020304*/
  sReqData[0] = (UCHAR)0x02;
  sReqData[1] = (UCHAR)0x1F;
  sReqData[2] = (UCHAR)0x0A;
  sReqData[3] = (UCHAR)0x21;
  sReqData[4] = Address_parameter[1];/* ex: 0x01FA -> FA 01 to be
sent */
  sReqData[5] = Address_parameter[0];
  sReqData[6] = Data_parameter[0];
  sReqData[7] = Data_parameter[1];
  sReqData[8] = Data_parameter[2];
  sReqData[9] = Data_parameter[3];
  iReqLen = 10; /* (number of characters :param=2 in request) */
  printf("\n\n\n\n");
}

```

```
    printf("\n>>> RF Write at adress %.2x%.2x data %.2x%.2x%.2x%.2x :
",Address_parameter[0],
Address_parameter[1],Data_parameter[0],Data_parameter[1],Data_param
eter[2], Data_parameter[3]);
    printf("\n    --> request : {021F} ");
    for (i=2; i<iReqLen; i++) printf("%.2X",sReqData[i]);
    iResult = FEISC_0xBF_ISOTranspCmd (iFeiscHandle, 0xFF,/* USB
parameters */
2,/* MODE 2 : write like answer */
iRspLength, sReqData,/* request */
iReqLen,/* USB request length */
&sRspData[0],&iRspLen,/* answer */
2 );/* length format 2 : Number of Bytes */

printf("\n    --> answer : ");
if (iRspLen == 0)
printf("No tag answer received");
else
for (i=0; i<iRspLen; i++) printf("%c",sRspData[i]);
printf("\n\n\n\n");
printf("\npress any key to continue");
printf("\n");
scanf("%x", &entry3);

/* RF REQUEST RESULT */
/* if(iResult == 0) PASS else FAIL */
/* if (iRspLen == 0) No transponder answer */
/* else sRspData contains the transponder answer */

return iResult;
}
```

## Appendix A FEIG reference documents

- Installation manual: M30100-3de-ID-B.pdf
- FEUSB manual: H00501-7e-ID-B.pdf
- FEISC manual: H9391-27e-ID-B.pdf
- Standard readers manual: H60700-2e-ID-B.pdf
- Standard readers manual: H60301-1e-ID-B.pdf

## Appendix B Useful source code zip files

The AN3224.zip package contains two simple projects to test the RF ISO 15693 High-level and Transparent commands. These projects can be used to understand how to develop an application to communicate with the FEIG RF USB readers:

- AN3224\_VB\_sourcecode folder contains the Visual Basic project
- AN3224\_C\_sourcecode folder contains the C/C++ project,

The AN3224.zip package can be downloaded from <http://www.st.com/dualeeprom>.

## Appendix C List of error codes

### C.1 FEUSB error codes

The error codes which are returned by the FEUSB ISO 15693 and I<sup>2</sup>C commands are the following:

#### C.1.1 Common errors

```
#define FEUSB_ERR_EMPTY_DEVICELIST -1100
#define FEUSB_ERR_EMPTY_SCANLIST -1101
#define FEUSB_ERR_POINTER_IS_NULL -1102
#define FEUSB_ERR_NO_MORE_MEM -1103
#define FEUSB_ERR_SET_CONFIGURATION -1104
#define FEUSB_ERR_KERNEL -1105
#define FEUSB_ERR_UNSUPPORTED_OPTION -1106
#define FEUSB_ERR_UNSUPPORTED_METHOD -1107
```

#### C.1.2 Scanning errors

```
#define FEUSB_ERR_NO_FEIG_DEVICE -1110
#define FEUSB_ERR_NO_FEIG_DEVICE -1110
#define FEUSB_ERR_SEARCH_MISMATCH -1111
#define FEUSB_ERR_NO_DEVICE_FOUND -1112
#define FEUSB_ERR_DEVICE_IS_SCANNED -1113
#define FEUSB_ERR_SCANLIST_OVERFLOW -1114
```

#### C.1.3 Handle errors

```
#define FEUSB_ERR_UNKNOWN_HND -1120
#define FEUSB_ERR_HND_IS_NULL -1121
#define FEUSB_ERR_HND_IS_NEGATIVE -1122
#define FEUSB_ERR_NO_HND_FOUND -1123
```

#### C.1.4 Communication errors

```
#define FEUSB_ERR_TIMEOUT -1130
#define FEUSB_ERR_TIMEOUT -1130
#define FEUSB_ERR_NO_SENDDATA -1131
#define FEUSB_ERR_UNKNOWN_INTERFACE -1132
#define FEUSB_ERR_UNKNOWN_DIRECTION -1133
#define FEUSB_ERR_RECBUF_TOO_SMALL -1134
#define FEUSB_ERR_SENDDATA_LEN -1135
#define FEUSB_ERR_UNKNOWN_DESCRIPTOR_TYPE -1136
#define FEUSB_ERR_DEVICE_NOT_PRESENT -1137
```

#### C.1.5 Open/close device errors

```
#define FEUSB_ERR_DEVICE_NOT_SCANNED -1140
#define FEUSB_ERR_DEVHND_NOT_IN_SCANLIST -1141
#define FEUSB_ERR_DRIVERLIST -1142
```

### C.1.6 Parameter errors

#define FEUSB_ERR_UNKNOWN_PARAMETER	-1150
#define FEUSB_ERR_PARAMETER_OUT_OF_RANGE	-1151
#define FEUSB_ERR_ODD_PARAMETERSTRING	-1152
#define FEUSB_ERR_INDEX_OUT_OF_RANGE	-1153
#define FEUSB_ERR_UNKNOWN_SCANOPTION	-1154
#define FEUSB_ERR_UNKNOWN_ERRORCODE	-1155

### C.1.7 Identification errors

#define FEUSB_ERR_DEV_DESC_LENGTH	-1160
#define FEUSB_ERR_CFG_DESC_LENGTH	-1161
#define FEUSB_ERR_INTF_DESC_LENGTH	-1162
#define FEUSB_ERR_ENDP_DESC_LENGTH	-1163
#define FEUSB_ERR_HID_DESC_LENGTH	-1164
#define FEUSB_ERR_STRG_DESC_LENGTH	-1165
#define FEUSB_ERR_READ_DEV_DESCRIPTOR	-1166
#define FEUSB_ERR_READ_CFG_DESCRIPTOR	-1167
#define FEUSB_ERR_READ_STRG_DESCRIPTOR	-1168
#define FEUSB_ERR_MAX_INTERFACES	-1170
#define FEUSB_ERR_MAX_ENDPOINTS	-1171
#define FEUSB_ERR_MAX_STRINGS	-1172

## C.2 FEISC error codes

### C.2.1 Common errors

#define FEISC_ERR_NEWREADER_FAILURE	-4000
#define FEISC_ERR_EMPTY_LIST	-4001
#define FEISC_ERR_POINTER_IS_NULL	-4002
#define FEISC_ERR_NO_MORE_MEM	-4003
#define FEISC_ERR_UNKNOWN_COMM_PORT	-4004
#define FEISC_ERR_UNSUPPORTED_FUNCTION	-4005
#define FEISC_ERR_NO_USB_SUPPORT	-4006
#define FEISC_ERR_OLD_FECOM	-4007

### C.2.2 Query errors

#define FEISC_ERR_NO_VALUE	-4010
----------------------------	-------

### C.2.3 Handle errors

#define FEISC_ERR_UNKNOWN_HND	-4020
#define FEISC_ERR_HND_IS_NULL	-4021
#define FEISC_ERR_HND_IS_NEGATIVE	-4022
#define FEISC_ERR_NO_HND_FOUND	-4023
#define FEISC_ERR_PORTHND_IS_NEGATIVE	-4024
#define FEISC_ERR_HND_UNVALID	-4025

### C.2.4 Communication errors

```
#define FEISC_ERR_PROTLEN -4030
#define FEISC_ERR_CHECKSUM -4031
#define FEISC_ERR_BUSY_TIMEOUT -4032
#define FEISC_ERR_UNKNOWN_STATUS -4033
#define FEISC_ERR_NO_RECPROTOCOL -4034
#define FEISC_ERR_CMD_BYTE -4035
#define FEISC_ERR_TRANSCEIVE -4036
#define FEISC_ERR_REC_BUS_ADR -4037
```

### C.2.5 Parameter errors

```
#define FEISC_ERR_UNKNOWN_PARAMETER -4050
#define FEISC_ERR_PARAMETER_OUT_OF_RANGE -4051
#define FEISC_ERR_ODD_PARAMETERSTRING -4052
#define FEISC_ERR_UNKNOWN_ERRORCODE -4053
#define FEISC_ERR_UNSUPPORTED_OPTION -4054
#define FEISC_ERR_UNKNOWN_EPC_TYPE -4055
```

### C.2.6 Plug-in errors

```
#define FEISC_ERR_NO_PLUGIN -4060
#define FEISC_ERR_PLUGIN_PRESENT -4061
#define FEISC_ERR_UNKNOWN_PLUGIN_ID -4062
#define FEISC_ERR_PI_BUILD_DATA -4063
#define FEISC_ERR_PI_BUILD_FRAME -4064
#define FEISC_ERR_PI_SPLIT_FRAME -4065
#define FEISC_ERR_PI_SPLIT_DATA -4066
```

### C.2.7 Communication data flow errors

```
#define FEISC_ERR_BUFFER_OVERFLOW -4070
```

### C.2.8 Task errors

```
#define FEISC_ERR_TASK_STILL_RUNNING -4080
#define FEISC_ERR_TASK_NOT_STARTED -4081
#define FEISC_ERR_TASK_TIMEOUT -4082
#define FEISC_ERR_TASK_SOCKET_INIT -4083
#define FEISC_ERR_TASK_BUSY -4084
#define FEISC_ERR_THREAD_CANCEL_ERROR -4085
```

## Revision history

**Table 21. Document revision history**

Date	Revision	Changes
15-Oct-2010	1	Initial release.
19-Sep-2011	2	Replaced part number "M24LR64-R" with "M24LRXX" throughout the document. Updated value of lngRspLength in <a href="#">Table 19: Example of Write single block command in Visual Basic</a> and <a href="#">Table 20: Example of Write single block command in C/C++</a> .



**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2011 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)