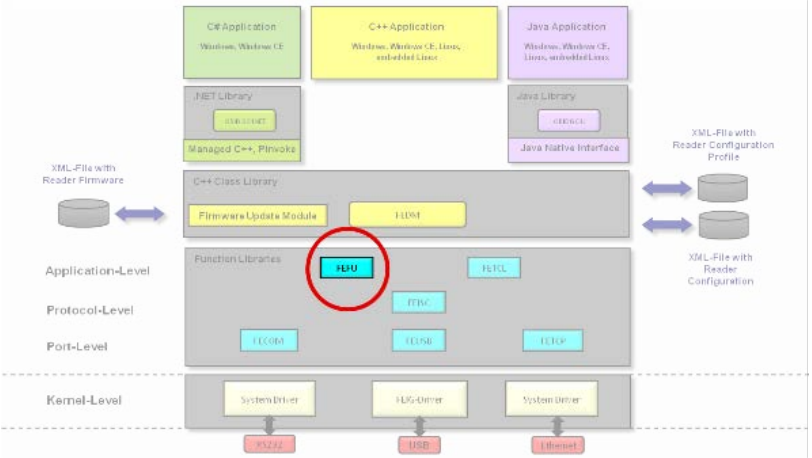
IDENTIFICATION

MANUAL

ID FEFU

Software Support for Function Units

Version 2.02.00



Operating System	Target		Notes
	32-Bit	64-Bit	
Windows Vista / 7 / 8 / 10	X	X	
Windows CE 6	X	-	On request
Linux	X	X	
Android	X	X	On request
Apple Max OS X	-	X	On request

final – public (B)
2018-04-19 – H30801-e-ID-B.docx

Note

FEIG ELECTRONIC GmbH
Lange Strasse 4
D-35781 Weilburg (Germany)
Tel.: +49 6471 3109-0
http://www.feig.de
identification-support@feig.de

With the edition of this manual, 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.
FEIG ELECTRONIC GmbH 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.
FEIG ELECTRONIC GmbH does not give any guarantee promise for perfect function of an Identification system in cross environments.

FEIG ELECTRONIC GmbH assumes no responsibility for the use of any information contained in this manual and makes no representation that they free of patent infringement. FEIG ELECTRONIC GmbH does not convey any license under its patent rights nor the rights of others.

All cited brand names, product names, or trademarks belong to their respective holders.

General Information Regarding this Document

- The sign "☞" indicates extensions or changes of this manual compared with the former issue.
- If bits within one byte are filled with "-", these bit spaces are reserved for future extensions or for internal testing- and manufacturing-functions. These bit spaces must not be changed, as this may cause faulty operation of the reader.
- The following figure formats are used:
  0...9: for decimal figures
  0x00...0xFF: for hexadecimal figures,
  b0...1 for binary figures.
- The hexadecimal value in brackets "[ ]" marks a control byte (command).

Licensing Agreement for Use of the Software

This is an agreement between you and FEIG ELECTRONIC GmbH (hereafter "FEIG") for use of the ID FEFU program library and the present documentation, hereafter called licensing material. By installing and using the software you agree to all terms and conditions of this agreement without exception and without limitation. If you are not or not completely in agreement with the terms and conditions, you may not install the licensing material or use it in any way. This licensing material remains the property of FEIG ELECTRONIC GmbH and is protected by international copyright.

§1 Object and Scope of the Agreement

1. FEIG grants you the right to install the licensing material provided and to use it under the following conditions.
2. You may install all components of the licensing material on a hard disk or other storage medium. The installation and use may also be done on a network fileserver. You may create backup copies of the licensing material.
3. FEIG grants you the right to use the documented program library ID FEFU for developing your own application programs or program libraries, and you may sell the runtime file FEFU.DLL, FEFUCE.DLL or LIBFEFU.SO.x.y.z[1] without licensing fees under the stipulation that these application programs or program libraries are used to control devices and/or systems which are developed and/or sold by FEIG.

§2. Protection of the Licensing Material

1. The licensing material is the intellectual property of FEIG and its suppliers. It is protected in accordance with copyright, international agreements and relevant national statutes where it is used. The structure, organization and code of the software are a valuable business secret and confidential information of FEIG and its suppliers.
2. You agree not to change, modify, translate, reverse develop, decompile, disassemble the program library or the documentation or in any way attempt to discover the source code of this software.
3. To the extent that FEIG has applied protection marks, such as copyright marks and other legal restrictions in the licensing material, you agree to keep these unchanged and to use them unchanged in all complete or partial copies which you make.
4. The transmission of licensing material in part or in full is prohibited unless there is an explicit agreement to the contrary between you and FEIG. Application programs or program libraries which are created and sold in accordance with §1 Par. 3 of this Agreement are excepted.

§3 Warranty and Liability Limitations

1. You agree with FEIG that it is not possible to develop EDP programs such that they are error-free for all application conditions. FEIG explicitly makes you aware that the installation of a new program can affect already existing software, including such software that does not run at the same time as the new software. FEIG assumes no liability for direct or indirect damages, for consequential damages or special damages, including lost profits or lost savings. If you want to ensure that no already installed program will be affected, you should not install the present software.
2. FEIG explicitly notes that this software makes it possible for irreversible settings and adaptations to be made on devices which could destroy these devices or render them unusable. FEIG assumes no liability for such actions, regardless of whether they are carried out intentionally or unintentionally.
3. FEIG provides the software „as is“ and without any warranty. FEIG cannot guarantee the performance or the results you obtain from using the software. FEIG assumes no liability or guarantee that the protection rights of third parties are not violated, nor that the software is suitable for a particular purpose.
4. FEIG calls explicit attention to the fact that the licensed material is not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to human health.
   To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures.

[1] *x.y.z represents the actual version number*

§4 Concluding Provisions

1. This Agreement contains the complete licensing terms and conditions and supercedes any prior agreements and terms. Changes and additions must be made in writing.
2. If any provision this agreement is declared to be void, or if for any reason is declared to be invalid or of no effect, the remaining provisions shall be in no manner affected thereby but shall remain in full force and effect. Both parties agree to replace the invalid provision with one which comes closest to its original intention.
3. This agreement is subject to the laws of the Federal Republic of Germany. Place of jurisdiction is Weilburg.

Content

1. Introduction

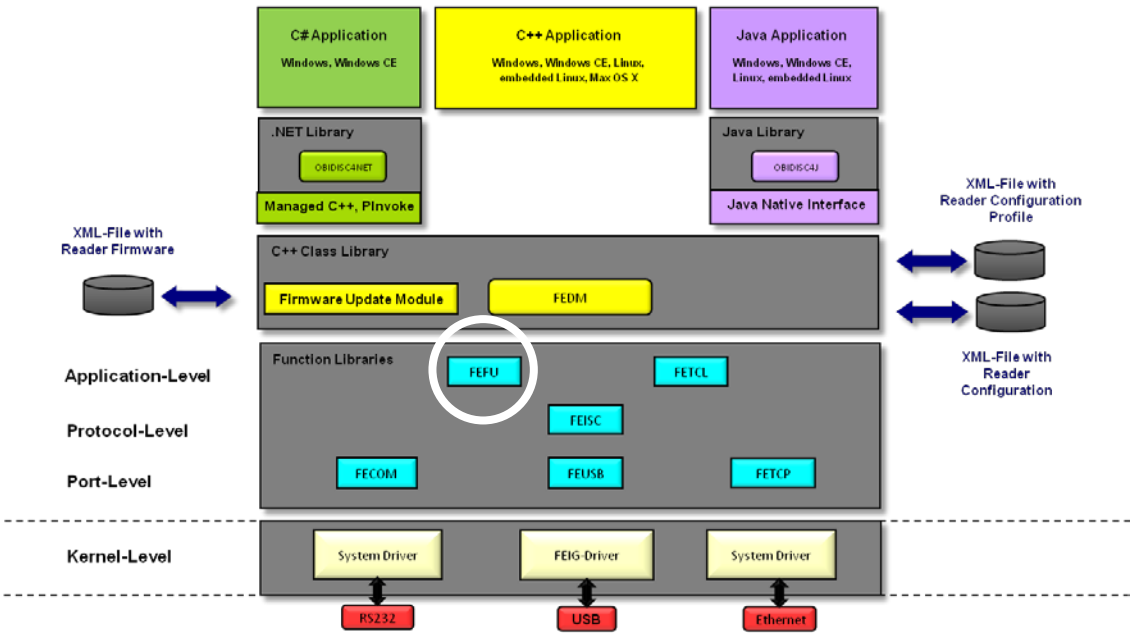
The support package ID FEFU is intended to assist in programming application software which integrates FEIG readers and external Function Units like ID ISC.ANT.MUX external multiplexers. It supports ANSI-C, ANSI-C++ languages and in principle any other language that can invoke C functions.

The support package offers a simple function interface to the FEIG function units. There is a separate function for each protocol documented in the system manual for the FEIG function unit. These commands are sent to the multiplexer through the FEIG reader. The ID FEISC support package is required for communication with the FEIG reader.

This library package can be used with the following Operating Systems:

Operating System	Target		Notes
	32-Bit	64-Bit	
Windows Vista / 7 / 8 / 10	X	X	
Windows CE 6	X	-	On request
Linux	X	X	
Android	X	X	On request
Apple Max OS X	-	X	On request

The library FEFU is part of the third level of a hierarchical structured, multi-tier FEIG library stack. It is only designed for managing Function Units and executing commands with them. The following picture shows the multi-tier library stack.



1.1. Shipment

This support package consists of files listed in the tables below. Normally, this package is shipped together with other libraries in a Software Development Kit (SDK) – e.g. ID ISC.SDK.Win.

1.1.1. Windows Vista / 7 / 8 / 10

File	Use
FEFU.DLL	DLL with all functions
FEFU.LIB	LIB file for linking for C/C++ projects
FEFU.H	Header file for C/C++ projects

1.1.2. Windows CE

File	**Use**
FEFUCE.DLL	DLL with all functions
FEFUCE.LIB	LIB file for linking with C/C++ projects
FEFU.H	Header file for C/C++ projects

1.1.3. Linux

File	Use
LIBFEFU.SO.x.y.z[2]	Function library
FEFU.H	Header file for C/C++ projects

[2] *x.y.z represents the actual version number*

2. Changes since the Previous Version

- New Functions:
  - **FEFU_DAT_SetRepeatParam**
  - **FEFU_MUX_SetRepeatParam**
  - **FEFU_UMUX_SetRepeatParam**

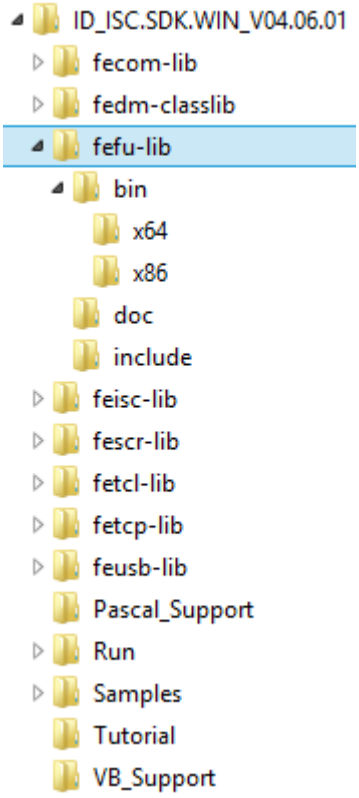
Please also note the revision history in the appendix to this document.

3. Installation

Normally, this package is shipped together with other libraries in a Software Development Kit (SDK). Copy the SDK into a directory of your choice.

The files of this library package can be found in the sub-directory fefu-lib.

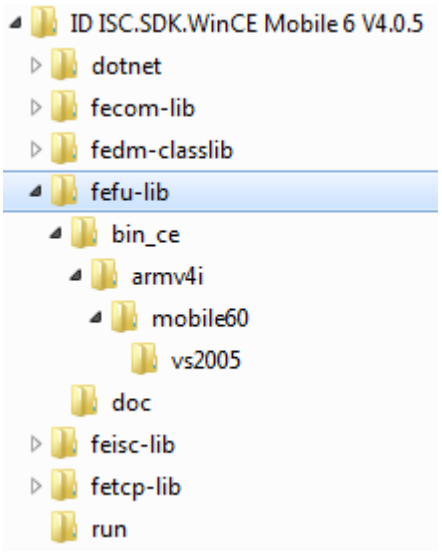
3.1. 32- and 64-Bit Windows Vista / 7 / 8 / 10



If you won't add your projects to the Samples path, we recommend the following steps:

- Copy FEFU.DLL into the directory of the application program (recommended) or into the Windows system directory.
- Copy FEFU.LIB into the project or LIB directory.
- Copy FEFU.H into the project or INCLUDE directory.

3.2. Windows CE 6

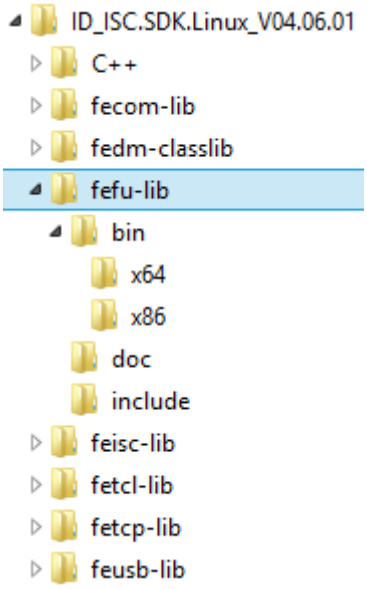


If you won't add your projects to the Samples path, we recommend the following steps:

- Copy FEFUCE.DLL into the application directory or system directory of the Windows CE system.
- Copy FEFUCE.LIB into the project or LIB directory.
- Copy FEFU.H into the project or INCLUDE directory

Note: you cannot use the DLL together with eMbedded Visual Basic 3.0.

3.3. 32- and 64-Bit Linux



Choose one option for installation:

Option 1: If an install.sh is shipped inside the SDK root directory, execute this install script. It will copy all library files into the directory /usr/lib resp. /usr/lib64 and creates symbolic links for each library file. The header file can be copied into a directory of your choice.

Option 2: Copy all files of this support package into a directory of your choice and create symbolic links for libfefu.so.x.y.z[3] in the directory /usr/lib resp. /usr/lib64 with the following calls:

cd /usr/lib (for 64 Bit : /usr/lib64)

ln –s /< your_directory>/libfefu.so.x.y.z libfefu.so.x

ln –s /< your_directory>/libfefu.so.x libfefu.so

ldconfig

The compiled library is linked against LibC V6 und LibStdC++ V6.

[3] *x.y.z represents the version number*

4. Incorporating into the Application Program

4.1. Supported Development Tools

Operating System	Development Tool	Supported
Windows Vista / 7 / 8 / 10	Visual Studio	Yes
	Borland C++ Builder	Yes
	Embarcadero C++ Builder	Yes
Windows CE 6	eMbedded Visual C++ 4	No
	Visual Studio 2005	No
	Visual Studio 2008	Yes
Linux	GCC	Yes
Mac OS X	GCC	Yes, for projects with x86_64 architecture
	Xcode	Yes, for projects with x86_64 architecture

4.2. Incorporating into Visual Studio

1. Add include path for the header file in project settings (category C/C++)
2. Add fefu.lib (optional with path) in project settings (category Linker)

ID FEISC and one of the communications libraries (FECOM, FEUSB, FETCP) must also be incorporated into your project.

5. Programming Interface

5.1. Overview

The FEFU-DLL incorporates for the programmer all the necessary functions for easy communication with external Function Units that are accessed by readers in the FEIG family. Fig. 1 shows the chain of communication within and outside the host.

Driving the Function Units requires the handle of a reader object from the FEISC library (see FEISC_NewReader) and knowledge of the bus address of the reader. The reader object within the FEISC in turn requires the handle of a port driver (e.g. from FECOM).

No objects are generated within the FEFU.DLL. The library is configured as a simple function collection.

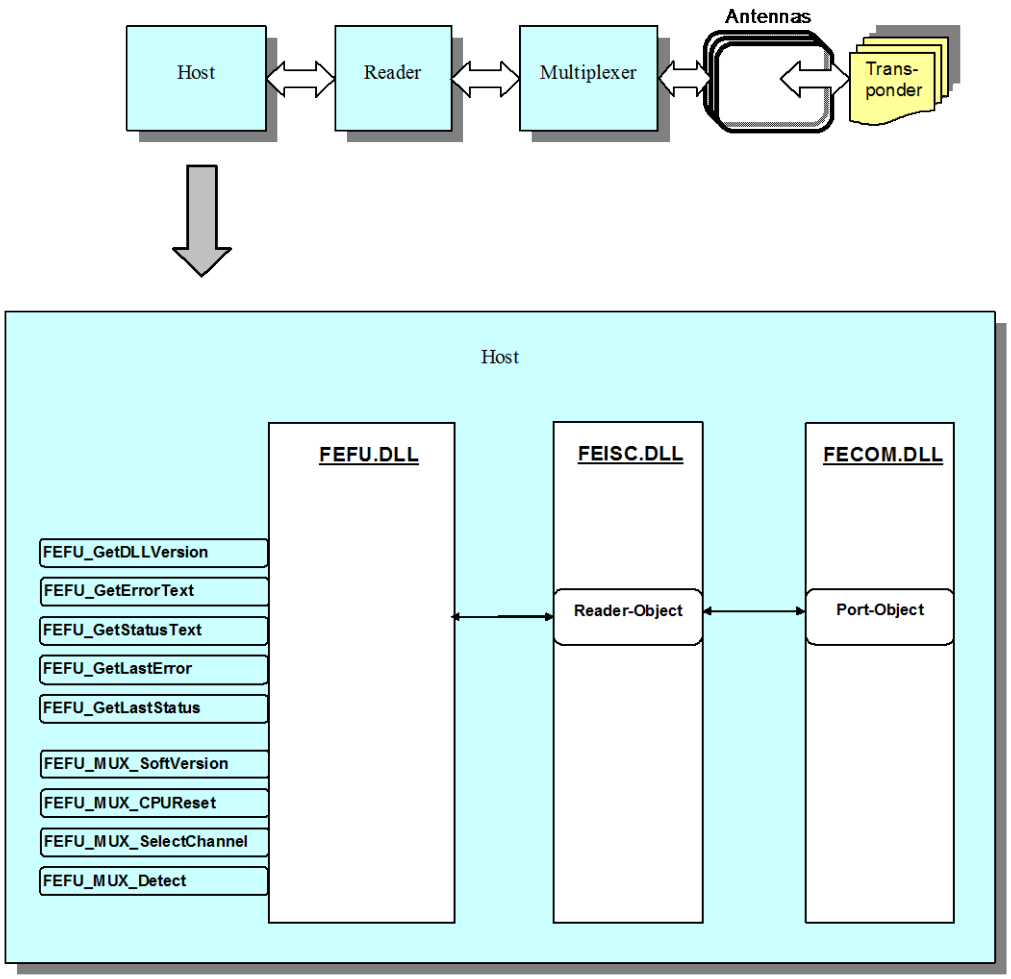


Fig. 1: Chain of communication for FEFU.DLL using the serial port as an example

5.2. Thread Security

In principle, all FEIG libraries are not fully thread safe. But respecting some guidance, a practical thread security can be realized allowing parallel execution of communication tasks. One should keep in mind, that all FEIG RFID readers work synchronously and can perform commands only in succession.

On the level of the transport layer (FECOM, FEUSB, FETCP) the communication with each port must be synchronized in the application, as the reader works synchronously. Using multiple ports and so multiple Readers from different threads simultaneously is possible, as the internal port objects acts independently from each other.

On the level of the protocol layer (FEISC), parallelism can be realized, when each reader object represents exactly one physical reader and is bound with an individual communication port. This is not true for the four specialized functions FEISC_BuildxxProtocol and FEISC_SplitxxProtocol, which use an internal global buffer for protocol data.

The library FEFU has no precautions for thread-safeness implemented. Thus, only one thread can call FEFU functions at the same time. Thread-safeness must be implemented on application side.

5.3. Functions

The support package contains all the functions for various tasks. These are divided into groups for a better overview.

General Functions

- **void FEFU_GetDLLVersion(** char* cVersion **)**
- **int FEFU_GetErrorText(** int iErrorCode, char* cErrorText **)**
- **int FEFU_GetStatusText(** UCHAR ucStatus, char* cStatusText **)**
- **int FEFU_GetLastState(** int iReaderHnd, char* cStatusText **)**
- **int FEFU_GetLastError(** int iReaderHnd , int* iErrorCode, char* cErrorText **)**

Special Communications Functions for HF Multiplexers

- **int FEFU_MUX_CPUReset (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr **)**
- **int FEFU_MUX_SoftVersion (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR* ucVersion **)**
- **int FEFU_MUX_SelectChannel (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucIn1, UCHAR ucIn2 **)**
- **int FEFU_MUX_Detect (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr**)**

Special Communications Functions for HF Dynamic Antenna Tuner

- **int FEFU_DAT_CPUReset (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr **)**
- **int FEFU_DAT_SoftVersion (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR* ucVersion **)**
- **int FEFU_DAT_Detect (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr**)**
- **int FEFU_DAT_GetValues (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR* ucValues**)**
- **int FEFU_DAT_ SetOutput(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucOut**)**
- **int FEFU_DAT_ StartTuning(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr**)**
- **int FEFU_DAT_ ReTuning(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr**)**
- **int FEFU_DAT_ StoreSettings(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr**)**
- **int FEFU_DAT_ SetAddress(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucNewDatAdr**)**
- **int FEFU_DAT_SetMode (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucMode**)**

Special Communications Functions for UHF Multiplexers

- **int FEFU_UMUX_CPUReset (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucFlags **)**
- **int FEFU_UMUX_SoftVersion (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucFlags, UCHAR* ucVersion **)**
- **int FEFU_UMUX_SelectChannel (** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucFlags, UCHAR ucChannelNo **)**
- **int FEFU_UMUX_Detect_GetPower(** int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucFlags, UCHAR* ucData **)**

5.3.1. FEFU_GetDLLVersion

Function	Query version number of the DLL
Syntax	**void FEFU_GetDLLVersion(char* cVersion)**
Description	The function returns the version number of the DLL. *cVersion* is an empty, null terminated string for returning the version number (e.g. “02.00.00”). The string should be capable of holding at least 256 characters.
Return value	None
Example	... #include "fefu.h" ... ... char cVersion[256]; FEFU_GetDLLVersion(cVersion); // Code for displaying the version number ... ...

5.3.1. FEFU_GetErrorText

Function	Get error text for the error code
Syntax	**int FEFU_GetErrorText(int iErrorCode, char* cErrorText)**
Description	The function passes the English error text associated with the *iErrorCode* in *cErrorText*. The buffer for *cErrorText* should be able to hold 256 characters.
Return value	If there is no error the function returns null and in case of error a value less than null. The list of error codes can be found in the appendix.
Example	... #include "fefu.h" ... ... char cErrorText[256]; ... int iBack = FEFU_GetErrorText(FEFU_ERR_TIMEOUT, cErrorText) // Code for displaying the text ... ...

5.3.2. FEFU_GetStatusText

Function	Gets short text for the status byte.
Syntax	**int FEFU_GetStatusText(UCHAR ucStatus, char* cStatusText)**
Description	The function passes the English short text associated with the *ucStatus* in *cStatusText*. The buffer for *cStatusText* should be able to hold 128 characters.
Return value	If there is no error the function returns null and in case of error a value less than null. The list of error codes can be found in the appendix.
Example	... #include "fefu.h" ... ... char cStatusText[128]; ... int iBack = FEFU_GetStatusText(0x11, cStatusText) // Code for displaying the text ... ...

5.3.3. FEFU_GetLastState

Function	Returns the status byte contained in the last receive protocol.
Syntax	**int FEFU_GetLastStatus(char* cStatusText)**
Description	This function can be used to get the status byte and a short text for the status byte of the last receive protocol. The buffer for the short text *cStateText* should be able to hold 256 characters.
Return value	If there is no error the return value contains the status byte. The list of error codes can be found in the appendix.

5.3.4. FEFU_GetLastError

Function	Gets the last error code and passes error text.
Syntax	**int FEFU_GetLastError(int* iErrorCode, char* cErrorText)**
Description	The function uses *iErrorCode* to return the last error code and the associated English error text in *cErrorText*. The buffer for *cErrorText* should be able to hold 256 characters.
Return value	If there is no error the function returns null and in case of error a value less than null. The list of error codes can be found in the appendix.

Example	...
	#include "fefu.h"
	...
	...
	char cErrorText[256];
	int iErrorCode = 0;
	...
	
	int iBack = FEFU_GetLastError(&iErrorCode, cErrorText)
	// Code for displaying the text
	...
	...

5.3.5. FEFU_MUX_SetRepeatParam

Function	Setting of MUX command repetition parameter.
Syntax	**void FEFU_MUX_SetRepeatParam(int iMaxRepeat, int iDelay)**
Parameter	iMaxRepeat is the maximum count of all MUX-command repetitions. Valid values range from 0 to 10. Default value is 2. *iDelay* is the delay before each command repetition in milliseconds. Default value is 100.
Return value	None.

5.3.6. FEFU_DAT_SetRepeatParam

Function	Setting of DAT command repetition parameter.
Syntax	**void FEFU_DAT_SetRepeatParam(int iMaxRepeat, int iDelay)**
Parameter	iMaxRepeat is the maximum count of all DAT-command repetitions. Valid values range from 0 to 10. Default value is 2. *iDelay* is the delay before each command repetition in milliseconds. Default value is 100.
Return value	None.

5.3.7. FEFU_UMUX_SetRepeatParam

Function	Setting of UMUX command repetition parameter.
Syntax	**void FEFU_UMUX_SetRepeatParam(int iMaxRepeat, int iDelay)**
Parameter	iMaxRepeat is the maximum count of all UMUX-command repetitions. Valid values range from 0 to 10. Default value is 2. *iDelay* is the delay before each command repetition in milliseconds. Default value is 100.
Return value	None.

5.3.8. FEFU_MUX_SoftVersion

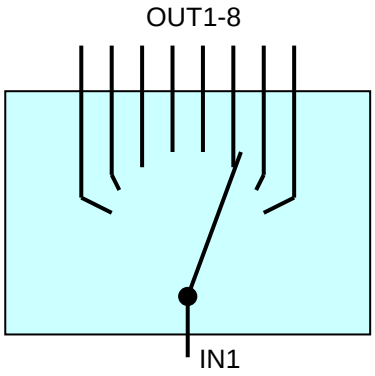
Function	Reads the version number of the multiplexer.
Syntax	**int FEFU_MUX_SoftVersion(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR* ucVersion)**
Description	Gets the version number of the multiplexer and stores it in uc*Version* The buffer for the version must be able to hold at least 8 bytes. 1 byte is provided for the NUL character. *ucMuxAdr* is the DIP switch settable cascading level. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Cross-reference	The structure of ucVersion is documented in the system manual H30701-xx-ID-B for the multiplexer.
Return value	If there is no error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

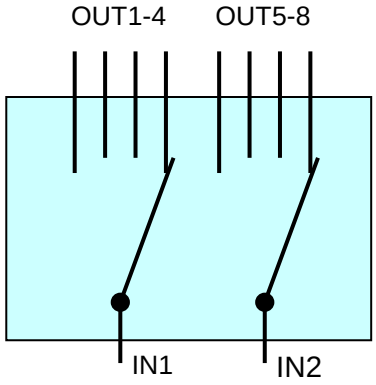
5.3.9. FEFU_MUX_CPUReset

Function	Triggers a reset in the CPU of the multiplexer.
Syntax	**int FEFU_MUX_CPUReset(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr)**
Description	Triggers a reset in the CPU of the multiplexer. *ucMuxAdr* is the DIP switch settable cascading level. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is no error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.10. FEFU_MUX_SelectChannel

Function	Sets multiplexer outputs.
Syntax	**int FEFU_MUX_SelectChannel(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucIn1, UCHAR ucIn2)**





Description	The function switches a multiplexer output for each input ucIn1 and ucIn2. In ucIn1 (2) you pass the number of the output connected to Input 1 (2). If Input 2 is not needed, set its value ucIn2 to 0.. *uucMuxAdr* is the DIP switch settable cascading level. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.11. FEFU_MUX_Detect

Function	Detects a multiplexer.
Syntax	**int FEFU_MUX_Detect(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr)**
Description	Detects a multiplexer. *ucMuxAdr* is the cascading level in which you want to detect. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.12. FEFU_DAT_SoftVersion

Function	Reads the version number of the dynamic antenna tuner.
Syntax	**int FEFU_MUX_SoftVersion(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR* ucVersion)**
Description	Gets the version number of the dynamic antenna tuner and stores it in uc*Version* The buffer for the version must be able to hold at least 8 bytes. 1 byte is provided for the NUL character. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Cross-reference	The structure of ucVersion is documented in the system manual H30701-xx-ID-B for the dynamic antenna tuner.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.13. FEFU_DAT_CPUReset

Function	Triggers a reset in the CPU of the dynamic antenna tuner.
Syntax	**int FEFU_DAT_CPUReset(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr)**
Description	Triggers a reset in the CPU of the dynamic antenna tuner. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.14. FEFU_DAT_Detect

Function	Detects a dynamic antenna tuner.
Syntax	**int FEFU_DAT_Detect(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr)**
Description	Detects a dynamic antenna tuner. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.15. FEFU_DAT_GetValues

Function	Reads tuning values from a dynamic antenna tuner.
Syntax	**int FEFU_DAT_GetValues(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR* ucValues)**
Description	Reads tuning values from a dynamic antenna tuner and stores it in ucValues. The buffer for the tuning values must be able to hold at least 7 bytes. 1 byte is provided for the NUL character. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Cross-reference	The structure of ucValues is documented in the system manual H30701-xx-ID-B for the dynamic antenna tuner.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.16. FEFU_DAT_SetOutput

Function	Set of outputs in a dynamic antenna tuner.
Syntax	**int FEFU_DAT_SetOutput(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucOut)**
Description	Set of outputs in a dynamic antenna tuner. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.17. FEFU_DAT_StartTuning

Function	Starts the tuning process in a dynamic antenna tuner.
Syntax	**int FEFU_DAT_StartTuning(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr)**
Description	The function starts the tuning process in a dynamic antenna tuner. The tuner responds first and afterwards the tuning process is startet. This tuning process needs time, depending on the installation situation. During this time, the tuner is not accessible. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.18. FEFU_DAT_ReTuning

Function	Starts a re-tuning in a dynamic antenna tuner.
Syntax	**int FEFU_DAT_ReTuning(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr)**
Description	Starts a re-tuning in a dynamic antenna tuner. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.19. FEFU_DAT_StoreSettings

Function	Saves the tuning values.
Syntax	**int FEFU_DAT_StoreSettings(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr)**
Description	The function saves the tuning values of a dynamic antenna tuner in its EEPROM. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.20. FEFU_DAT_SetAddress

Function	Sets the address in a dynamic antenna tuner.
Syntax	**int FEFU_DAT_SetAddress(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucNewDatAdr)**
Description	Sets the address in a dynamic antenna tuner. *ucNewDatAdr* is the new tuner address. *ucDatAdr* is the actual tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.21. FEFU_DAT_SetMode

Function	Sets a new mode in a dynamic antenna tuner.
Syntax	**int FEFU_DAT_SetMode(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucDatAdr, UCHAR ucMode)**
Description	Sets a new mode in a dynamic antenna tuner. *ucMode* is the mode byte. *ucDatAdr* is the tuner address. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.22. FEFU_UMUX_SoftVersion

Function	Reads the version number of the UHF multiplexer.
Syntax	**int FEFU_UMUX_SoftVersion(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR* ucVersion)**
Description	Gets the version number of the UHF multiplexer and stores it in uc*Version* The buffer for the version must be able to hold at least 8 bytes. 1 byte is provided for the NUL character. *ucMuxAdr* is the DIP switch settable cascading level. *ucFlags* contains control flags. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Cross-reference	The structure of ucVersion is documented in the system manual H80302-xx-ID-B for the UHF multiplexer.
Return value	If there is no error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.23. FEFU_UMUX_CPUReset

Function	Triggers a reset in the CPU of the UHF multiplexer.
Syntax	**int FEFU_UMUX_CPUReset(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr)**
Description	Triggers a reset in the CPU of the UHF multiplexer. *ucMuxAdr* is the DIP switch settable cascading level. *ucFlags* contains control flags. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is no error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.24. FEFU_UMUX_SelectChannel

Function	Sets UHF multiplexer outputs.
Syntax	**int FEFU_UMUX_SelectChannel(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR ucChannelNo)**
Description	The function switches the input to the output *ucChannelNo*. *uucMuxAdr* is the DIP switch settable cascading level. *ucFlags* contains control flags. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

5.3.25. FEFU_UMUX_Detect_GetPower

Function	Detects a UHF multiplexer and requests actual RF power values.
Syntax	**int FEFU_UMUX_Detect_GetPower(int iReaderHnd, UCHAR ucReaderBusAdr, UCHAR ucMuxAdr, UCHAR* ucData)**
Description	Detects a UHF multiplexer and requests actual RF power values from the selected output channel. The power values are stored in *ucData*. *ucMuxAdr* is the cascading level in which you want to detect. *ucFlags* contains control flags. *iReaderHnd* is the handle for the reader object in the FEISC library. *ucReaderBusAdr* is the bus address set in the reader.
Cross-reference	The structure of ucData is documented in the system manual H80302-xx-ID-B for the UHF multiplexer.
Return value	If there is not error the return value contains the status byte of the reply protocol. The list of error codes can be found in the appendix.

APPENDIX

5.4. Error Codes

Error constant	Value	Description
FEFU_ERR_POINTER_IS_NULL	-4102	Pointer to passing parameter is null
FEFU_ERR_NO_MORE_MEM	-4103	No more system memory
FEFU_ERR_UNSUPPORTED_FUNCTION	-4105	Not a supported function
FEFU_ERR_PROTLEN	-4130	Protocol length error
FEFU_ERR_CHECKSUM	-4131	Checksum error
FEFU_ERR_TIMEOUT	-4132	Communication timeout
FEFU_ERR_UNKNOWN_STATUS	-4133	Unknown status byte
FEFU_ERR_NO_RECDATA	-4134	No data received
FEFU_ERR_UNKNOWN_PARAMETER	-4150	Unknown passing parameter
FEFU_ERR_PARAMETER_OUT_OF_RANGE	-4151	Passing parameter too large or too small
FEFU_ERR_UNKNOWN_ERRORCODE	-4153	Unknown error code

5.5. Revision History

V2.01.00

- Windows:

  Migration to Visual Studio 2012.

  DLL without MFC

  First release of 64-Bit version

  Discontinued support for Windows XP

- Windows CE

  Discontinued support for Windows CE 4 and CE 5

- Linux:

  Version for 64-Bit

- Functions without modifications

V2.00.00

- Windows / Windows CE: Migration of the development environment from Visual Studio 6 to Visual Studio 2008.
- Functions without modifications

V1.03.00

- New functions for UHF multiplexer ID ISC.UMUX.

V1.02.00

- Modifications for the timing of the function FEFU_DAT_GetValues.
- Modified licence agreement
- The library is compiled under SuSE Linux 9.1 with the GNU Compiler Collection V3.3.3

V1.01.00

- Support for Dynamic Antenna Tuner ID ISC.DAT
- First Linux release (SuSE Linux 8.2, GNU Compiler Collection V3.3-23, glibc V2.3.2-6)

V1.00.00

- This is the first version.