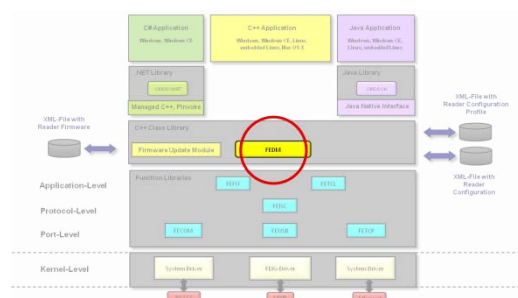


C++ Class Library ID FEDM

Version 4.05.00

Part B.ISC

Software-Support for OBID i-scan® and OBID® classic-pro



Operating System	Target		Notes
	32-Bit	64-Bit	
Windows XP	X	(X)	with 64-Bit OS: only with 32-Bit Runtime Environment
Windows Vista / 7 / 8	X	X	
Windows CE	X	-	
Linux	X	(X)	with 64-Bit OS: only with 32-Bit Runtime Environment
Apple Max OS X	-	X	OS X V10.7.3 or higher Architecture x86_64

Note

© Copyright 2001-2013 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
eMail: obid-support@feig.de

This manual supercedes all previous editions.

The information contained in this manual is subject to change without notice.

Copying of this document, and giving it to others and the use or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design.

The information contained in this manual has been gathered with all due care and to the best of our knowledge. FEIG ELECTRONIC GmbH assumes no liability for the accuracy and completeness of the data in this manual. In particular, FEIG ELECTRONIC GmbH cannot be held liable for consequential damages resulting from inaccurate or incomplete information. Since even with our best efforts this document may still contain mistakes, please contact us should you find any errors.

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.

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 OBID®-system in cross surroundings.

OBID® and OBID i-scan® are registered trademarks of FEIG ELECTRONIC GmbH.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Windows Vista is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries

Linux® is a registered Trademark of Linus Torvalds.

Apple, Mac, Mac OS, OS X, Cocoa and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries.

Electronic Product Code (TM) is a Trademark of EPCglobal Inc.

I-CODE® and Mifare® are registered Trademarks of Philips Electronics N.V.

Tag-it (TM) is a registered Trademark of Texas Instruments Inc.

Jewel (TM) is a trademark of Innovision Research & Technology plc.

Licensing agreement for use of the software

This is an agreement between you and FEIG ELECTRONIC GmbH (hereafter "FEIG") for use of the ID FEDM C++ Class 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 for developing your own application programs or program libraries, and you may sell the runtime files `FedmlscCoreVCxx.dll`¹, `FedmlscCoreCE.dll`, `libFedmlscCore.x.y.z.dylib`² and `libFedmlscCore.so.x.y.z`² 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 call explicit attention 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 a human.
To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures.

¹ xx represents the version number of the dependent MFC library

² 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.
4. This agreement is subject to the laws of the Federal Republic of Germany. Place of jurisdiction is Frankfurt a. M.

Contents

Licensing agreement for use of the software.....	3
Contents	5
Notes concerning the documentation for this library.....	7
1. Overview	8
2. Revisions since the previous version.....	10
3. Installation.....	11
3.1. 32- and 64-Bit Windows XP/Vista/7/8	11
3.2. Windows CE.....	13
3.3. 32- and 64-Bit Linux	14
3.4. 64-Bit Mac OS X.....	15
3.5. Source Code	16
3.6. Dependencies	19
4. Integration into application projects	20
4.1. Supported Development Tools.....	20
4.2. The quick way.....	20
4.2.1. Incorporating into Visual Studio.....	20
4.2.2. Incorporating into Xcode	24
4.3. Manual integration.....	25
4.3.1. Include files	25
4.3.2. Source code files	25
4.3.3. Required project settings	26
4.3.4. Windows	26
4.3.5. Linux	26
4.3.6. Optional project settings.....	26
5. Installation on the target computer	27
5.1. 32-Bit Libraries on a 32- and 64-Bit Windows	27
5.2. 64-Bit Libraries on a 64-Bit Windows.....	29
5.3. 32- and 64-Bit Linux	30
5.4. 64-Bit Mac OS X.....	30
6. Class description	31
6.1. FEDM_ISCReader	31

6.1.1. Implemented data containers	31
6.1.2. Implemented tables.....	31
6.1.3. Methods (public)	32
6.2. FEDM_ISCReaderModule.....	35
6.2.1. Methods (public)	35
6.3. Concept of TagHandler classes	38
6.4. Working with the Reader classes.....	39
6.4.1. Important initializations.....	39
6.4.2. Management of the reader configuration.....	41
6.4.3. Examples for using the method SendProtocol.....	44
6.4.4. Asynchronous tasks for relieving the load on applications.....	53
6.4.5. Serializing in XML-Format	57
6.5. Table for ISO Host Commands.....	59
6.5.1. Anomaly of the addressed mode.....	60
6.5.2. Examples for using the table with [0xB0] Commands.....	61
6.5.3. Examples for using the table with [0xB3] Commands.....	70
6.6. Table for Buffered Read Mode.....	72
6.6.1. Examples for using of the table	72
6.7. FEDM_ISCFunctionUnit	75
6.7.1. Constructor	76
6.7.2. Implemented data containers	76
6.7.3. Implemented lists	76
6.7.4. Methods (public)	76
6.7.5. Important initializations.....	77
6.7.6. Examples for using the method SendProtocol.....	78
6.8. FedmIscPeopleCounter	80
6.8.1. Methods (public)	80
6.8.2. Example for using theclass	80
6.8.3. Example for automatic notification.....	81
7. Appendix	83
7.1. Supported OBID® Readers.....	83
7.2. Supported Transponders.....	84
7.3. TCP-Status.....	85
7.4. List of constants.....	86
7.4.1. Internal Constants.....	86
7.4.2. General Constants	86
7.4.3. Constants for uiTableID	86
7.4.4. Constants for uiDataID.....	88
7.5. Revision history	95

Notes concerning the documentation for this library

This manual describes a software library which is also available as annotated source code. For this reason we have limited the documentation to what is absolutely necessary for understanding the functionality and use of the classes. It is assumed that the user of this library reads the source code and becomes familiar with the details using this document, the header files and the included comments.

To understand the internal program sequences you will also have to refer to the system manuals for whichever OBID® readers and function libraries you are using.

FEIG ELECTRONIC GmbH does not repeat the same information about OBID® readers in different manuals or use cross-references to certain pages in a different document. This is necessary due to the constant updating of manuals, and it prevents confusion caused by information in out-of-date documents. The user of this library is therefore well advised to check regularly that he has the most current manuals. If not, these can of course be obtained whenever needed from FEIG ELECTRONIC GmbH.

Important notes:

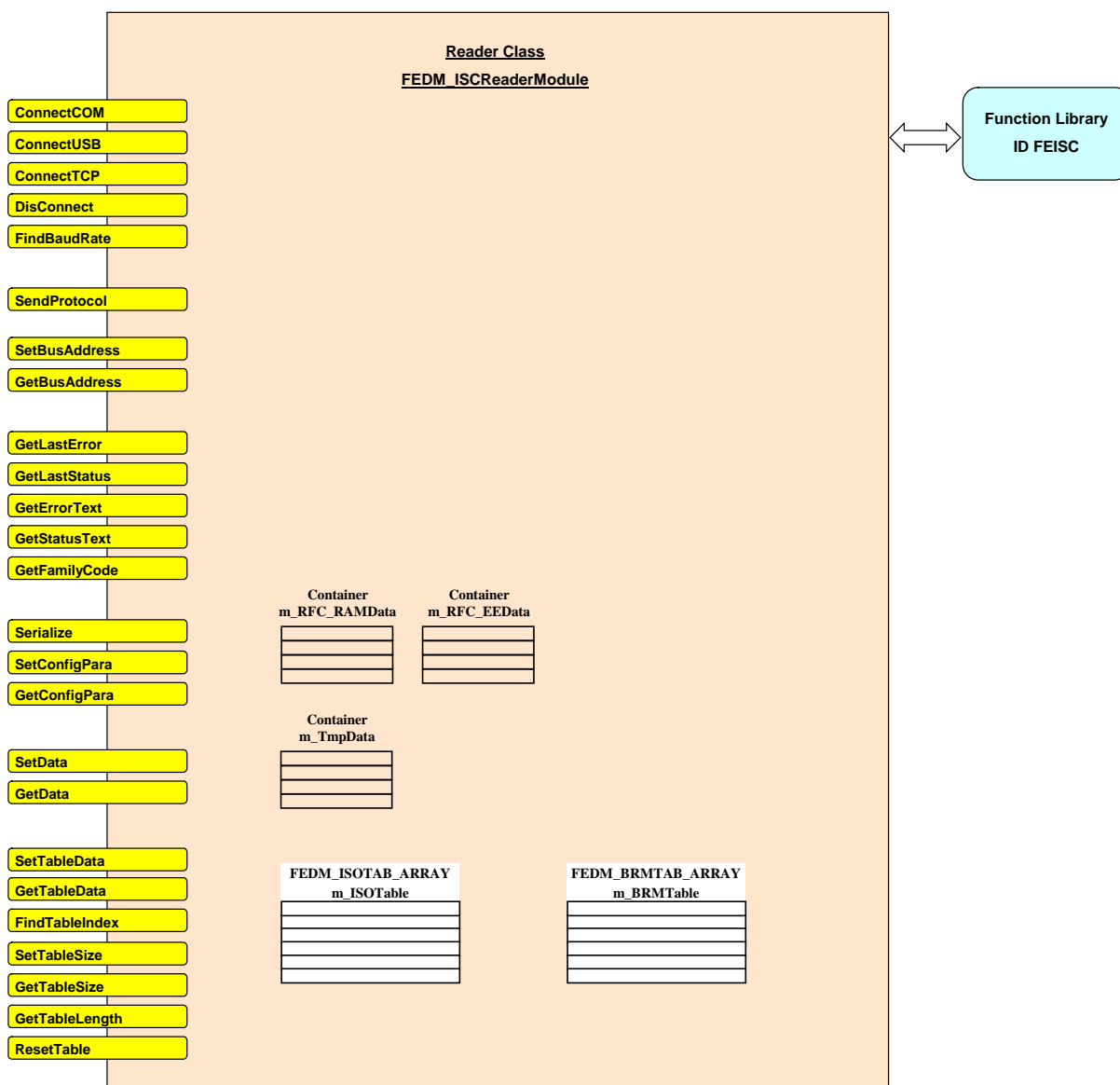
You are only permitted to use this library if you have first agreed to the license conditions on the back of this page.

Anyone is free to modify source code. Therefore you should work only with libraries you have received directly from FEIG ELECTRONIC GmbH. In any case further transmission of the source code is prohibited.

1.Overview

Support for OBID i-scan® Readers consists of the FEDM_ISCReader derived from the base class FEDM_DataBase and two tables integrated into this reader class which in turn is stored as an array of type FEDM_ISOTabItem resp. FEDM_BRMTabItem classes. This documentation is the second part of the documentation for the ID FEDM Class Library, the concept and base classes of which are described in document number H10102-xe-ID-B.

The component diagram shows an overview of all the methods and data containers as well as the tables.



The reader class supports all OBID *i-scan*® and OBID® *classic-pro* Readers, which means that not all options in the class are usable with all reader models.

The library is supplemented with classes for external function units: FEDM_ISCFunctionUnit, FedmIsExternalIO and FedmIsPeopleCounter.

For efficient programming of Transponder communication a concept named *TagHandler classes* is provided. This library part consists of a large pool of proxy classes with an easy API for each supported transponder type (e.g. EPC Class 1 Gen 2).

FedmIsTagHandler_EPC_Class1_Gen2
+BANK_RESERVED : unsigned int +BANK_EPC : unsigned int +BANK_TID : unsigned int +BANK_USER : unsigned int +UNCHANGED : unsigned int +UNLOCK : unsigned int +UNLOCK_PERMANENTLY : unsigned int +LOCK : unsigned int +LOCK_PERMANENTLY : unsigned int
+FedmIsTagHandler_EPC_Class1_Gen2(uiTabIndex : unsigned int, uiTagHandlerType : unsigned int, pTabItem : FEDM_ISOTabItem *) +FedmIsTagHandler_EPC_Class1_Gen2() +ReadMultipleBlocks(uiBank : unsigned int, uiFirstDataBlock : unsigned int, uiNoOfDataBlocks : unsigned int, sPassword : string, pucData : unsigned char *) : int +WriteMultipleBlocks(uiBank : unsigned int, uiFirstDataBlock : unsigned int, uiNoOfDataBlock : unsigned int, sPassword : string, pucData : unsigned char *) : int +WriteEPC(sNewEPC : string, sPassword : string = "") : int +Kill(sPassword : string) : int +Lock(sPassword : string, uiKillSetting : unsigned int, uiAccessPasswordSetting : unsigned int, uiEPCMemorySetting : unsigned int, uiTIDMemorySetting : unsigned int, uiUserMemorySetting : unsigned int) : int

Example: TagHandler class for EPC Class 1 Gen 2

2. Revisions since the previous version

- Support for new Reader: **ID CPR47.xx**
- Update of namespaces and access constants for reader configuration
- New namespace **OBID::Fedm::Core::i_scan::ReaderCommand** containing all command parameters in a structured manner.
- Class **FEDM_ISCReader**: new methods SetCommandPara and GetCommandPara.
- Support for **ISO 18000-3M3** Transponder.
- Support for new ISO 15693-Transponders: **STM M24LRxxE-R** and **STM LRIS64K**
- **New TagHandler classes** for **ISO 18000-3M3**, **STM M24LRxxE-R** and **STM LRIS64K**
- **TagHandler class for NXP ICode SLI-L**: new method PasswordProtectAFI
- **TagHandler class for EPC Class 1 Gen 2**:
 1. Support for Recommissioning-Bits in the Kill method.
 2. Support for Extended PC
 3. Bugfix for EPCs with 8 Byte length.
 4. Bugfix in WriteEPC: Return of an error code instead of the EPC length
- Beta-Version of a new and simplified API, realized with the class **ReaderModule** in the namespace **OBID::Fedm::Core::i_scan**. Documentation on request.

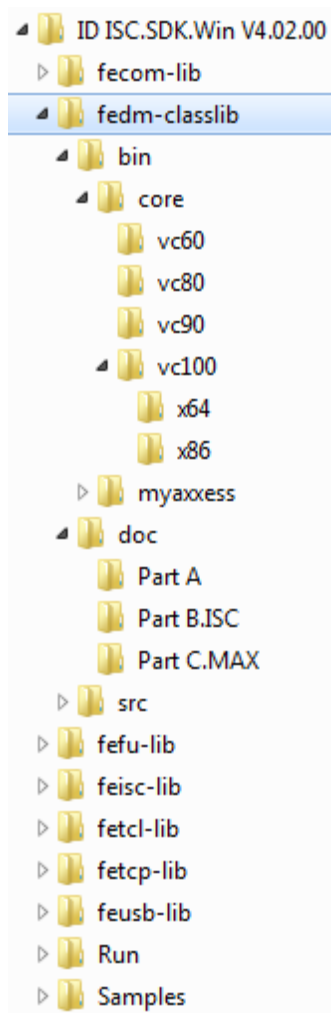
Please note also the revision history in the Appendix to this document.

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 fedm-classlib.

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



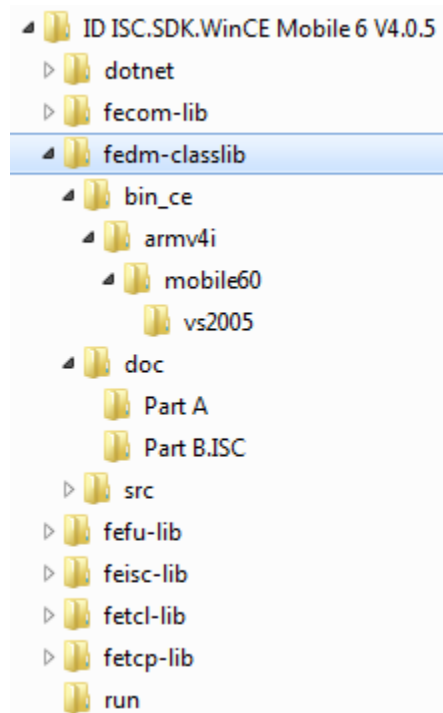
If you won't add your projects to the Samples path, we recommend the following steps:

- Copy all required DLL files from the Run directory into the directory of the application program.
- Use all required LIB files from the SDK directories (recommended) or copy all required LIB files in the project or LIB directory.
- Use all required Include Directories from the SDK directory (recommended) or copy all required Header files into the project or INCLUDE path. Note: the directory struktur in src must be maintained.
- As you can see, the library myAXXESS for access control applications is installed too. The manual for this library part can be found in Part C.MAX (H90080-xe-ID-B.DOC).

After the installation the directory fedm-classlib contains the following files:

Files in sub-directory src	Description
FedmlscCore.h	contains all includes and preprocessor definitions
Files in sub-directory bin\core\vcxx\x86	Description
FedmlscCoreVC60.dll/.lib	with Visual Studio 6 compiled 32-Bit library (release version), compatible with MFC-Version 6.0
FedmlscCoreVC60d.dll/.lib	with Visual Studio 6 compiled 32-Bit library (debug version), compatible with MFC-Version 6.0
FedmlscCoreVC80.dll/.lib	with Visual Studio 2005 compiled 32-Bit library (release version), compatible with MFC-Version 8.0
FedmlscCoreVC80d.dll/.lib	with Visual Studio 2005 compiled 32-Bit library (debug version), compatible with MFC-Version 8.0
FedmlscCoreVC90.dll/.lib	with Visual Studio 2008 compiled 32-Bit library (release version), compatible with MFC-Version 9.0
FedmlscCoreVC90d.dll/.lib	with Visual Studio 2008 compiled 32-Bit library (debug version), compatible with MFC-Version 9.0
FedmlscCoreVC100.dll/.lib	with Visual Studio 2010 compiled 32-Bit library (release version), compatible with MFC-Version 10.0
FedmlscCoreVC100d.dll/.lib	with Visual Studio 2010 compiled 32-Bit library (debug version), compatible with MFC-Version 10.0
FedmlscCoreVC110.dll/.lib	with Visual Studio 2012 compiled 32-Bit library (release version), compatible with MFC-Version 11.0
FedmlscCoreVC110d.dll/.lib	with Visual Studio 2012 compiled 32-Bit library (debug version), compatible with MFC-Version 11.0
Files in sub-directory bin\core\vcxx\x64	Description
FedmlscCoreVC100.dll/.lib	with Visual Studio 2010 compiled 64-Bit library (release version), compatible with MFC-Version 10.0
FedmlscCoreVC100d.dll/.lib	with Visual Studio 2010 compiled 64-Bit library (debug version), compatible with MFC-Version 10.0
FedmlscCoreVC110.dll/.lib	with Visual Studio 2012 compiled 64-Bit library (release version), compatible with MFC-Version 11.0
FedmlscCoreVC110d.dll/.lib	with Visual Studio 2012 compiled 64-Bit library (debug version), compatible with MFC-Version 11.0

3.2. Windows CE



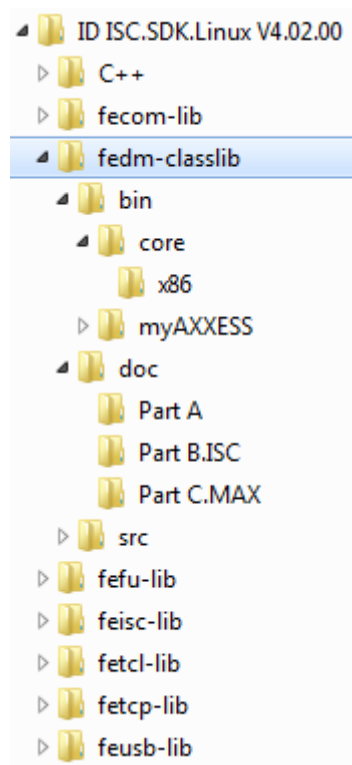
If you won't add your projects to the Samples path, we recommend the following steps:

- Copy all required DLL files from the Run directory into the directory of the application program.
- Use all required LIB files from the SDK directories (recommended) or copy all required LIB files in the project or LIB directory.
- Use all required Include Directories from the SDK directory (recommended) or copy all required Header files into the project or INCLUDE path. Note: the directory struktur in src must be maintained.

After the installation the directory fedm-classlib contains the following files:

Files in sub-directory src	Description
FedmlscCore.h	contains all includes and preprocessor definitions
Files in sub-directory bin_ce\...	Description
FedmlscCoreCE.dll/.lib	with Visual Studio compiled library, compatible with the Windows CE platform

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 (./install.sh). It will copy all library files into the directory /usr/lib and creates symbolic links for each library file.

Option 2: Copy all files of this support package into a directory of your choice and create symbolic links for libFedmlscCore.so.x.y.z¹ in the directory /usr/lib with the following calls:

```
cd /usr/lib
```

```
ln -s /<your_directory>/libFedmlscCore.so.x.y.z libFedmlscCore.so.x
```

```
ln -s /<your_directory>/libFedmlscCore.so.x libFedmlscCore.so
```

```
ldconfig
```

After the installation the directory fedm-classlib contains the following files:

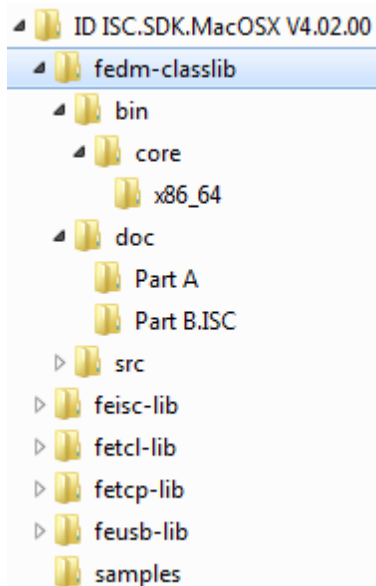
Files in sub-directory src	Description
FedmlscCore.h	contains all includes and preprocessor definitions
Files in sub-directory bin/core	Description
libFedmlscCore.so.x.y.z	with GCC compiled library for Linux

x.y.z represents the version number of the library

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

¹ x.y.z represents the version number

3.4. 64-Bit Mac OS X



Choose one option for installation:

Option 1: If an install.sh is shipped inside the SDK root directory, execute this install script (./install.sh). It will copy all library files into the directory /usr/lib and creates symbolic links for each library file.

Option 2: Copy all files of this support package into a directory of your choice and create symbolic links for libFedmlscCore.x.y.z.dylib¹ in the directory /usr/local/lib with the following calls:

```
cd /usr/local/lib
```

```
ln -s libFedmlscCore.x.y.z.dylib libFedmlscCore.x.dylib
```

```
ln -s libFedmlscCore.x.dylib libFedmlscCore.dylib
```

Note: The library is compiled under Mac OS X V10.7.3 with Xcode V4.3.2 and is compatible with the architecture x86_64.

¹ x.y.z represents the version number

3.5. Source Code

After the installation the source code of the library can be found in the directory *src*.

The sub-directory *api* contains the new and simplified API, which embeds the old API from the sub-directory *impl*.

src\api\core	Description
Const.h/.cpp	contains common constants in the namespace OBID::Fedm::Core::Const
Error.h/.cpp	contains all error codes in the namespace OBID::Fedm::Core::Error
HexConvert.h/.cpp	contains global functions in the namespace OBID::Fedm::Core
ReaderTypes.h/.cpp	contains Reader Type Numbers in the namespace OBID::Fedm::Core::Const::ReaderType

src\api\core\i_scan	Description
ISO_IEC_ICManufacturerRegistration.h	contains Tag Manufacturer Codes in the namespace OBID::Fedm::Core::i_scan::ICManufacturer

src\api\core\i_scan\reader	Description
ReaderModule.h/.cpp	Reader class. Embeds FEDM_ISCReaderModule.
ReaderCommandPara.h/.cpp	contains all command parameter names in the namespace OBID::Fedm::Core::i_scan::ReaderCommand
ReaderConfigPara.h/.cpp	contains all configuration parameter names in the namespace OBID::Fedm::Core::i_scan::ReaderCommand
IAsyncGroup.h/.cpp	internal interface in the ReaderModule class for asynchronous tasks
IBrmTableGroup.h/.cpp	internal interface in the ReaderModule class for Bufferd-Read-Mode table
IHmTableGroup.h/.cpp	internal interface in the ReaderModule class for Host-Mode table
ICommandGroup.h/.cpp	internal interface in the ReaderModule class for common commands
IConfigGroup.h/.cpp	internal interface in the ReaderModule class for Reader configuration management
IInfoGroup.h/.cpp	internal interface in the ReaderModule class with methods for information requests
IPortGroup.h/.cpp	internal interface in the ReaderModule class with port specific methods, e.g. connect
ITagGroup.h/.cpp	internal interface in the ReaderModule class for tag communication

src\api\core\i_scan>tag_handler	Description
TagHandler_Includes.h	include file
TagHandler_Types.h	mapping of all TagHandler classes into the namespace OBID::Fedm::Core::i_scan::TagHandler

src\impl\core	Description
FEDM.h, FEDM_ISC.h and FEDM_Xml.h	contains includes, constants and macros
FEDM_Base.h/.cpp	base class
FEDM_DataBase.h/.cpp	abstract base class derived from FEDM_Base
FEDM_Functions.h/.cpp	global functions
FEDM_XmlBase.h/.cpp	base class for serialization in XML
FEDM_XmlReaderCfgDataModul.h/.cpp	specialized class for serialization of the reader configuration in XML
FEDM_XmlReaderCfgProfileModul.h/.cpp	specialised class für serialization of a profile configuration in XML

src\impl\core\i_scan	Description
FEDM_ISCReader.h/.cpp	from FEDM_DataBase derived reader class
FEDM_ISCReaderModule.h/.cpp	from FEDM_ISCReader derived reader class
FEDM_ISCReaderInfo.h	Structure with information about the reader
FEDM_ISCReaderDiagnostic.h	Structure with state information about the reader
FEDM_TabItem.h	base class for tables
FEDM_BRMTabItem.h/.cpp	table class for Buffered Read Mode
FEDM_ISOTabItem.h/.cpp	table class for Host-Commands
FEDM_ISCReaderID.h	access constants for reader configuration and temporary communication parameter
FEDM_ISCReaderConfig_XXX.h/.cpp	files with access constants for reader configuration

src\impl\core\i_scan\classic_pro	Description
FEDM_ISCReaderConfig_XXX.h/.cpp	files with access constants for reader configuration

src\impl\core\i_scan\function_unit	Description
FEDM_ISCFunctionUnit.h/.cpp	specialized class for Function Units (Multiplexer, Dynamic Antenna Tuner)
FEDM_ISCFunctionUnitID.h	access constants for temporary communication parameter for Function Units

src\impl\core\i_scan\peripheral_devices	Description
FedmlscPeripheralDevice.h/.cpp	base class for external Units (e.g. People-Counter)
FedmlscPeopleCounter.h/.cpp	from FedmlscPeripheralDevice derived class for People-Counter
FedmlscExternalIO.h/.cpp	from FedmlscPeripheralDevice derived class for External-IO

src\impl\core\i_scan>tag_handler	Description
FedmlscTagHandler.h/.cpp	base class TagHandler
FedmlscTagHandler_Includes.h	include file
FedmlscTagHandler_XXX.h/.cpp	specialised TagHandler class, derived from FedmlscTagHandler

src\impl\core\i_scan\utility	Description
FedmlscReport_ReaderInfo.h/.cpp	report class containing important static reader information
FedmlscReport_ReaderDiagnostic.h/.cpp	report class containing important status information from reader

3.6. Dependencies

The FEDM class library depends on the function libraries for the communications interfaces (FECOM, FEUSB, FETCP) and the reader family (FEISC, FEFU, FETCL). These libraries are part of the respective SDK and must be installed on the target system.

The class FEDM_ISCReader contains a method *EvalLibDependencies* for the verification of the version numbers of the dependent function libraries. It is recommended to invoke this method once in the application after the program started.

4.Integration into application projects

The library FEDM can be linked only from 32-Bit applications. 64-Bit application development is not supported. This is true for Windows and Linux.

4.1.Supported Development Tools

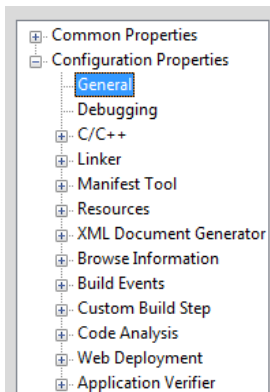
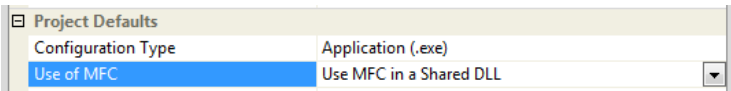
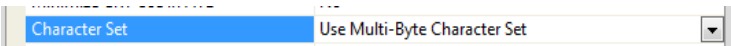
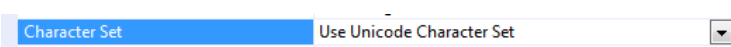
Operating System	IDE	Support
Windows XP / Vista / 7 / 8	Visual Studio 6	yes
	Visual Studio 2005 / 2008 / 2010 / 2012	yes, beginning with Professional Version
	Borland C++ Builder	on request
	Embarcadero C++ Builder	on request
Windows CE	eMbedded Visual C++ 4	no
	Visual Studio 2005 / 2008	yes, beginning with Professional Version
Linux	GCC	yes
Mac OS X	GCC	yes, for projects with x86_64 architecture
	Xcode ≥ V4.3.2	yes, for projects with x86_64 architecture

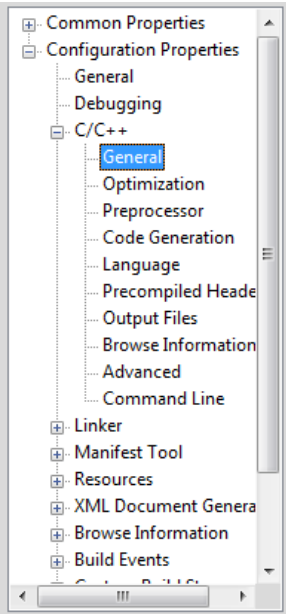
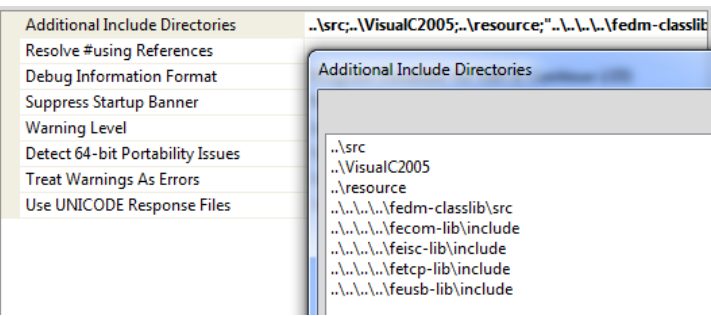
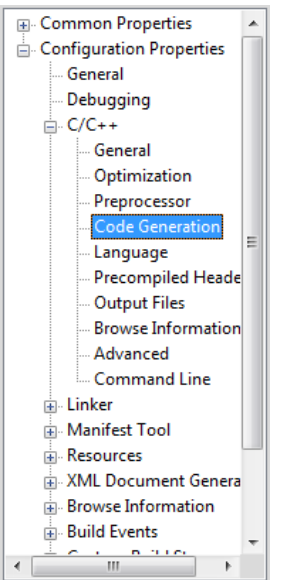
4.2. The quick way

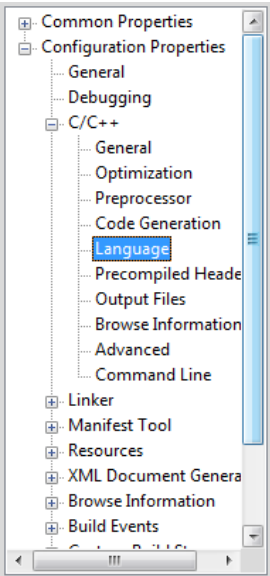

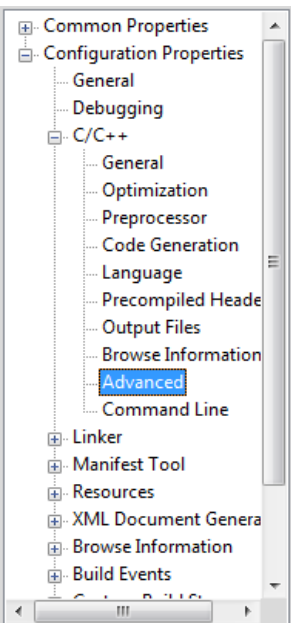
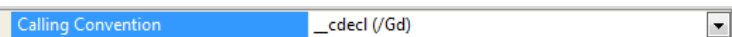
All preprocessor definitions are compiled in the include file FedmlscCore.h for using the compiled libraries and it is sufficient only to link this include file for the compiler.

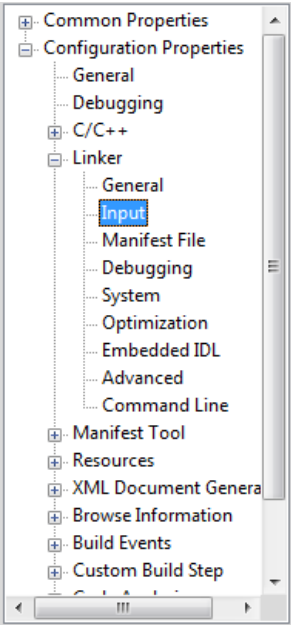
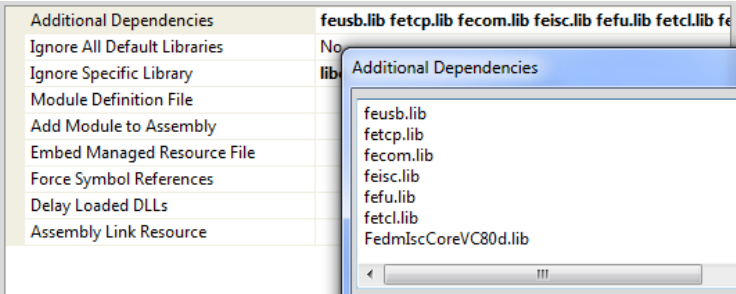
4.2.1. Incorporating into Visual Studio

Important settings for Visual Studio projects (Example VS2005):

Project Group	Property Setting
	 <p>For Windows:</p>  <p>For Windows CE:</p> 

Project Group	Property Setting				
	<p>Add all required directories for Header files.</p> 				
	<p>for Release Version:</p> <table border="1" data-bbox="639 925 1433 976"><tr><td>Runtime Library</td><td>Multi-threaded DLL (/MD)</td></tr></table> <p>for Debug Version:</p> <table border="1" data-bbox="639 1104 1433 1155"><tr><td>Runtime Library</td><td>Multi-threaded Debug DLL (/MDd)</td></tr></table>	Runtime Library	Multi-threaded DLL (/MD)	Runtime Library	Multi-threaded Debug DLL (/MDd)
Runtime Library	Multi-threaded DLL (/MD)				
Runtime Library	Multi-threaded Debug DLL (/MDd)				

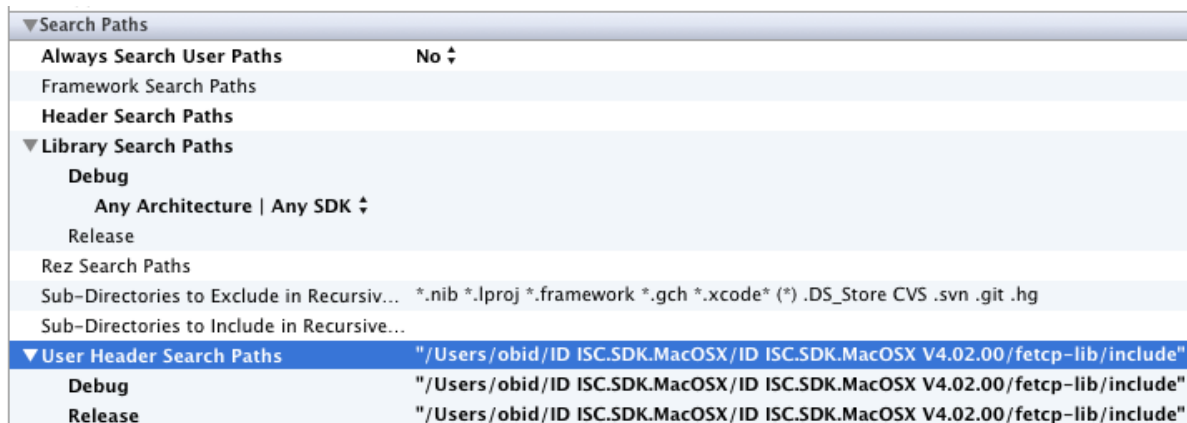
Project Group	Property Setting
	
	

Project Group	Property Setting
	<p>add FedmIscCoreVCxx.lib (Release-Version) respectively FedmIscCoreVCxxd.lib (Debug-Version)</p> 

4.2.2. Incorporating into Xcode

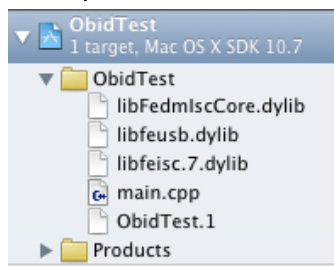
1. Add path for the header file in project settings (User Header Search Paths in category Search Paths)

Example:



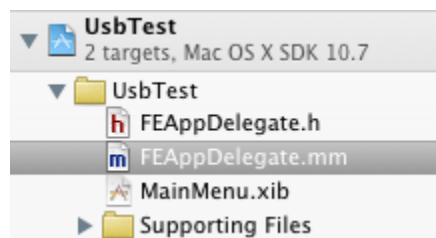
2. add all required dylib files with drag'n drop to your project

Example:



3. For Objective-C project: rename alle file extensions of source files from .m into .mm if C++ classes of FEDM were used. This activates the C++ compiler for the renamed source files.

Beispiel:



4.3. Manual integration

If you want to or have to integrate the source code of the library in the application directly, the necessary steps are described below.

Selecting manual integration has the option for Visual Studio projects to bind the MFC statically.

4.3.1. Include files

Add the include file FedmiscCore.h to the project. This uses the preprocessor definitions described below and integrates other include files from the src subdirectory.

Important: Comment out the line `#define _FEDM_DLL` or use the preprocessor definition `_FEDM_NO_DLL`. The preprocessor definition `_FEDM_DLL` must only be used if the precompiled library is integrated for the runtime.

4.3.2. Source code files

Add all source code files from the subdirectories

```
src/api/core  
src/api/core/i_scan  
src/api/core/i_scan/reader  
src/api/core/i_scan/tag_handler  
src/impl/core  
src/impl/core/i_scan  
src/impl/core/i_scan/function_unit  
src/impl/core/i_scan/classic_pro  
src/impl/core/i_scan/tag_handler  
src/impl/core/i_scan/utility  
src/impl/core/i_scan/peripheral_devices
```

to the project.

The MS Visual C++ development environment also requires the following project settings:

The *.cpp files of the library FEDM which have previously been added to the project must be released from the automatic use of the precompiled header file. This can be done on the tab C/C++ in the category "Precompiled header" by selecting the option "Do not use precompiled header".

4.3.3.Required project settings

In general you must set a preprocessor definition for each communications library you are using (already set in FedmlscCore.h):

- FECOM: **_FEDM_COM_SUPPORT**
- FEUSB: **_FEDM_USB_SUPPORT**
- FETCP: **_FEDM_TCP_SUPPORT**

4.3.4.Windows

Under Windows you must set the preprocessor definition **_FEDM_WINDOWS** (already set in FedmlscCore.h).

4.3.5.Linux

Under Linux you must set the preprocessor definition **_FEDM_LINUX** (already set in FedmlscCore.h).

4.3.6.Optional project settings

For activating the XML support (serialization of the reader configuration to a file) the preprocessor definition **_FEDM_XML_SUPPORT** must be set (already set in FedmlscCore.h).

Projects using TagHandler classes must set the pre-processor definition **_FEDM_TAG_HANDLER**. The amount of TagHandler-Types can be reduced with **_FEDM_NO_TAG_HANDLER_EPC_C1_G2**, **_FEDM_NO_TAG_HANDLER_ISO14443**, **_FEDM_NO_TAG_HANDLER_ISO15693** and **_FEDM_NO_TAG_HANDLER_ISO18000_3M3**.

Projects, who are based on the Microsoft Foundation Classes (MFC) and want participate of the MFC interface of the FEDM (basically CString), must have set the pre-processor definition **_FEDM_MFC_SUPPORT** (already set in FedmlscCore.h).

Projects, which prefer the static binding of the transport libraries FECOM, FEUSB and FETCP must have set the pre-processor definition **_FEDM_SUPPORT_SLINK** (not set in FedmlscCore.h).

Projects using the main include file FedmlscCore.h which wants additionally exclude unnecessary parts can set one or more of the following preprocessor definitions:

_FEDM_NO_XML_SUPPORT, **_FEDM_NO_COM_SUPPORT**, **_FEDM_NO_USB_SUPPORT**,
_FEDM_NO_TCP_SUPPORT, **_FEDM_NO_TAG_HANDLER**, **_FEDM_NO_FU_SUPPORT**,
_FEDM_NO_MFC_SUPPORT, **_FEDM_NO_PD_SUPPORT**

All named pre-processor definitions admit the customization of the library.

5. Installation on the target computer

Together with the application files, the runtime file of the library FedmlscCore (if dynamically linked) and the runtime files of the function libraries FECOM, FEUSB, FETCP, FEISC, FETCL and FEFU must be installed on the target computer.

5.1. 32-Bit Libraries on a 32- and 64-Bit Windows

It is recommended to keep the library files in the directory of the application. This avoids version conflicts with later installations which also install these library files, but possibly different versions.

The library files FedmlscCoreVC80.dll, FedmlscCoreVC90.dll, FedmlscCoreVC100.dll and FedmlscCoreVC110.dll depend on newer C/C++ runtime libraries which are usually not present on the target computer. Therefore, they must be installed. So-called Merge Modules are provided with Visual Studio which can be incorporated in a Setup project and which install the MFC libraries. The following table lists for each FedmlscCore library file the Merge Module to be installed to.

Library File	MFC Version	Merge Modules
FedmlscCoreVC60.dll	Version 6.0	MFC library is installed from Windows 2000
FedmlscCoreVC80.dll	Version 8.0 (8.0.50727.6195 s. MS11-025 ¹)	Microsoft_VC80_MFC_x86.msm Microsoft_VC80_CRT_x86.msm policy_8_0_Microsoft_VC80_MFC_x86.msm policy_8_0_Microsoft_VC80_CRT_x86.msm
FedmlscCoreVC90.dll	Version 9.0 (9.0.30729.6161 s. MS11-025 ²)	Microsoft_VC90_MFC_x86.msm Microsoft_VC90_CRT_x86.msm policy_9_0_Microsoft_VC90_MFC_x86.msm policy_9_0_Microsoft_VC90_CRT_x86.msm
FedmlscCoreVC100.dll	Version 10.0 (10.0.30319.460 s. MS11-025 ³)	Microsoft_VC100_MFC_x86.msm Microsoft_VC100_CRT_x86.msm
FedmlscCoreVC110.dll	Version 11.0 (11.0.51106.1)	Microsoft_VC110_MFC_x86.msm Microsoft_VC110_CRT_x86.msm

Alternatively, the installation of the Visual C++ Runtime Libraries can be realized with the download site of Microsoft. For each version of MFC you can find a file called vcredist_x86.exe for download.

1st Note: The file vcredist_x86.exe must be of version of at least the specified number above.

2nd Note: Merge Modules can only be updated with Windows Update.

¹ Microsoft Security Bulletin Article-ID: 2538218 from Juni 14, 2011

² Microsoft Security Bulletin Article-ID: 2538243 from Juni 14, 2011

³ Microsoft Security Bulletin Article-ID: 2542054 from Juni 14, 2011

3rd Note: Debug versions of FEDM, marked with a **d** at the end of the file name (e.g. FedmlscCoreVC80**d**.dll) cannot be installed on target computers. The reason is that every merge module does install only the release version of a MFC library.

5.2. 64-Bit Libraries on a 64-Bit Windows

It is recommended to keep the library files in the directory of the application. This avoids version conflicts with later installations which also install these library files, but possibly different versions.

The library file FedmlscCoreVC100.dll or FedmlscCoreVC110.dll depends on newer C/C++ runtime libraries which are usually not present on the target computer. Therefore, they must be installed. So-called Merge Modules are provided with Visual Studio which can be incorporated in a Setup project and which install the MFC libraries. The following table lists for each FedmlscCore library file the Merge Module to be installed to.

Library File	MFC Version	Merge Modules
FedmlscCoreVC100.dll	Version 10.0 (10.0.30319.460 s. MS11-025 ¹)	Microsoft_VC100_MFC_x64.msm Microsoft_VC100_CRT_x64.msm
FedmlscCoreVC110.dll	Version 11.0 (11.0.51106.1)	Microsoft_VC110_MFC_x64.msm Microsoft_VC110_CRT_x64.msm

Alternatively, the installation of the Visual C++ Runtime Libraries can be realized with the download site of Microsoft. For each version of MFC you can find a file called vcredist_x64.exe for download.

1st Note: The file vcredist_x64.exe must be of version of at least the specified number above.

2nd Note: Merge Modules can only be updated with Windows Update.

3rd Note: Debug versions of FEDM, marked with a **d** at the end of the file name (e.g. FedmlscCoreVC100d.dll) cannot be installed on target computers. The reason is that every merge module does install only the release version of a MFC library.

¹ Microsoft Security Bulletin Article-ID: 2542054 from Juni 14, 2011

5.3. 32- and 64-Bit Linux

The installation on the target is equivalent to [3.3. 32- and 64-Bit Linux](#). But only the runtime files *.so may be installed on the target computer.

The installation instructions for the dependent function libraries can be found in the respective manuals.

5.4. 64-Bit Mac OS X

The installation on the target is equivalent to [3.4. 64-Bit Mac OS X](#). But only the runtime files *.dylib may be installed on the target computer.

The installation instructions for the dependent function libraries can be found in the respective manuals.

6. Class description

The ID FEDM class library undergoes a continuous adaptation process. We will make every effort to maintain the documented status. Changes are still however possible.

6.1. FEDM_ISCReader

The class FEDM_ISCReader is based on the abstract base class FEDM_DataBase and thus inherits the general, type-neutral interface. The abstract interface methods are implemented in FEDM_ISCReader, so that you can work with an object from this class. Alternately you may build upon this class to derive your own reader class and design it such that you can add new functionalities or overwrite methods and implement a different behavior.

6.1.1. Implemented data containers

Data container	Description
m_RFC_EEData	For reader configuration parameters (for all OBID i-scan® and OBID® <i>classic-pro</i> Reader types)
m_RFC_RAMData	For temporary reader configuration parameters (for all OBID i-scan® and OBID® <i>classic-pro</i> Reader types)
m_ACC_EEData	for additional configuration parametersforReaderwith AC-Controller (ID ISC.LRU2000)
m_ACC_RAMData	for additional temporary configuration parametersforReaderwith AC-Controller(ID ISC.LRU2000)
m_TmpData	For general temporary protocol data (for all OBID i-scan® and OBID® <i>classic-pro</i> Reader types)

The size of the data containers is determined statically in the class constructor. All data containers are initialized in the constructor with 0x00.

Any non-listed data container has a length of 0 and is not usable.

6.1.2. Implemented tables

Table	Description
m_ISOTable	Supports data exchange with transponders via the ISO host commands.
m_BRMTTable	Collects the data provided by a long-range Reader of type ISC.LRxxx in Buffered Read Mode.

The table sizes must be set with SetTableSize before using for the first time! The size is determined by the maximum number of transponders that are in the Reader's antenna field at the same time.

In applications that do not use the Buffered Read Mode of the reader, you do not need to set a table size for m_BRMTTable. The same is true of course vice-versa for the table m_ISOTable.

6.1.3. Methods (public)

Method	Description
EvalLibDependencies	Method verifies the compatibility with dependent function libraries.
SendProtocol	The central communication method. For more details see section 6.4.3. Examples for using the method SendProtocol .
GetLastProt	Method for getting the last send or receive protocol. sID="SEND" gives you the last send protocol sID="SENDSTR" gives you the last send protocol with preceding date/time of day sID="REC" gives you the last receive protocol sID="RECSTR" gives you the last receive protocol with preceding date/time of day
FindBaudRate	Method finds a reader identifiable by the port handle (stored in the reader object in FEISC) and gets the baud rate and the protocol frame. This method cannot be used with the USB or TCP/IP port.
SetReaderType	Set of the reader type for which the reader class should be represented for. It is recommended to call the method ReadReaderInfo of FEDM_ISCReaderModule at once after a successful connection. This method sets also the reader type.
SetPortHnd	Sets the port handle as a parameter in the reader object in the FEISC library
GetPortHnd	Gets the port handle of the reader object in the FEISC library
GetReaderName	returns the reader name
GetReaderInfo	returns a pointer to the structure FEDM_ISC_READER_INFO
GetReaderDiagnostic	returns a pointer to the structure FEDM_ISC_READER_DIAGNOSTIC
SetProtocolFrameSupport	Selects the protocol type for communication with the reader (preselection: Standard Protocol Frame). Transfer values are: FEDM_PRT_FRAME_STANDARD FEDM_PRT_FRAME_ADVANCED
GetProtocolFrameSupport	Queries the protocol type.
DisableReadCfgBeforeWriteCfg	This method disables the check for Read before Write behavior before every writing of a configuration blocks into the reader. It is recommended to let this check enabled.
EnableTagHandler	enables the TagHandler support
GetLastError	Gets the last error code stored at FEDM_ISC_TMP_LAST_ERROR in the data container TmpData.
GetLastStatus	Gets the status byte of the last protocol which is stored at FEDM_ISC_TMP_LAST_STATE in the data container TmpData.
GetErrorText	Gets a text corresponding to a sent error code. The error code may also come from the function collection sector ID FEISC or the underlying communication library FECOM, FETCP or FEUSB. The language for the text can be set with the method SetLanguage of the base class FEDM_Base.
GetStatusText	Gets a text corresponding to the sent status byte. The language for the text can be set with the method SetLanguage of the base class FEDM_Base.
Serialize	Main method for serializing. Allows serializing of the container data in files. Two versions are implemented: one version for file type XML (s. 6.4.4. Serializing in XML-Format) and a second version for MFC-based applications.

Method	Description
GetCommandPara	<p>The central (overlaid) method for reading a command parameter value from a data container.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a string with the parameter name from the namespace ReaderCommand according the definition in the system manual of the reader.</p>
SetCommandPara	<p>The central (overlaid) method for writing a command parameter value to a data container.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a string with the parameter name from the namespace ReaderCommand according the definition in the system manual of the reader.</p>
GetConfigPara	<p>The central (overlaid) method for reading a configuration parameter value from a data container.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a string with the parameter name from the namespace ReaderConfig according the definition in the system manual of the reader. A third parameter defines the location RAM or EEPROM.</p>
SetConfigPara	<p>The central (overlaid) method for writing a configuration parameter value to a data container.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a string with the parameter name from the namespace ReaderConfig according the definition in the system manual of the reader. A third parameter defines the location RAM or EEPROM.</p>
TestConfigPara	<p>Checks, whether a string with the parameter name from the namespace ReaderConfig is defined for a reader type</p>
GetTableData	<p>The central (overlaid) method for reading a parameter value or data blocks from a table.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a Table-ID as a parameter; this ID differentiates the tables m_ISOTable and m_BTMTTable. The parameter uiDataID has an ID for the value to be written. All access IDs are listed in the file FEDM_ISC.h. Which access IDs are supported can be determined in 7.3.4. Constants for uiDataID.</p> <p>Alternatively, this method can be replaced by GetISOTableItem and GetBRMTTableItem which enables the direct access to all transponder values.</p>
SetTableData	<p>The central (overlaid) method for writing a parameter value or data blocks to the table.</p> <p>This version supports data types: bool, UCHAR, UCHAR-Array, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a Table-ID as a parameter; this ID differentiates the tables m_ISOTable and m_BRMTTable. The parameter uiDataID has an ID for the value to be written. All access IDs are listed in the file FEDM_ISC.h. Which access IDs are supported can be determined in 7.3.4. Constants for uiDataID.</p> <p>Alternatively, this method can be replaced by GetISOTableItem and GetBRMTTableItem which enables the direct access to all transponder values.</p>
FindTableIndex	<p>The central (overlaid) method for getting the table index based on a value, starting at a Start-Index.</p>

Method	Description
	<p>This version supports data types: bool, UCHAR, UINT, __int64 and CString resp. AnsiString and STL-string.</p> <p>The method expects a Table-ID as a parameter; this ID differentiates the tables m_ISOTable and m_BTMTTable. The parameter uiDataID has an ID for the value to be found. All access IDs are listed in the file FEDM_ISC.h. Which access IDs are supported can be determined in 7.3.4. Constants for uiDataID.</p>
SetTableSize	<p>Sets the size of the table m_ISOTable or m_BRMTTable and initializes each table line with 0. You can change the size after the fact, though the old content is lost. The method expects a Table-ID as a parameter; this ID differentiates the tables m_ISOTable and m_BRMTTable.</p> <p>A second overloaded implementation has extended parameters for dimensioning of the data arrays RxDB and TxDB. This allows the adaptation of the memory requirement.</p>
GetTableSize	Gets the size of the table m_ISOTable or m_BRMTTable. The method expects a Table-ID as a parameter; this ID differentiates the tables M_ISOTable and m_BRMTTable.
GetTableLength	Gets the number of valid table entries in m_ISOTable or m_BRMTTable. The method expects a Table-ID as a parameter, which differentiates between m_ISOTable and m_BRMTTable.
ResetTable	<p>Resets the table m_ISOTable or m_BRMTTable. The method expects a Table-ID as a parameter, which differentiates between m_ISOTable and m_BRMTTable.</p> <p>Only the variables m_iISOTableLength or m_iBRMTTableLength are set to 0 (the table data are not deleted). The additional parameter uiDataFlags = FEDM_ISC_DATA_ALL initializes all data fields with 0.</p> <p>Default: No initialization of the data fields.</p>
GetPeripheralDevices	<p>Returns a sorted list with previously detected externalUnits (PeopleCounter, ExternalIO). The detection is executed by invoke of the command [0x66] Reader Info with mode 0x61 or with call of the method ReadReaderInfo() of class FEDM_ISCReaderModule.</p> <p>The sorted list does not contain any Function Units (Antenna Tuner, Multiplexer) looped into the antenna cable.</p>
GetISOTableItem	<p>For operating mode Host-Mode: method returns a pointer of a table item selected by index with the direct access to all transponder data.</p> <p>This method is an alternative to GetTableData.</p>
GetBRMTTableItem	<p>For operating modes Buffered-Read-Mode and Notification-Mode: method returns a pointer of a table item selected by index with the direct access to all transponder data.</p> <p>This method is an alternative to GetTableData.</p>

6.2. FEDM_ISCReaderModule

The class FEDM_ISCReaderModule is derived from the Reader class FEDM_ISCReader and implements high-level methods for the diverse communication port types and for the Notification. It is recommended to use this class in an application.

The constructor of the class generates automatically a Reader object in the FEISC function library.

6.2.1. Methods (public)

Method	Description
ConnectCOMM	Opens a serial port with the setting Baudrate=38400, Frame=8E1, Timeout=3000ms. NOTE: this method does not check the connection to a reader. This must be done with a call of FindBaudrate
ConnectUSB	Opens a USB-Channel for an USB-Reader. The Device-ID of the Reader must be read first with the function library FEUSB.
ConnectTCP	Opens a socket connection to a Reader.
DisConnect	Close the existing connection. When closing a TCP/IP connection, the last status of the connection is returned. But if the status TIME_WAIT is detected, a 0 will be returned to signal a successful disconnection. See also: 7.3. TCP-Status
IsConnected	Signals that the application has called one of the Connect methods and no further connection can be established. It is not checked, if the Reader is still connected and a communication is possible.
GetTcpConnectionState	Detects with a Kernelfunction the status of a TCP/IP connection. It is not possible to get the status with continuous polling to detect a broken connection caused by loss of power or lost network cable. This method can help to find reasons <u>after</u> a communication error. See also: 7.3. TCP-Status
ReaderAuthentication	Overloaded methods executes the Reader authentication to startup a secured session. These methods encapsulates the function FEISC_0xAE_ReaderAuthent. More information can be found in the manual H9391-xx-ID-B of the library FEISC in chapter „Secured data transmission with encryption“.
ReadReaderInfo	Method to request static reader information with multiple [0x66] Reader Info commands. The data are collected inside the structure FEDM_ISC_READER_INFO. Additionally, the reader type identifier is stored internally and guarantees the proper functionality.
ReadReaderDiagnostic	Method to request important status information from reader with multiple [0x6E] Reader Diagnostic commands. The data are collected inside the structure FEDM_ISC_READER_DIAGNOSTIC.
ReadCompleteConfiguration	Reads the complete reader configuration from the reader and saves it in the respective data container.

Method	Description
WriteCompleteConfiguration	Writes the complete reader configuration from the respective data container into the reader. This method fails, if the complete reader configuration is previously not read.
TransferReaderCfgToXmlFile	Reads the complete reader configuration from the reader, saves it in the respective data container and writes afterwards the configuration data into an XML file.
TransferXmlFileToReaderCfg	Opens an XML file, saves the configuration data in the respective data container and writes afterwards the complete reader configuration into the reader.
StartAsyncTask	Starts an asynchronous task for notifications. The Reader must work in Notification Mode or in Host Mode with support for notifications.
CancelAsyncTask	<p>Cancels the actual asynchronous task.</p> <p>Notification tasks must always be canceled with this function.</p> <p>The cancellation of the task is locked if the task execution is just inside the callback function. This prevents deadlocks. In this case this function returns directly with the return value FEISC_ERR_TASK_BUSY (-4084) and the application must invoke CancelAsyncTask until the return value is not -4084. On application-side the return from the callback function must be guaranteed.</p>
TriggerAsyncTask	Trigger the actual with TaskID=FEDM_TASKID EVERY_NEW_TAG initiated asynchronous task, which is waiting after a callback invoke for this trigger
SetPortHnd	<p>Saves the port handle and sets the communication mode.</p> <p>Use this method, if the communication port is not opened with one of the Connect method, but with the Open function of FECOM, FEUSB or FETCP.</p>
SetPortPara	Configures the open communication channel. The configuration parameters are documented in the manuals of FECOM, FEUSB or FETCP.
GetPortPara	Queries the configuration of the open communication channel. The configuration parameters are documented in the manuals of FECOM, FEUSB or FETCP.
cbsTaskRsp1	Static callback function for asynchronous task.
cbsTaskRsp2	Static callback function for asynchronous task.

Method	Description
Optional methods for Transponder communication based upon TagHandler classes	
TagInventory	Call of an Inventory and returning of all found tags in a TagList of type FedmIscTagHandler.
TagSelect	Call of a Select command. For some ISO 14443 Transponder thereturned TagHandler class is adjusted
GetTagHandler	Return of a TagHandler class for a Transponder identified by the serial number (UID)
GetSelectedTagHandler	Return of the TagHandler class of the selected transponder
GetTagList	The list with all available TagHandler classes is created and returned
CreateNonAddressedTagHandler	<p>Transponders supporting the non-addressed mode can communicate without an Inventory. If an application use only one Transponder type, but with chip-specific extensions and a TagHandler-Class is available, the necessary TagHandler can be created with this method.</p> <p>All TagHandler types are collected in the class FedmIscTagHandler, while not every TagHandler type is specified for non-addressed tag communication. More information about this communication mode can be found in the datasheet of the transponder.</p> <p>Important notes:</p> <ol style="list-style-type: none"> 1. A Reader object can always work with only one Non-Addressed TagHandler 2. Every call of CreateNonAddressedTagHandler destroys the actual Non-Addressed TagHandler 3. A mixed operation with Addressed and Non-Adressed TagHandler is not supported.
Convert_EPC_C1_G2_TagHandler	The specification for Class1 Gen2 contains no definition of identification attributes for manufacturer and chip types as for ISO 15693. If an application use only one EPC Class1 Gen2 Transponder type, but with chip-specific extensions and a TagHandler-Class is available, the necessary TagHandler can be created with this method.

6.3. Concept of TagHandler classes

The concept of TagHandler classes provides a new library part for more efficient programming with different transponder types. TagHandler can be used only when the Reader works in **Host-Mode** and when the support is activated in the class FEDM_ISCReader with the method EnableTagHandler.

The concept is based on the automatic identification of the type of the transponder after a successful inventory. With ISO 15693 compliant transponders the manufacturer ID and the chipID, which are part of the serial number, are evaluated. With ISO 14443 compliant transponders the type of the TagHandler can be determined after a mandatory Select command based on the returned Card-Info or, in case of the explicit selection of a transponder driver with the Select comand, the transponder driver selects the type of the TagHandler.

All TagHandler classes are derived from the base class FedmIscTagHandler. Furthermore, the relationship between the different transponder types is mapped to derivations between TagHandler classes.

TagHandler classes are created, managed and deleted internally. After each call of TagInventory (or call of SendProtocol(0xB0) for Cmd=0x01) the reader class checks the present state of each TagHandler and removes the handler if the dedicated transponder is out of field. Thus, the live cycle of a TagHandler is normally one inventory cycle.

Short example:

```
FedmIscTagHandler* pTagHandler = NULL;

// get tags
FEDM_ISC_TAG_LIST* pTagList = m_Reader.TagInventory();

// do we have tags received?
FEDM_ISC_TAG_LIST_ITERATOR itor = pTagList->begin();
if(itor == pTagList->end())
    return;

// select tag with driver for MIFARE DESFire and return
// specialized tag handler
pTagHandler = Reader.TagSelect(itor->second, 9);

// check specialized tag handler
if(dynamic_cast<FedmIscTagHandler_ISO14443_4_MIFARE_DESFire*>(pTagHandler) != NULL)
{
    // do anything with the tag (e.g. authentication)
    FedmIscTagHandler_ISO14443_4_MIFARE_DESFire* pDesFire =
        (FedmIscTagHandler_ISO14443_4_MIFARE_DESFire*)pTagHandler;

    int iRetCode = pDesFire->Authenticate(uiAppID, ucReaderKeyIndex, ucDesFireKeyNo);
    ...
}
```

FedmIscTagHandler_ISO14443_4_MIFARE_DESFire
+FedmIscTagHandler_ISO14443_4_MIFARE_DESFire()
+FedmIscTagHandler_ISO14443_4_MIFARE_DESFire()
+Init()
+GetTagName()
+GetErrorSource()
+GetErrorCode()
+Authenticate()
+ChangeKeySettings()
+ChangeKey()
+SetConfiguration()
+GetKeyVersion()
+CreateApplication()
+DeleteApplication()
+GetApplicationIDs()
+FreeMemory()
+GetDFNames()
+GetKeySettings()
+SelectApplication()
+FormatPICC()
+GetVersion()
+GetCardUID()
+ChangeFileSettings()
+GetFileIDs()
+GetFileSettings()
+CreateStdDataFile()
+CreateBackupDataFile()
+CreateValueFile()
+CreateLinearRecordFile()
+CreateCyclicRecordFile()
+DeleteFile()
+GetISOFileIDs()
+ReadStandardData()
+WriteStandardData()
+GetValue()
+Credit()
+Debit()
+LimitedCredit()
+WriteRecord()
+ReadRecords()
+ClearRecordFile()
+CommitTransaction()
+AbortTransaction()

6.4. Working with the Reader classes

Applications using the Host-Mode can be designed to use the table-oriented programming methods or can use TagHandler classes. It is recommended to check first, if the use of the TagHandler classes is applicable, because these classes are based on the table-oriented management of transponders and have the more efficient API. Only in the case that TagHandler classes are missed (transponder type is not supported by the SDK) or the use of the non-addressed Mode for transponder communication is necessary, the (old) table oriented methods have to be used. Chapter [6.5. Table for ISO Host Commands](#) contains then the best startup information.

6.4.1. Important initializations

Before using the protocol method for the first time, some initializing must be performed:

1. Bus address The bus address for the Reader is preset in the class for 255. To set a different address, use the SetBusAddress method.

Note: The bus address is relevant for the serial communication.
2. ReaderHandle **FEDM_ISCReader:** The handle of a Reader object in the FEISC function library must always be stored in an instance of the Reader class using the SetReaderHnd method.

FEDM_ISCReaderModule: The constructor of the class generates automatically a Reader object in the FEISC function library.
3. PortHandle **FEDM_ISCReader:** The handle for an interface that was opened with FECOM, FETCP or FEUSB must be stored in the Reader object of the FEISC function library. This can be done either by creating the Reader object using FEISC_NewReader or after the fact by using the method SetPortHnd. Making the change after the fact is also always possible if you need to change the port during run time.

FEDM_ISCReaderModule: The opening of the communication port is applicable with one of the connect methods. The interface handle is set internally.
4. Reader type The reader type must be set in the reader class with one of two options:
 1. The call of the method ReadReaderInfo after a successful connection (recommended).
 2. Set of reader type with the method SetReaderType. The constants of all reader types are listed in the file FEDM_ISC.h.
5. Language support Error texts can be invoked in several languages. You can set the (optional) language using the SetLanguage method. The preset is for English.
6. Table size The integrated tables for Buffered Read Mode (BRM) and ISO Host

Mode are not initialized. Before the initial communication, you must set the table size using the method `SetTableSize`. The size is selected equal to the maximum number of transponders located in the antenna field at the same time.

Only the size of the table actually being used needs to be set.

7. TagHandler-Support (optional) The support for TagHandler classes is disabled by default. If TagHandler classes are used in an application, the support must be enabled with the method `EnableTagHandler..`

6.4.2. Management of the reader configuration

Each OBID *i-scan*® and OBID®*classic-pro* reader are controlled by parameters which are stored grouped in blocks in an EEPROM and are described in detail in the system manual for the respective reader. After switching on or resetting the reader, all parameters are loaded into RAM, evaluated and incorporated in the controller.

All parameters can be modified using a protocol so that the behaviour of the reader can be adapted to the application. Ideally, the program ISOStart is used for this adaptation and normally no parameters have to be changed in the application. Despite this, it can happen that one or more parameters from a program have to be changed. This chapter should familiarise you with the procedure using the reader class as an example.

A common characteristic of all readers is the grouping in blocks of thematically related parameters to 14 bytes per configuration block. Each parameter cannot be addressed individually but must always be retrieved together with a configuration block using the protocol [0x80] Read Configuration, then modified and finally written back to the reader with the protocol [0x81] Write Configuration. This cycle must always be complied with and is also checked by the reader class FEDM_ISCReader. This means that writing a configuration block without previously reading the same block is not possible.

The reader class manages the configuration data in a (public) byte array `m_RFC_EEData` for data from the EEPROM and `m_RFC_RAMData` for data from the RAM of the reader. The differentiation is important as changes in RAM are used immediately while changes in the EEPROM of the reader do not become active until after a reset. Therefore the reader class has its own byte arrays for both configuration sets.

Using the example of the configuration block CFG2 of the reader ID ISC.LR2000 which contains parameters for the configuration of the digital inputs and outputs, the following should explain how you specifically modify a parameter using the reader class FEDM_ISCReader.

Byte	0	1	2	3	4	5	6
Contents	IDLE-MODE		FLASH-IDLE		IN-ACTIVE	0x00	REL1-TIME
Default	0x88A8		0xCC00		0x00		0x00

Byte	7	8	9	10	11	12	13
Contents	REL1-TIME	OUT1-TIME		REL2-TIME		REL3-TIME	REL4-TIME
Default	0x00	0x0000		0x0000			0x0000

IDLE-MODE:

Defines the status of the signal emitters (OUT1 and RELx) during the idle mode.

Bit:	15	14	13	12	11	10	9	8
Function:	REL1 mode		0	0	OUT1 mode		0	0

	7	6	5	4	3	2	1	0
	REL2 mode		REL3 mode		REL4 mode		0	0

Mode	Function	
b 0 0	UNCHANGED	no effect on the status of the signal emitter
b 0 1	ON	signal emitter on
b 1 0	OFF	signal emitter off
b 1 1	FLASH	signal emitter alternating on

The assignment of the configuration block CFG2 is shown above. The parameter IDLE-MODE occupies two bytes and contains sub parameters for four relays and one digital output. Each output can be configured for one of four states according to the table. As the IDLE-MODE field is not greyed out, the modification can be made in the RAM of the reader.

The following steps are now necessary for the modification of REL1 mode inside IDLE-MODE:

```
// the example shows the reading, modification and rewriting of one block of the reader configuration
// m_Reader is an object of the reader class FEDM_ISCReader or FEDM_ISCReaderModule

unsigned char ucCfgAdr = 2;           // Address of the configuration block
bool bEEProm = false;                // Configuration data from/in RAM of the reader
unsigned int uIdleModeRel1           // Parameter IDLE-MODE

// Defaults for the next SendProtocol
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG, (UCHAR)0x00); // reset everything
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG_ADR, ucCfgAdr); // set address
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG_LOC, bEEProm);  // set memory location on RAM

// read configuration data
m_Reader.SendProtocol(0x80);

uIdleModeRel1 = 3;                   // REL1 alternating on (Note: set frequency in Parameter IDLE-FLASH)

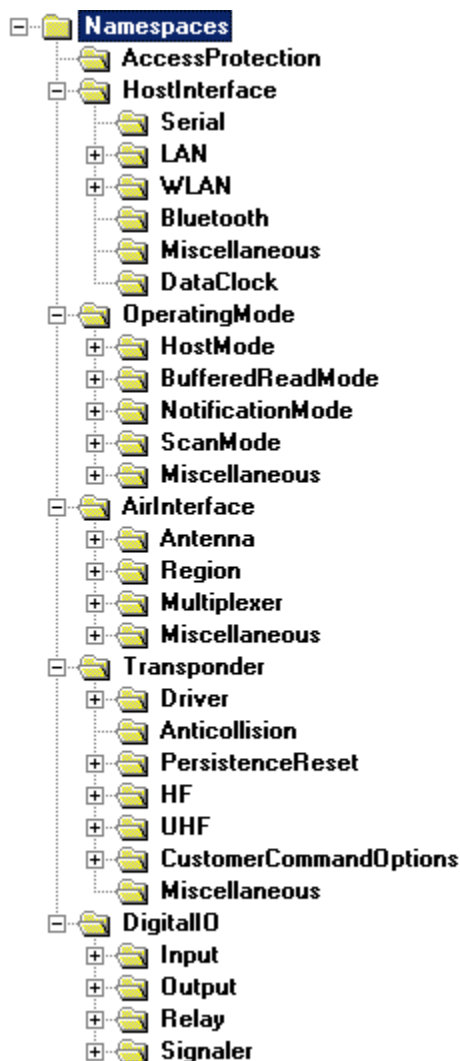
m_Reader.SetConfigPara(ReaderConfig::DigitalIO::Relay::No1::IdleMode, uIdleMode, false); // change value in RAM

// Defaults for the next SendProtocol
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG, (UCHAR)0x00); // reset everything
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG_ADR, ucCfgAdr); // set address
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG_LOC, bEEProm);  // set memory location on EEPROM
```

```
// rewrite configuration data
```

```
m_Reader.SendProtocol(0x81);
```

The methods `GetConfigPara` and `SetConfigPara` receive a string with parameter name from the namespace `ReaderConfig`. This main namespace contains further namespaces in tree order and collects all parameter names of all OBID *i-scan*® and OBID®*classic-pro* reader in a unique manner. The picture below shows the main namespaces.



The advantage of this schematic is the support by the intellisense functionality of modern IDEs which speeds-up the search for the proper parameter name.

6.4.3. Examples for using the method SendProtocol

SendProtocol is of key importance to the protocol transfer. For this reason an example is shown for each control byte¹ which is intended to show which data are to be stored in data containers with which access constants before each protocol transfer and which data are available after the protocol transfer. Some protocols allow various data to be transferred. In such cases only a single example is shown.

All the access constants are listed in the file FEDM_ISCReaderID.h and should be studied carefully in conjunction with the explanation of the protocol data found in the system manual.

For reasons of clarity the processing of the return values of the methods is not shown here. Of course it should always be included in applications.

[Control byte] Protocol	Example
[0x18] Destroy	<pre> UCHAR ucMode = 0; // mode byte UCHAR ucPW[3]; // password UCHAR ucEPC[32]; // buffer for EPC int iEpcLen = 0; // number of bytes in ucEPC // get the data, e.g. from a text control // get the number of bytes in ucEPC SetData(FEDM_ISC_TMP_EPC_DESTROY_MODE, (UCHAR)0); SetData(FEDM_ISC_TMP_EPC_DESTROY_PASSWORD, ucPW, 3); SetData(FEDM_ISC_TMP_DESTROY_EPC, ucEPC, iEpcLen); SendProtocol(0x18); </pre>
[0x1A] Halt	SendProtocol (0x1A);
[0x1B] Reset QUIET Bit	SendProtocol (0x1B);
[0x1C] EAS	SendProtocol (0x1C);
[0x21]Read Buffer	<pre> UCHAR ucDataSets = 1; // number of data sets requested UCHAR ucTrData = 0; // data set structure UCHAR ucRecSets = 0; // number of data sets in the protocol SetData(FEDM_ISCLR_TMP_BRM_SETS, ucDataSets); SendProtocol(0x21); // read data from transponder with Buffered Read Mode GetData(FEDM_ISCLR_TMP_BRM_TRDATA, &ucTrData); GetData(FEDM_ISCLR_BRM_RECSETS, &ucRecSets); // All other transponder are contained in m_Table. Example for data access see // 6.6.1. Examples for using of the table </pre>

¹not all commands are supported by every Reader. Detailed informations about the supported commands can be found in the system manual of the Reader.

[Control byte] Protocol	Example
[0x22] Read Buffer	<pre> UINT uiDataSets = 1; // number of data sets requested UCHAR ucTrData = 0; // data set structure UINT uiRecSets = 0; // number of data sets in the protocol SetData(FEDM_ISC_TMP_ADV_BRM_SETS, uiDataSets); SendProtocol(0x22); // read data from transponder with Buffered Read Mode GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1, &ucTrData); GetData(FEDM_ISC_TMP_ADV_BRM_RECSETS, &uiRecSets); // All other transponder are contained in m_Table. Example for data access see // 6.6.1. Examples for using of the table </pre>
[0x31] Read Data Buffer Info	<pre> UINT uiTabSize = 0; // size of the data buffer UINT uiTabStart = 0; // start address for the first data set UINT uiTabLen = 0; // number of data sets in the data buffer SendProtocol(0x31); GetData(FEDM_ISCLR_TMP_TAB_SIZE, &uiTabSize); GetData(FEDM_ISCLR_TMP_TAB_START, &uiTabStart); GetData(FEDM_ISCLR_TMP_TAB_LEN, &uiTabLen); </pre>
[0x32] Clear Data Buffer	SendProtocol (0x32);
[0x33] Initialize Buffer	SendProtocol (0x33);
[0x52] Baud Rate Detection	SendProtocol (0x52);
[0x55] Start Flash Loader	SendProtocol (0x55);
[0x63] CPU Reset	SendProtocol (0x63);
[0x64] System Reset	<pre> UCHAR ucMode = 0; // LRU1000 RF-Controller (1 for LRU1000 AC-Controller) SetData(FEDM_ISC_TMP_SYSTEM_RESET_MODE, ucMode); SendProtocol(0x64); </pre>
[0x65] Get Software Version	<pre> string sSoftVer; // Software Version as STL-string SendProtocol(0x65); GetData(FEDM_ISC_TMP_SOFTVER, sSoftVer); </pre>
[0x66] Get Reader Info	<pre> string sSoftVer; // Software-Version as STL-string SetData(FEDM_ISC_TMP_READER_INFO_MODE, (UINT)0); // same as [0x65] //SetData(FEDM_ISC_TMP_READER_INFO_MODE, (UINT)1); // LRU1000: AC-Controller SendProtocol(0x66); GetData(FEDM_ISC_TMP_SOFTVER, sSoftVer); // same as [0x65] //GetData(FEDM_ISC_TMP_READER_INFO, sSoftVer); // LRU1000: AC-Controller </pre>
[0x69] RF Reset	SendProtocol (0x69);
[0x6A] RF ON/OFF	<pre> UCHAR ucRF = 1; // RFON SetData(FEDM_ISC_TMP_RF_ONOFF, ucRF); SendProtocol(0x6A); </pre>

[Control byte] Protocol	Example
[0x6B] Centralized RF Sync	SetData (FEDM_ISC_TMP_0x6B_MODE, (unsigned char)0); SetData (FEDM_ISC_TMP_0x6B_TX_CHANNEL, (unsigned char)1); SetData (FEDM_ISC_TMP_0x6B_TX_PERIOD, (unsigned int)1); SetData (FEDM_ISC_TMP_0x6B_RES1, (unsigned char)0); SetData (FEDM_ISC_TMP_0x6B_RES2, (unsigned char)0); SendProtocol (0x6B);
[0x6C] Set Noise Level	UINT uiNLMin = 500; // minimum noise level UINT uiNLAvg = 1000; // average noise level UINT uiNLMax = 1500; // maximum noise level SetData (FEDM_ISC_TMP_NOISE_LEVEL_MIN, uiNLMin); SetData (FEDM_ISC_TMP_NOISE_LEVEL_AVG, uiNLAvg); SetData (FEDM_ISC_TMP_NOISE_LEVEL_MAX, uiNLMax); SendProtocol (0x6C);
[0x6D] Get Noise level	UINT uiNLMin = 0; // minimum noise level UINT uiNLAvg = 0; // average noise level UINT uiNLMax = 0; // maximum noise level SendProtocol (0x6D); GetData (FEDM_ISC_TMP_NOISE_LEVEL_MIN, &uiNLMin); GetData (FEDM_ISC_TMP_NOISE_LEVEL_AVG, &uiNLAvg); GetData (FEDM_ISC_TMP_NOISE_LEVEL_MAX, &uiNLMax);
[0x6E] Reader Diagnostic	UCHAR ucDiagMode = 1; // diagnostic mode SetData (FEDM_ISC_TMP_DIAG_MODE, ucDiagMode); SendProtocol (0x6E);
[0x6F] Base Antenna Tuning	SendProtocol (0x6F); // the Long-Range-Reader starts internally a special tuning mode // the Reader can left the mode only with a reset
[0x71] Set Output	// Example 1 from System Manual ID ISC.M01 SetData (FEDM_ISCM_TMP_OUT_OS, (UINT)0); // reset OS-Bytes SetData (FEDM_ISCM_TMP_OUT_OS_OUT1, (UCHAR)0x01); // activate Output 1 SetData (FEDM_ISCM_TMP_OUT_OS_LED_G, (UCHAR)0x10); // green LED off SetData (FEDM_ISCM_TMP_OUT_OS_LED_R, (UCHAR)0x01); // red LED on SetData (FEDM_ISCM_TMP_OUT_OS_BEEPER, (UCHAR)0x11); // beeper on alternating SetData (FEDM_ISCM_TMP_OUT_OSF, (UINT)0); // reset OSF-Bytes SetData (FEDM_ISCM_TMP_OUT_OSF_BEEPER, (UCHAR)0x01); // beeper at 4Hz SetData (FEDM_ISCM_TMP_OUT_OSTIME, (UINT)5); // 500ms active time for beeper and LEDs SetData (FEDM_ISCM_TMP_OUT_OUTTIME, (UINT)3); // output 1 active for 300ms SendProtocol (0x71);

[Control byte] Protocol	Example
[0x72] Set Output	<pre>// Example from the system manual ID ISC.LRU1000 SetData(FEDM_ISC_TMP_0x72_OUT_MODE, (UCHAR)0x00); // set mode to 0 SetData(FEDM_ISC_TMP_0x72_OUT_N, (UCHAR)0x03); // activate 3 outputs SetData(FEDM_ISC_TMP_0x72_OUT_NR_1, (UCHAR)0x01); // output 1 SetData(FEDM_ISC_TMP_0x72_OUT_TYPE_1, (UCHAR)0x00); // type: general output SetData(FEDM_ISC_TMP_0x72_OUT_MODE_1, (UCHAR)0x03); // alternating SetData(FEDM_ISC_TMP_0x72_OUT_FREQ_1, (UCHAR)0x01); // 4 Hz SetData(FEDM_ISC_TMP_0x72_OUT_TIME_1, (UINT)5); // 500 ms SetData(FEDM_ISC_TMP_0x72_OUT_NR_2, (UCHAR)0x01); // relais 1 SetData(FEDM_ISC_TMP_0x72_OUT_TYPE_2, (UCHAR)0x04); // type: relais SetData(FEDM_ISC_TMP_0x72_OUT_MODE_2, (UCHAR)0x02); // switching off SetData(FEDM_ISC_TMP_0x72_OUT_FREQ_2, (UCHAR)0x00); // unchanged SetData(FEDM_ISC_TMP_0x72_OUT_TIME_2, (UINT)2); // 200 ms SetData(FEDM_ISC_TMP_0x72_OUT_NR_3, (UCHAR)0x02); // relais 2 SetData(FEDM_ISC_TMP_0x72_OUT_TYPE_3, (UCHAR)0x04); // type: relais SetData(FEDM_ISC_TMP_0x72_OUT_MODE_3, (UCHAR)0x01); // switching on SetData(FEDM_ISC_TMP_0x72_OUT_FREQ_3, (UCHAR)0x00); // unchanged SetData(FEDM_ISC_TMP_0x72_OUT_TIME_3, (UINT)10); // 1000 ms SendProtocol((0x72);</pre>
[0x74] Get Input	<pre>// Example for ID ISC.LR bool bIn1 = false; // Input 1 bool bIn2 = false; // Input 2 bool bDip1 = false; // DIP switch 1 bool bDip2 = false; // DIP switch 2 bool bDip3 = false; // DIP switch 3 bool bDip4 = false; // DIP switch 4 SendProtocol((0x74); GetData(FEDM_ISC_TMP_INP_STATE_IN1, &bIn1); GetData(FEDM_ISC_TMP_INP_STATE_IN2, &bIn2); GetData(FEDM_ISC_TMP_INP_STATE_DIP1, &bDip1); GetData(FEDM_ISC_TMP_INP_STATE_DIP2, &bDip2); GetData(FEDM_ISC_TMP_INP_STATE_DIP3, &bDip3); GetData(FEDM_ISC_TMP_INP_STATE_DIP4, &bDip4);</pre>
[0x75] Adjust Antenna	<pre>UINT uiAntValue = 0; // antenna voltage SendProtocol((0x75); GetData(FEDM_ISCM_TMP_ANTENNA_VALUE, &uiAntValue);</pre>

[Control byte] Protocol	Example
[0x80] Read Configuration and [0x81] Write Configuration	<pre>// the example shows reading and resetting a reader configuration block UCHAR ucCfgAdr = 2; // address of the configuration block bool bEEProm = false; // configuration data from/into Reader EEPROM UCHAR ucBusAddress; // bus address of ISC.LR-Reader from Block 2 SetData(FEDM_ISC_TMP_READ_CFG, (UCHAR)0x00); // reset all SetData(FEDM_ISC_TMP_READ_CFG_ADR, ucCfgAdr); // set address SetData(FEDM_ISC_TMP_READ_CFG_LOC, bEEProm); // set memory location on EEPROM SendProtocol(0x80); // read configuration data GetData(FEDM_ISCLR_EE_COM_BUSADR, &ucBusAdr); // e.g. get bus address SetData(FEDM_ISC_TMP_WRITE_CFG, (UCHAR)0x00); // reset all SetData(FEDM_ISC_TMP_WRITE_CFG_ADR, ucCfgAdr); // set address SetData(FEDM_ISC_TMP_WRITE_CFG_LOC, bEEProm); // set memory location on EEPROM SendProtocol(0x81); // write back configuration data</pre>
[0x82] Save Configuration	<pre>SetData(FEDM_ISC_TMP_SAVE_CFG, (UCHAR)0x00); // reset all SetData(FEDM_ISC_TMP_SAVE_CFG_ADR, (UCHAR)0x00); // set address SetData(FEDM_ISC_TMP_SAVE_CFG_MODE, TRUE); // save all blocks SendProtocol(0x82); // save configuration data from RAM to EEPROM</pre>
[0x83] Set Default Configuration	<pre>SetData(FEDM_ISC_TMP_RESET_CFG, (UCHAR)0x00); // reset all SetData(FEDM_ISC_TMP_RESET_CFG_ADR, (UCHAR)0x02); // set address SetData(FEDM_ISC_TMP_RESET_CFG_LOC, FALSE); // select RAM SetData(FEDM_ISC_TMP_RESET_CFG_MODE, FALSE); // only Block 2 to default SendProtocol(0x83); // set configuration data of Block 2 in RAM to default</pre>
[0x85] Set System Timer	<pre>SetData(FEDM_ISCLR_TMP_TIME_H, (UINT)16); // 16 hours SetData(FEDM_ISCLR_TMP_TIME_M, (UINT)20); // 20 minutes SetData(FEDM_ISCLR_TMP_TIME_MS, (UINT)2000); // 2000 milliseconds SendProtocol(0x85); // set timer</pre>
[0x86] Get System Timer	<pre>UINT uiHour = 0; // hours UINT uiMinute = 0; // minutes UINT uiMilliSec = 0; // milliseconds SendProtocol(0x86); // read timer GetData(FEDM_ISCLR_TMP_TIME_H, &uiHour); // get hours GetData(FEDM_ISCLR_TMP_TIME_M, &uiMinute); // get minutes GetData(FEDM_ISCLR_TMP_TIME_MS, &uiMilliSec); // get milliseconds</pre>
[0x87] Set System Date	<pre>SetData(FEDM_ISC_TMP_DATE_CENTURY, (UINT)20); // 20. century SetData(FEDM_ISC_TMP_DATE_YEAR, (UINT)4); // year 04 in century SetData(FEDM_ISC_TMP_DATE_MONTH, (UINT)9); // September SetData(FEDM_ISC_TMP_DATE_DAY, (UINT)15); // 15. September SetData(FEDM_ISC_TMP_DATE_TIMEZONE, (UINT)0); // actually unused SetData(FEDM_ISC_TMP_DATE_HOUR, (UINT)12); // hours SetData(FEDM_ISC_TMP_DATE_MINUTE, (UINT)00); // minutes SetData(FEDM_ISC_TMP_DATE_MILLISECOND, (UINT)0); // milliseconds (incl. seconds) SendProtocol(0x87); // set date and time</pre>

[Control byte] Protocol	Example
[0x88] Get System Date	<pre> UCHAR ucCentury = 0; // century UCHAR ucYear = 0; // year in century UCHAR ucMonth = 0; // month UCHAR ucDay = 0; // day UCHAR ucTimezone = 0; // timezone (actually unused) UCHAR ucHour = 0; // hour UCHAR ucMinute = 0; // minute UINT uiMilliSec = 0; // milliseconds SendProtocol(0x88); // read date and time GetData(FEDM_ISC_TMP_DATE_CENTURY, &ucCentury); // century GetData(FEDM_ISC_TMP_DATE_YEAR, &ucYear); // year in century GetData(FEDM_ISC_TMP_DATE_MONTH, &ucMonth); // month GetData(FEDM_ISC_TMP_DATE_DAY, &ucDay); // day GetData(FEDM_ISC_TMP_DATE_TIMEZONE, &ucTimezone); // actually unused GetData(FEDM_ISC_TMP_DATE_HOUR, &ucHour); // hours GetData(FEDM_ISC_TMP_DATE_MINUTE, &ucMinute); // minutes GetData(FEDM_ISC_TMP_DATE_MILLISECOND, &uiMilliSec); // milliseconds </pre>
[0x8A] Read Configuration und [0x8B] Write Configuration	<pre> // the example shows reading and resetting a reader configuration block UINT uiCfgAdr = 2; // address of the configuration block UCHAR ucBusAddress; // bus address of ISC.LRU3000 from Block 1 SetData(FEDM_ISC_TMP_0x8A_READ_DEVICE, (UCHAR)0x02); // RF-Controller SetData(FEDM_ISC_TMP_0x8A_READ_BANK, (UCHAR)0x01); // bank Main SetData(FEDM_ISC_TMP_0x8A_READ_MODE, (UCHAR)0x00); // clear mode byte SetData(FEDM_ISC_TMP_0x8A_READ_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8A_READ_REQ_CFG_ADR, uiCfgAdr); // configuration address SetData(FEDM_ISC_TMP_0x8A_READ_REQ_CFG_N, (UCHAR)1); // 1 configurationblock SendProtocol(0x8A); // execute command // retrieve the bus address GetConfigPara(ReaderConfig::HostInterface::Serial::BusAddress, &ucBusAdr); // change parameters with SetConfigPara(ReaderConfig::,); SetData(FEDM_ISC_TMP_0x8B_WRITE_DEVICE, (UCHAR)0x02); // RF-Controller SetData(FEDM_ISC_TMP_0x8B_WRITE_BANK, (UCHAR)0x01); // bank Main SetData(FEDM_ISC_TMP_0x8B_WRITE_MODE, (UCHAR)0x00); // clear byte SetData(FEDM_ISC_TMP_0x8B_WRITE_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8B_WRITE_CFG_ADR, uiCfgAdr); // configuration address SetData(FEDM_ISC_TMP_0x8B_WRITE_CFG_N, (UCHAR)1); // 1 configurationblock SendProtocol(0x8B); // execute command </pre>
[0x8C] Set Default Configuration	<pre> SetData(FEDM_ISC_TMP_0x8C_RESET_DEVICE, (UCHAR)0x02); // RF-Controller SetData(FEDM_ISC_TMP_0x8C_RESET_BANK, (UCHAR)0x01); // bank Main SetData(FEDM_ISC_TMP_0x8C_RESET_MODE, (UCHAR)0x00); // clear byte SetData(FEDM_ISC_TMP_0x8C_RESET_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8C_RESET_CFG_ADR, (UINT)1); // configuration address SetData(FEDM_ISC_TMP_0x8C_RESET_CFG_N, (UCHAR)1); // 1 configuration block SendProtocol(0x8C); // execute command </pre>
[0xA0] Reader Login	<pre> UCHAR ucPW[] = {0x00, 0x00, 0x00, 0x00}; // password SetData(FEDM_ISCLR_TMP_READER_PW, ucPW, 4); // set password SendProtocol(0xA0); // send password to reader </pre>
[0xA2] Write Mifare Keys	<pre> UCHAR ucKey[6]; </pre>

[Control byte] Protocol	Example
	<pre>// get the Mifare-Key, e.g. from a text control SetData(FEDM_ISC_TMP_ISO14443A_KEY_TYPE, (UCHAR)0); SetData(FEDM_ISC_TMP_ISO14443A_KEY_ADR, (UCHAR)0); SetData(FEDM_ISC_TMP_ISO14443A_KEY, ucKey, 6)); SendProtocol(0xB0); // send Mifare-Key to the reader</pre>
[0xA3] Write DES/AES Keys	<pre>UCHAR ucKey[16]; // get the Key e.g from a text control SetData(FEDM_ISC_TMP_0xA3_MODE, (UCHAR)0); // Location: RAM SetData(FEDM_ISC_TMP_0xA3_KEY_INDEX, (UCHAR)0); SetData(FEDM_ISC_TMP_0xA3_AUTHENTICATE_MODE, (UCHAR)5); // AES SetData(FEDM_ISC_TMP_0xA3_KEY_LEN, (UCHAR)16); SetData(FEDM_ISC_TMP_0xA3_KEY, ucKey, 16)); SendProtocol(0xA3); // send Key to the reader</pre>
[0xB0] ISO Mandatory and Optional Commands	<pre>// example shows the [0x01] Inventory // alternative: use of the method TagInventory and work with the TagHandler classes SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x01); // Inventory SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // no more-flag SendProtocol(0xB0); // the inventory data are in m_Table. Example for data accesses see 6.5.2. Examples for using the table with [0xB0] Commands</pre>
[0xB1] ISO15693 Customer and Proprietary Commands (TagHandler-Classes provide an easier API)	<pre>// example shows the [0xA2] Set EAS // all other 0xB1-Commands according to this example string sSnr; // STL-string for serial number UCHAR uclISOError = 0; // for ISO error code SetData(FEDM_ISC_TMP_B1_CMD, (UCHAR)0xA2); // Set EAS SetData(FEDM_ISC_TMP_B1_MFR, (UCHAR) FEDM_ISC_ISO_MFR_PHILIPS); // Manufacturer SetData(FEDM_ISC_TMP_B1_MODE, (UCHAR) FEDM_ISC_ISO_MODE_ADR); // addressed // ... get serial number from text field for example and save in sSnr SetData(FEDM_ISC_TMP_B1_REQ_UID, sSnr); int iStatus = SendProtocol(0xB1); if(iStatus == 0x95) { // get ISO error code GetData(FEDM_ISC_TMP_B1_ISO_ERROR, &uclISOError) }</pre>

[Control byte] Protocol	Example
<p>[0xB2] ISO14443 Special Commands</p> <p>[0x2B] ISO14443-4 Transponder Info</p> <p>(TagHandler-Classes provide an easier API)</p>	<pre> UCHAR ucFSCI = 0; UCHAR ucFWI = 0; UCHAR ucDSI = 0; UCHAR ucDRI = 0; UCHAR ucNad = 0; UCHAR ucCid = 0; SetData(FEDM_ISC_TMP_B2_CMD, (UCHAR)0x2B); // ISO14443-4 Transponder Info int iStatus = SendProtocol(0xB2); if(iStatus == 0x00) { // get the table index of the selected transponder int iIdx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_IS_SELECTED, true); if(iIdx >= 0) { // get transponder data from table GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_FSCI, &ucFSCI) GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_FWI, &ucFWI) GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_DSI, &ucDSI) GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_DRI, &ucDRI) GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_NAD, &ucNad) GetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_CID, &ucCid) } } </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB0] Authent Mifare</p> <p>(TagHandler-Classes provide an easier API)</p>	<pre> UCHAR ucDBAdr = 0; // address of the first datablock to be accessed to the transponder UCHAR ucKeyType = 0; // keytype for authentication UCHAR ucKeyAdr = 0; // EEPROM address where the key is stored in the reader SetData(FEDM_ISC_TMP_B2_CMD, (UCHAR)0xB0); // Authent Mifare SetData(FEDM_ISC_TMP_B2_MODE, (UCHAR) FEDM_ISC_ISO_MODE_SEL); // selected SetData(FEDM_ISC_TMP_B2_REQ_DB_ADR, ucDBAdr); SetData(FEDM_ISC_TMP_B2_REQ_KEY_TYPE, ucKeyType); SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR, ucKeyAdr); SendProtocol(0xB2); </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB1] Authent my-d</p> <p>(TagHandler-Classes provide an easier API)</p>	<pre> UCHAR ucKeyAdrTag = 5; // address of the key in the transponder UCHAR ucKeyAdrSam = 2; // address of the key in the security authentication modul (SAM) UCHAR ucCntAdr = 3; // address of the authentication counter UCHAR ucAuthSeq = 0; // authentication sequence SetData(FEDM_ISC_TMP_B2_CMD, (UCHAR)0xB1); // Authent my-d SetData(FEDM_ISC_TMP_B2_MODE, (UCHAR) FEDM_ISC_ISO_MODE_SEL); // selected SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR_TAG, ucKeyAdrTag); SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR_SAM, ucKeyAdrSam); SetData(FEDM_ISC_TMP_B2_REQ_AUTH_COUNTER_ADR, ucCntAdr); SetData(FEDM_ISC_TMP_B2_REQ_KEY_AUTH_SEQUENCE, ucAuthSeq); SendProtocol(0xB2); </pre>

[Control byte] Protocol	Example
[0xB2] ISO14443 Special Commands [0x30] Mifare Value Commands (TagHandler-Classes provide an easier API)	<pre> UCHAR ucMFCmd = 0x01; // Mifare Command UCHAR DBAdr = 0x05; // address of datablock UCHAR ucOpValue[] = {0x00, 0x00, 0x00, 0x03}; // OP_VALUE UCHAR ucDestAdr = 0x05; // address of destination SetData(FEDM_ISC_TMP_B2_CMD, (UCHAR)0x30); // Mifare Value Commands SetData(FEDM_ISC_TMP_B2_MODE, (UCHAR) FEDM_ISC_ISO_MODE_SEL); // selected SetData(FEDM_ISC_TMP_B2_REQ_MF_CMD, ucMFCmd); SetData(FEDM_ISC_TMP_B2_REQ_DB_ADR, ucDBAdr); SetData(FEDM_ISC_TMP_B2_REQ_OP_VALUE, ucOpValue, 4); SetData(FEDM_ISC_TMP_B2_REQ_DEST_ADR, ucDestAdr); SendProtocol(0xB2); </pre>

6.4.4. Asynchronous tasks for relieving the load on applications

A recurring task of applications is inventorying transponders in the antenna field of the reader. Ideally this should run in the background and then tell the application when transponders are in the field or when the notification has arrived.

This is precisely the functionality you can implement using the **FEISC_StartAsyncTask** function. Internally a thread is started which waits for the reply protocol of the reader and provides the reply data to the application using a callback function.

Asynchronous tasks are defined for two types of applications: for inventory in host mode or for receiving Buffered-Read-Mode data in Notification Mode.

Asynchronous tasks can be specified for multiple Readers at the same time as long as they were given their own object in the DLL using **FEISC_NewReader**. Readers on an RS485 bus are problematic. In this case you can only “monitor” one Reader at a time, since they are all connected on the same interface.

The features of the tasks are described in the table below:

Task	TaskID	Remarks
One-time Inventory	FEDM_TASKID_FIRST_NEW_TAG	<p>A task can only started if the following option is integrated in the Reader's firmware: the Reader protocol [0xB0][0x01] Inventory must support an optional NOTIFY flag in its Mode byte.</p> <p>After receiving the Reader protocol within the specified time, the task automatically closes itself. If the time is exceeded, the callback function is invoked and the status 0x01 (No transponder in read field) send and the task ended. In case of error the task is always ended immediately and the callback function transmits the error code.</p> <p>Serial, USB and TCP/IP interfaces are supported, whereby the ports must be open before starting the task. Autonomous opening of the connection via TCP/IP by the Reader or a suitable converter for sending the data is not possible.</p> <p>Callback-Function in FEDM_TASK_INIT: cbFct1</p> <p>The response data are located in the ISOTable and can be retrieved with GetISOTableItem.</p>
Repeating Inventory	FEDM_TASKID EVERY_NEW_TAG	<p>The same conditions as for one-time inventory apply, with the following difference:</p> <p>Repeating inventory defines a cyclical task which can only be cancelled by FEDM_ISCReaderModule::CancelAsyncTask. A cycle corresponds to a one-time inventory and ends on a wait loop until the next cycle has been triggered by the application using FEDM_ISCReaderModule::TriggerAsyncTask. Application-side triggering ensures that an application has time for receiving and processing the inventory data.</p> <p>Callback-Function in FEDM_TASK_INIT: cbFct1</p> <p>The response data are located in the ISOTable and can be retrieved with GetISOTableItem.</p>

Receiving notifications	FEDM_TASKID_NOTIFICATION	<p>A task should only be started if the Notification Mode is integrated and activated in the Reader's firmware. Only TCP/IP communication is supported. Possible connection options are (see system manual for the Reader):</p> <ul style="list-style-type: none">- Temporary opening of the connection by the Reader for the duration of data transmission- Continuous opening of the connection by the Reader (in development)- Continuous opening of the connection by the host (in development) <p>The task defines an endless task which can only be cancelled using FEDM_ISCReaderModule::CancelAsyncTask or in case of error during the initialization phase is ended immediately after invoking the callback function.</p> <p>The task waits for reception of the Buffered-Read-Mode data and then invokes the callback function. After the callback function returns, data can immediately be received again by the Reader.</p> <p>In case of transmission errors the callback function is invoked with the error code and the receiving procedure then resumed. If the Keep-Alive option is activated (recommended), then the listener socket is closed automatically after a break of the network cable or after loss of power and is recovered again. This ensures the reliability of the network connection.</p> <p>Note: Depending on the Reader setting large quantities of data may be sent by the Reader in very short time intervals. Without use of a handshake procedure (see system manual for the Reader) data may be lost if the host is not appropriate for the quantity of notifications.</p> <p>Callback-Function in FEDM_TASK_INIT: cbFct1 and cbFct2</p> <p>The response data are located in the BRMTable and can be retrieved with GetBRMTableItem.</p>
-------------------------	--------------------------	---

The internal behavior is determined essentially by the structure **FEDM_TASK_INIT**, which is sent using **FEDM_ISCReaderModule::StartAsyncTask**. Among other things it contains the necessary parameters for the callback function:

```
typedef struct _FEDM_TASK_INIT
{
    unsigned int    uiUse;           // specifies the Task (e.g. FEDM_TASKID_NOTIFICATION)
    unsigned int    uiFlag;         // specifies the use of the union (e.g. FEDM_TASKCB2)
    void*          pAny;           // pointer to anything, which is reflected as the first parameter
                                // in the callback function (e.g. can be used to pass the object pointer)

    int            iConnectByHost;  // always 0: TCP/IP connection is initiated by reader
    char           cIPAdr[16];     // server ip address
                                // note: only for channel type FEISC_TASK_CHANNEL_TYPE_NEW_TCP
    int            iPortAdr;       // server or host port address
                                // note: only for channel type FEISC_TASK_CHANNEL_TYPE_NEW_TCP
    UINT           uiTimeout;      // timeout for asynchronous task in steps of 100ms or
                                // timeout for notification task in steps of 1s

    // only for authentication in notification mode
    bool           bCryptoMode;    // security mode on/off
    unsigned int    uiAuthentKeyLength; // authent key length
    unsigned char   ucAuthentKey[32]; // authent key

    // only for notification
    bool           bKeepAlive;     // if true, keep alive option will be enabled (recommended)
    unsigned int    uiKeepAliveIdleTime; // wait time in ms for first probe after connection is dropped down
                                // for Linux: time is rounded up to seconds
    unsigned int    uiKeepAliveProbeCount; // only for Linux: number of probes
                                // for Windows Server 2003, and XP it is fixed to 5 by Microsoft
                                // for Windows Vista and later it is fixed to 10 by Microsoft
    unsigned int    uiKeepAliveIntervalTime; // wait time in ms between probes
                                // for Linux: time is rounded up to seconds

    union
    {
        // for notification and inventory task
        void (*cbFct1)( void* pAny,           // [in] pointer to anything (from struct _FEISC_TASK_INIT)
                        int iError,          // [in] OK (=0), error code (<0) or status byte from reader (>0)
                        unsigned char ucCmd); // [in] reader command

        // only for notification task
        void (*cbFct2)( void* pAny,           // [in] pointer to anything (from struct _FEISC_TASK_INIT)
                        int iError,          // [in] OK (=0), error code (<0) or status byte from reader (>0)
                        unsigned char ucCmd, // [in] reader command
                        char* cIPAdr,       // [in] ip address of the reader
                        int iPortNr);       // [in] local port number which received the notification
    };

    union
    {
        int iNotifyWithAck; // 0: notification without acknowledge
                        // 1: notification with acknowledge
    };
} FEDM_TASK_INIT;
```

The core element of the structure is the *union* with the function pointers. Selection of the callback function is handled by the parameter *uiFlag*. The parameter *pAny* can be used for any data and is returned in the first parameter of the callback function. C++ programmers can thus have a pointer for the invoking object sent to the static declared callback function and in this way access class functions. *uiTimeout* defines the timeout for an inventory cycle or the maximal time for receiving a

notification protocol. The value depends on the specifications in the system manual for the reader for the protocol [0xB0][0x01] Inventory or in seconds for notification timeout.

The structure variables *cClientIP* and *iPortAdr* are intended only for the Notification task.

Important Note: the structure FEDM_TASK_INIT must always be initialized on application-side with 0 with a call of `memset(myTaskInit, 0, sizeof(FEDM_TASK_INIT));`

6.4.5. Serializing in XML-Format¹

The standardizing of XML (Extensible Markup Format) has resulted in a description language for documents which can be used independently of the computer language and operating system. It makes sense therefore to use this language to define the structure of a reader configuration file. In the following the contents of an XML file created with ISOStart is shown:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<OBID>
  <file-header>
    <document-type>Reader Configuration File</document-type>
    <document-version>1.0</document-version>
    <reader-family>ISC</reader-family>
    <reader-name>ID ISC.MR100</reader-name>
    <reader-type>74</reader-type>
    <host-address>192.168.3.3</host-address>
    <port-number>10001</port-number>
    <communication-mode>TCP</communication-mode>
    <program-name>ID ISOStart</program-name>
    <program-version>05.03.03</program-version>
    <fedm-version>01.08</fedm-version>
    <date>07/18/03</date>
    <time>11:13:28</time>
  </file-header>
  <data-array name="Reader EEPROM-Parameter" blocks="16" size="16">
    <CFG0 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG1 b0="00" b1="00" b2="08" b3="01" b4="00" b5="00" b6="00" b7="0A" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG2 b0="00" b1="20" b2="00" b3="25" b4="00" b5="04" b6="00" b7="2F" b8="0A" b9="64" b10="00"
      b11="00" b12="04" b13="00" b14="00" b15="00"/>
    <CFG3 b0="00" b1="39" b2="00" b3="07" b4="00" b5="00" b6="06" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG4 b0="00" b1="00" b2="00" b3="00" b4="09" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG5 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="04" b12="00" b13="00" b14="00" b15="00"/>
    <CFG6 b0="00" b1="00" b2="00" b3="01" b4="00" b5="00" b6="00" b7="0A" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG7 b0="02" b1="20" b2="2C" b3="01" b4="0D" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG8 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG9 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG10 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG11 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG12 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG13 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG14 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG15 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
  </data-array>
  <data-array name="Reader RAM-Parameter" blocks="16" size="16">
    <CFG0 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG1 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG2 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG3 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG4 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG5 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
  </data-array>
</OBID>
```

¹ available only if pre-processor definition **_FEDM_XML_SUPPORT** is set

```

b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG6 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG7 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG8 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG9 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG10 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG11 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG12 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG13 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG14 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG15 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
</data-array>
</OBID>

```

In addition to some header data, the tags `<data-array name="Reader EEPROM-Parameter" blocks="16" size="16">` and `<data-array name="Reader RAM-Parameter" blocks="16" size="16">` contain the reader parameters as hex values.

The serialize method can now be used to create this file or read the reader configuration of such a file and copy it to the internal memory EEData or RAMData. The prerequisite for generating the configuration file is that the reader configuration has already been read.

To create a reader configuration file, call:

```
Serialize(false, "c:\\tmp\\myreader.xml")
```

and to read the data from a reader configuration file call:

```
Serialize(true, "c:\\tmp\\myreader.xml")
```

6.5. Table for ISO Host Commands

Note: this chapter is only relevant, if your project does not use TagHandler classes.

The [0xB0] ISO Host Commands are used to exchange data with multiple transponders located in the antenna field of the Reader. The table `m_ISOTable` (protected) is implemented within the Reader class `FEDM_ISCReader` for structured storage of this transponder data, with the protocol data for a transponder administered as a table entry. If for example data from three transponders arrives in a protocol, three table entries are filled with the transponder data.

The table is preset size of 0. This means you must initialize it with the method `SetTableSize` before first using. The size is determined by the maximum number of transponders that can be located in the antenna field of the Reader at the same time.

The data are entered in the empty table always starting from Index 0. The (protected) variable `m_iSOTableLength` for the Reader class tells you how many entries are valid.

The [0x0B] [0x01] Inventory keeps appending new transponder data. Thus the number of valid table entries continues to grow until the application program invokes the method `ResetTable`, or until the maximum size of the table as defined by the method `SetTableSize` is reached. In general, therefore, you will delete the table before each new Inventory.

All ISO commands in Non-Addressed Mode always use only Table Index 0. By developing an application dedicated to this special mode, you can set the table size to 1 and thus save storage space.

The table data are always accessed using the methods `GetTableData` and `SetTableData` for the Reader class `FEDM_ISCReader`. `FEDM_ISC_ISO_TABLE` is always passed as parameter `uiTableID`. A list and explanation of the access constants in `uiDataID` is found in [7.3.4. Constants for uiDataID](#).

The method `FindTableIndex` is provided for finding certain table entries, using the series number for example. Other query methods are: `GetTableSize` and `GetTableLength`. `ResetTable` deletes the table. Optionally this also allows you to initialize all data fields entered with the parameter `uiDataFlags=FEDM_ISC_DATA_ALL` with a value of 0. All these methods are described in the section on the Reader class on page [6.1. FEDM_ISCReader](#).

Separate data buffers for data blocks from receive (`m_ucRxDB`) and data blocks from send protocols (`m_ucTxDB`) are integrated in the table `m_ISOTable`. This allows simple verification of the written data blocks with the read data blocks.

After a [0xB0] [0x25] Select the corresponding table entry is marked as selected (`m_bIsSelected` set to true). All following ISO commands in Selected Mode automatically look for this table entry and handle the data exchange. Only one table entry can be marked as selected, since only one transponder at a time can be selected in the antenna field of the Reader. Repeating Select with a different series number (UID) therefore automatically deselects the last selected table entry before the new table entry is marked as selected. A [0xB0] [0x26] Reset to Ready cancels the last selection both of the transponder and of the table.

The method

```
FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_IS_SELECTED, true)
```

can be used to get the table index currently marked as selected.

The storage space requirement for a table entry is 17496 bytes.

Important Note: The ISO-Table has no support for the [0xB1] ISO15693 Custom and Proprietary Commands.

6.5.1. Anomaly of the addressed mode

Most of the Host Commands can be used in the addressed mode. In this case the serial number – or unified identifier (UID) – is part of the send protocol. In former versions the library has only supported UIDs with a length of 8 byte. With an extension flag in the mode byte (UID_LF) different UID length are now possible. If the UID_LF flag is set, the length of the UID must be added to the send protocol.

The following example demonstrates the use of a different UID length in a [0xB0][0xB23] Read Multiple Blocks:

```
// set UID for addressed mode (up to 32 byte)
SetData(FEDM_ISC_TMP_B0_REQ_UID, sUid);
SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, ucUidLen); // number of byte in UID

SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x23); // Command Read Multiple Blocks
SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // clear mode byte
SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // addressed mode
SetData(FEDM_ISC_TMP_B0_MODE_UID_LF, true); // UID_LF flag
SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // request one data block
SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr); // set data block address

SendProtocol(0xB0); // communication wit reader/transponder
```

6.5.2. Examples for using the table with [0xB0] Commands

The following examples suggest how data exchange with the table FEDM_ISOTable is handled. For reasons of clarity the processing of the return values of the methods is omitted. This should however always be done in applications.

[Control byte] Protocol	Example
<p>[0x01] Inventory</p> <p>für HF-Transponder:</p> <ul style="list-style-type: none"> - Philips I-CODE1 - Texas Instruments Tag-it HF - ISO15693 - ISO14443A - ISO14443B - EPC (Electronic Product Code) - Philips I-CODE UID - Innovision Jewel - EPC Class1 Gen2 HF <p>für UHF-Transponder:</p> <ul style="list-style-type: none"> - ISO18006-6-B - EM4222 - EPC Class0/0+ - EPC Class1 Gen1 - EPC Class1 Gen2 	<pre> UCHAR ucTrType = 0; // for transponder type CString sSnr; // for serial number (for EPC too) string sHeader; // for EPC field Header string sDomain; // for EPC field DomainManager string sObject; // for EPC field ObjektClass string sEPC; // for EPC ("Header.DomainManager.ObjectClass.Serialnumber") SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x01); // Command Inventory SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // no more-flag // clear complete table content (with option FEDM_ISC_DATA_ALL) or // set table length to 0 (without option FEDM_ISC_DATA_ALL) ResetTable(FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_ALL); SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_ISO_TABLE); ++iCnt) { // get transponder type GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TRTYPE, &ucTrType); switch(ucTrType) { // HF-Transponder case 0x00: // Philips I-CODE1 case 0x01: // Texas Instruments Tag-it HF case 0x03: // ISO15693 case 0x04: // ISO14443A case 0x05: // ISO14443B case 0x07: // I-Code UID case 0x08: // Innovision Jewel case 0x09: // EPC Class1 Gen2 HF // UHF-Transponder case 0x81: // ISO18000-6-B case 0x83: // EM4222 case 0x84: // EPC Class1 Gen2 case 0x88: // EPC Class0/0+ case 0x89: // EPC Class1 Gen1 // get the serial number as a string GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); break; case 0x06: // EPC (Electronic Product Code) // get the EPC fields... GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC_HEADER, sHeader); GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC_DOMAIN, sDomain); GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC_OBJECT, sObject); GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC_SNR, sSnr); } } </pre>

[Control byte] Protocol	Example
	<pre> // ...or get the complete EPC as a string GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC, sEPC); break; } } </pre>
[0x02] Stay Quiet	<pre> string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x02); // Command Stay Quiet SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>
[0x22] Lock Multiple Blocks	<pre> /* <u>Attention</u>: you lock the data blocks forever!! string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // first find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x22); // Command Lock Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // lock one data block SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, (UCHAR)0x00); // set data block address SendProtocol(0xB0); // communication with reader/transponder </pre>
[0x23] Read Multiple Blocks (normal address mode)	<pre> UCHAR ucDB[32]; // buffer for a data block (max. block size 32) UCHAR ucDBAdr = 5; // Data block address 5 string sSnr; // for serial number // ... get serial number from text field for example // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x23); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // read a data block SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr); // set data block address SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table // first find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // get size of the data block (block size) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, &ucBlockSize); // ... do something with block size // get a data block (ucDB will contain only ucBlockSize data bytes) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_RxDB, ucDBAdr, ucDB, 32); </pre>

[Control byte] Protocol	Example
	// ... do something with the data block
[0x23] Read Multiple Blocks (extended address mode)	<pre> UCHAR ucDB[32]; // buffer for a data block (max. block size 32) UINT uiDBAdr = 5; // Data block address 5 (range value: 0..65535) UCHAR ucPwLen; // length of Access Password string sPw; // Access Password string sSnr; // for serial number // ... get serial number from text field for example // ... get password from text field for example and save in sPw, ditto with length // set serial number for Addressed Mode (> 8 bytes allowed) SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, sSnr.length() / 2); // length of UIDin byte SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x23); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode-byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_EXT_ADR, true); // enable extended Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_UID_LF, true); // enable UID length value in protocol SetData(FEDM_ISC_TMP_B0_REQ_BANK, (UCHAR)0x00); // reset memory bank SetData(FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR, (UCHAR)0x03); // memory bank UserMem SetData(FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG, true); // enable Access Password SetData(FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH, ucPwLen); // set length of password SetData(FEDM_ISC_TMP_B0_ACCESS_PW, sPw); // password SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT, uiDBAdr); // set data block address SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // read one data block SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table // first find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // get size of the data block (block size) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, &ucBlockSize); // ... mach was mit der Blocksize // hole einen Datenblock (ucDB wird nur ucBlockSize Datenbytes enthalten) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_RxDB, uiDBAdr, ucDB, 32); // ... do something with block size </pre>

[Control byte] Protocol	Example
<p>[0x24] Write Multiple Blocks</p> <p>(normal address mode)</p>	<pre> /* the example shows the [0x24] Write Multiple Blocks. You must first have performed an [0x01] Inventory, if addressed mode is used. Caution: if you have still not performed [0x23] Read Multiple Blocks, the block size will be preset to 4. But if the transponder in the active zone supports another block size, this must first be set in the table for this transponder! You can use GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) for example to check whether the block size has already been read with [0x23] Read Multiple Blocks. If the block size is known (e.g. 8) and only writing to the transponder is necessary the block size must be set in the table with: SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); */ UCHAR ucDB[32]; // buffer for a data block (max. block size 32) UCHAR ucDBAdr = 5; // data block address 5 string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // ... get data block from a text field for example and save in ucDB[] // find table index for the serial number int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x24); // Command Write Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr); // set data block address SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // write a data block // set block size to 8 SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, 8); // write a data block to the table SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, ucDBAdr, ucDB, 8); SendProtocol(0xB0); // communication with reader/transponder </pre>

[Control byte] Protocol	Example
<p>[0x24] Write Multiple Blocks (extended address mode)</p>	<pre> /* the example shows the [0x24] Write Multiple Blocks. You must first have performed an [0x01] Inventory, if addressed mode is used. Caution: if you have still not performed [0x23] Read Multiple Blocks, the block size will be preset to 4. But if the transponder in the active zone supports another block size, this must first be set in the table for this transponder! You can use GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) for example to check whether the block size has already been read with [0x23] Read Multiple Blocks. If the block size is known (e.g. 8) and only writing to the transponder is necessary the block size must be set in the table with: SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); */ UCHAR ucDB[32]; // buffer for a data block (max. block size 32) UINT uiDBAdr = 5; // data block address 5 UCHAR ucPwLen; // length of Access Password string sPw; // Access Password string sSnr; // serial number // ... get serial number from text field for example and save in sSnr // ... get password from text field for example and save in sPw, ditto with length // ... get data block from a text field for example and save in ucDB[] // find table index for the serial number int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, sSnr.length() / 2); // length of UID in byte SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x24); // Command Write Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_EXT_ADR, true); // enable extended Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_UID_LF, true); // enable UID length value in protocol SetData(FEDM_ISC_TMP_B0_REQ_BANK, (UCHAR)0x00); // reset memory bank SetData(FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR, (UCHAR)0x03); // memory bank UserMem SetData(FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG, true); // enable Access Password SetData(FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH, ucPwLen); // set length of password SetData(FEDM_ISC_TMP_B0_ACCESS_PW, sPw); // password SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT, uiDBAdr); // set data block address SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // write a data block // set block size to 8 SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); // write a data block to the table SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, uiDBAdr, ucDB, 8); SendProtocol(0xB0); // communication with reader/transponder </pre>

[Control byte] Protocol	Example
[0x25] Select	<pre> string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x25); // Command Select SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>
[0x25] Select with Option Card Information for ISO14443 Transponder	<pre> CString sSnr; // for serial number UCHAR ucFormat = 0; // format byte from the response protocol // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x25); // Command Select SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_CINF, true); // CINF-Flag SendProtocol(0xB0); // communication with reader/transponder // the format byte is stored in TmpData GetData(FEDM_ISC_TMP_B0_RSP_FORMAT, &ucFormat); // format byte // the Card Information is stored in TmpData beginning at index 2048 UCHAR* ucCardInfo = &TmpData[2048]; // address of the first byte </pre>
[0x26] Reset to Ready	<pre> string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x26); // Command Reset to Ready SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>

[Control byte] Protocol	Example
[0x27] Write AFI	<pre> string sSnr; // for serial number UCHAR ucAFI = 0; // for AFI // ... get serial number from text field for example and save in sSnr // ... get AFI from edit control and in ucAFI // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); // find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // write AFI to the table SetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_AFI, ucAFI); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x27); // Command Write AFI SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>
[0x28] Lock AFI	<pre> string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x28); // Command Lock AFI SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>
[0x29] Write DSFID	<pre> string sSnr; // for serial number UCHAR ucDSFID = 0; // for DSFID // ... get serial number from text field for example and save in sSnr // ... get DSFID from edit control for example and save in ucDSFID // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); // find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // write DSFID to the table SetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_DSFID, ucDSFID); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x29); // Command Write DSFID SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder </pre>

[Control byte] Protocol	Example
[0x2A] Lock DSFID	<pre> string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x2A); // Command Lock DSFID SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x2B] Get System Information	<pre> UCHAR ucDSFID = 0; // for DSFID UCHAR ucAFI = 0; // for AFI UCHAR ucMemSize[] = {0, 0}; // for Memory-Size UCHAR ucICRef = 0; // for IC-Reference string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x2B); // Command Get System Information SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table // find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // get AFI from the table GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_AFI, &ucAFI); // ... do something with the AFI // ... get the other data values from the table </pre>
[0x2C] Get Multiple Block Security Status	<pre> UCHAR ucSecStatus; // for Security Status string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x05); // 5 data blocks SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, (UCHAR)0x00); // set first data block address SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x2C); // Command Get Multiple Block // Security Status SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table // find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // get the Security Status for data blocks 0..4 for(int iCnt=0; iCnt<5; ++iCnt) </pre>

[Control byte] Protocol	Example
	<pre> { GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SEC_STATUS, iCnt, &ucSecStatus, 1); // ... do something with the ucSecStatus } </pre>
<p>[0xA0] Read Config Block</p> <p>only for I-Code 1</p>	<pre> UCHAR ucCB[4]; // buffer for a data block (block size is always 4) UCHAR ucCBAAdr = 0; // data block address 0 string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0xA0); // Command Read Configuration Block SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, ucCBAAdr); // set data block address SendProtocol(0xB0); // communication with reader/transponder // all transponder data are contained in the table // find table index for the serial number int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // get the data block GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_RxCB, ucCBAAdr, ucCB, 4); // ... do something with the data block </pre>
<p>[0xA1] Write Config Block</p> <p>only for I-Code 1</p>	<pre> /* <u>Caution</u>: You can modify a transponder in that way, that he will be unusable for every time.!! UCHAR ucCB[4]; // buffer for a data block (block size is always 4) UCHAR ucCBAAdr = 0; // data block address 0 string sSnr; // for serial number // ... get serial number from text field for example and save in sSnr // ... get data block from a text field for example and save in ucCB[] // find table index for the serial number int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // set serial number for Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0xA1); // Command Write Multiple Block SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // reset Mode byte SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed mode SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, ucCBAAdr); // set data block address // write the data block into the table SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxCB, ucCBAAdr, ucCB, 4); SendProtocol(0xB0); // communication with reader/transponder </pre>

6.5.3.Examples for using the table with [0xB3] Commands

[Steuerbyte] Protokoll	Beispiel
<p>[0x18] Kill</p> <p>for UHF-Transponder: - EPC Class1 Gen1 - EPC Class1 Gen2</p>	<pre> /* <u>Attention</u>: this ISO Command destroys the transponder !! string sEpc; // for EPC string sPw; // for Kill Password unsigned char ucEpcLen = 0; // length of EPC in byte unsigned char ucPwLen = 0; // length of Kill Password // ... get serial number from text field for example // ... get password from text field for example, ditto with length // find table index for the EPC int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // get length of EPC GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR_LEN, &ucEpcLen); // set EPC for addressed mode SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // length of EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x18); // Command Kill SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // clear mode byte SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_UID_LF, true); // from 8 different UID length SetData(FEDM_ISC_TMP_B3_KILL_PW_LENGTH, ucPwLen); // length of password SetData(FEDM_ISC_TMP_B3_KILL_PW, sPw); // password SendProtocol(0xB3); // communication with reader/transponder </pre>
<p>[0x22] Lock Multiple Blocks</p> <p>for UHF-Transponder: - EPC Class1 Gen1 - EPC Class1 Gen2</p>	<pre> /* <u>Attention</u>: you lock the data blocks forever!! string sEpc; // for EPC string sLockData; // for optional lock data string sPw; // for optional access password unsigned char ucEpcLen = 0; // length of EPC in byte unsigned char ucTrType = 0; // transponder typ unsigned char ucLockDataLen = 0; // length of lock data in byte unsigned char ucPwLen = 0; // length of optional access password in byte // ... get serial number from text field for example // ... get lock data from text field for example, ditto with length // ... get password from text field for example, ditto with length // find table index for the EPC int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // get length of EPC GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR_LEN, &ucEpcLen); // get transponder type GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TRTYPE, &ucTrType); // set EPC for addressed mode SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // length of EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x22); // Command Lock SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // clear mode byte SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_UID_LF, true); // from 8 different UID length SetData(FEDM_ISC_TMP_B3_REQ_TR_TYPE, ucTrType); // transponder type </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre> SetData(FEDM_ISC_TMP_B3_LOCK_DATA_LENGTH, ucLockDataLen); // length of lock data SetData(FEDM_ISC_TMP_B3_LOCK_DATA, sLockData); // lock data SetData(FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH, ucPwLen); // length of password if(ucPwLen > 0) SetData(FEDM_ISC_TMP_B3_ACCESS_PW, sPw); // password SendProtocol(0xB3); // communication with reader/transponder </pre>
<p>[0x24] Write Multiple Blocks</p> <p>for UHF-Transponder: - EPC Class1 Gen2</p>	<pre> /* the example shows the [0x24] Write Multiple Blocks. You must first have performed an [0x01] Inventory, if addressed mode is used. Caution: if you have still not performed [0x23] Read Multiple Blocks, the block size will be preset to 4. But if the transponder in the active zone supports another block size, this must first be set in the table for this transponder! You can use GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) for example to check whether the block size has already been read with [0x23] Read Multiple Blocks. */ UCHAR ucDB[32]; // buffer for a data block (max. block size 32) UCHAR ucDBAdr = 5; // data block address 5 string sSnr; // for serial number string sPw; // for optional access password unsigned char ucEpcLen = 0; // length of EPC in byte unsigned char ucPwLen = 0; // length of optional access password in byte // ... get EPC from text field for example and save in sEPC // ... get password from text field for example and save in sPw, ditto with length // ... get data block from a text field for example and save in ucDB[] // find table index for the serial number int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // set EPC for addressed mode SetData(FEDM_ISC_TMP_B3_REQ_UID, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // length of EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x24); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // clear mode byte SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_UID_LF, true); // from 8 different UID length SetData(FEDM_ISC_TMP_B3_MODE_EXT_ADR, true); // 2 byte DB address SetData(FEDM_ISC_TMP_B3_BANK, (UCHAR)0); // reset memory bank SetData(FEDM_ISC_TMP_B3_BANK_BANK_NR, (UCHAR)0x01); // EPC memory bank SetData(FEDM_ISC_TMP_B3_BANK_ACCESS_FLAG, true); // with access password SetData(FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH, ucPwLen); // set length of password SetData(FEDM_ISC_TMP_B3_ACCESS_PW, sPw); // password SetData(FEDM_ISC_TMP_B3_REQ_DB_ADR_EXT, (UINT)0); // set data block address SetData(FEDM_ISC_TMP_B3_REQ_DBN, (UCHAR)0x06); // write six data blocks SetData(FEDM_ISC_TMP_B3_REQ_DB_SIZE, (UCHAR)0x02); // blocksize for protocol // set same blocksize in table SetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)2); // set data blocks in table for(int iAdr=0; iAdr<6; ++iAdr) SetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, iAdr, &ucDB[iAdr*2], 2); SendProtocol(0xB3); // communication with reader/transponder </pre>

6.6. Table for Buffered Read Mode

The [0x21] Read Buffer (only for Long-Range-Reader ID ISC.LRxxx) and respectively [0x22] Read Buffer permits data from multiple transponders located in the antenna field of the Reader to be read. The table `m_BRMTable` (protected) is implemented within the Reader class `FEDM_ISCReader` for structured storage of this transponder data, with the protocol data for a transponder administered as a table entry. If for example data from three transponders arrives in a protocol, three table entries are filled with the transponder data.

The table is preset size of 0. This means you must initialize it with the method `SetTableSize` before first using. The size is determined by the maximum number of transponders that can be located in the antenna field of the Reader at the same time.

The data are entered in the empty table always starting from Index 0. This means that each [0x21] Read Buffer ([0x22] Read Buffer) causes the old table data to be overwritten. The (protected) variable `m_iBRMTableLength` for the Reader class tells you how many entries are valid.

The table data are always accessed using the methods `GetTableData` and `SetTableData` for the Reader class `FEDM_ISCReader`. `FEDM_ISC_BRM_TABLE` is always passed as parameter `uiTableID`. A list and explanation of the access constants in `uiDataID` is found in [7.3.4. Constants for uiDataID](#).

The method `FindTableIndex` is provided for finding certain table entries, using the series number for example. Other query methods are: `GetTableSize` and `GetTableLength`. `ResetTable` deletes the table. Optionally this also allows you to initialize the data fields entered in the parameter `uiDataFlags` with a value of 0. All these methods are described in the section on the Reader class on page [6.1. FEDM_ISCReader](#).

The storage space requirement for a table entry is 1104 bytes.

6.6.1. Examples for using of the table

The following examples suggest how data exchange with the table `FEDM_BRMTable` is handled. For reasons of clarity the processing of the return values of the methods is omitted. This should however always be done in applications.

[Control byte] Protocol	Example
[0x21] Read Buffer	<pre>// the example shows reading of data sets with serial number, data block and timer value UCHAR ucDataSets = 1; // number of data sets requested UCHAR ucRecSets = 0; // number of data sets in the protocol UCHAR ucDB[4]; // buffer for a data block UINT uiTimer = 0; // for timer value __int64 i64Snr = 0; // for serial number BOOL bSNR = FALSE; // flag for serial number in data set BOOL bDB = FALSE; // flag for data block in data set BOOL bTimer = FALSE; // flag for timer in data set</pre>

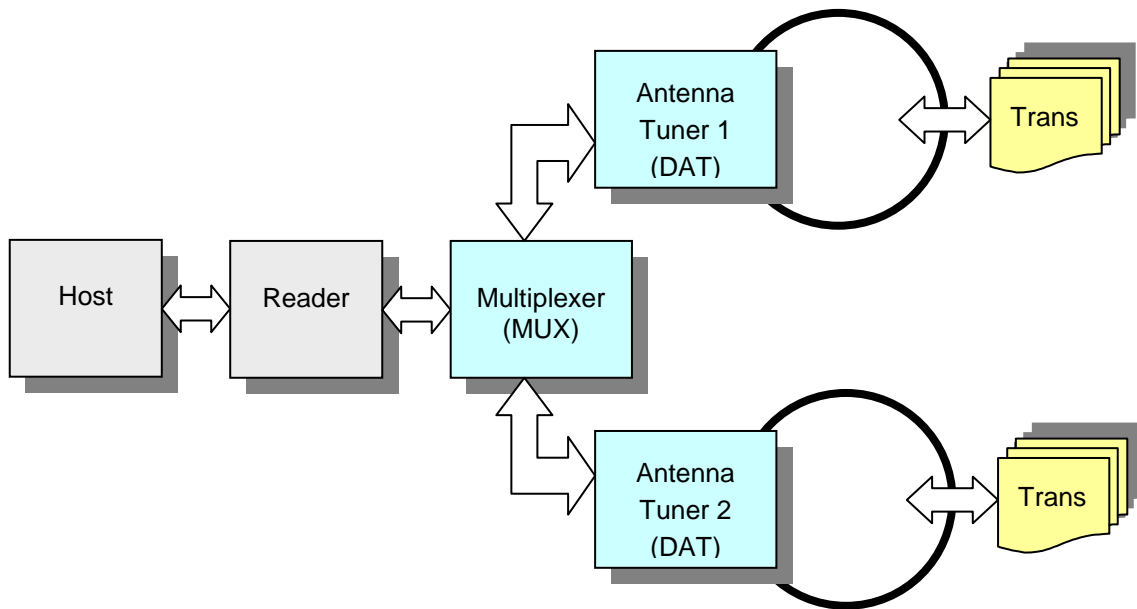
[Control byte] Protocol	Example
	<pre> SetData(FEDM_ISCLR_TMP_BRM_SETS, ucDataSets); SendProtocol(0x21); // read data from transponder with Buffered Read Mode GetData(FEDM_ISCLR_TMP_BRM_TRDATA_SNR, &bSNR); GetData(FEDM_ISCLR_TMP_BRM_TRDATA_DB, &bDB); GetData(FEDM_ISCLR_TMP_BRM_TRDATA_TIME, &bTimer); GetData(FEDM_ISCLR_TMP_BRM_RECSETS, &ucRecSets); // all transponder data are contained in the table for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_BRM_TABLE); iCnt++) { if(bSNR) // get serial number GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_SNR, &i64Snr); if(bDB) // get Data Block 1 GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_RxDB, 1, ucDB, 4); if(bTIMER) // get timer value GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_TIMER, &uiTimer); } </pre>
[0x22] Read Buffer	<pre> // the example shows reading of data sets with serial number, data block, date and timer value and // antenna number UINT uiDataSets = 1; // number of data sets requested UINT uiRecSets = 0; // number of data sets in the protocol UCHAR ucAnt; // antenna number UCHAR ucDate[5]; // buffer for date field UCHAR ucInput; // for input byte UCHAR ucStatus; // for status byte UCHAR ucSize; // blocksize of one datablock UINT uiDBN = 0; // number of datablocks UINT uiTimer = 0; // for timer value string sSnr = 0; // for serial number string sDB; // for datablocks BOOL bSNR = FALSE; // flag (in TR-DATA1) for serial number in data set BOOL bDB = FALSE; // flag (in TR-DATA1) for datablocks in data set BOOL bANT = FALSE; // flag (in TR-DATA1) for antenna number in data set BOOL bDate = FALSE; // flag (in TR-DATA1) for date in data set BOOL bTime = FALSE; // flag (in TR-DATA1) for time in data set BOOL bExt = FALSE; // EXTENSION flag (in TR-DATA1): signals, that a second TR-DATA // byte is following, where additional flags continues the definition of a // data set BOOL bInput = FALSE; // flag (in TR-DATA2) for input and status byte in data set SetData(FEDM_ISC_TMP_ADV_BRM_SETS, uiDataSets); SendProtocol(0x22); // read data from transponder with Buffered Read Mode GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_SNR, &bSNR); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_DB, &bDB); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_ANT, &bANT); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_TIME, &bTime); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_DATE, &bDate); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1_EXT, &bExt); GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA2_INPUT, &bInput); GetData(FEDM_ISC_TMP_ADV_BRM_RECSETS, &uiRecSets); // all transponder data are contained in the table </pre>

[Control byte] Protocol	Example
	<pre> for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_BRM_TABLE); iCnt++) { if(bSNR) // get serial number GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_SNR, sSnr); if(bDB) // get all datablocks { // get number of datablocks GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_DBN, &uiDBN); // get blocksize GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, &ucSize); // get datablocks for(int i=0; i<uiDBN; ++i) { GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_RxDB, i, sDB); // do anything with the datablocks } } if(bANT) // get antenna number GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_ANT_NR, ucAnt); if(bTime) // get time value GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_TIMER, &uiTimer); if(bDate) // get date GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_DATE, ucDate, 5); if(bExt && bInput) // get input and status byte { GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_INPUT, ucInput); GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_STATE, ucStatus); } } </pre>

6.7. FEDM_ISCFunctionUnit

The class **FEDM_ISCFunctionUnit** represents an external function unit integrated in the antenna cable of the reader. The class has no base class. For a deeper understanding of the possibilities of function units you should read the system manual H30701-xe-ID-B (HF) or H80302-xe-ID-B (UHF). Additional information can be found in the installation guides of the function units.

In consideration of the fact that a function unit needs always a reader as a communication bridge, the class **FEDM_ISCFunctionUnit** can only be instantiated if a reader object of type **FEDM_ISCReader** is previously created.



The picture above demonstrates also that external function units are arranged in hierarchical order. The function unit class pattern this topology with a list of successors of type **FEDM_ISCFunctionUnit**. Beginning with the first function unit after the reader one can traverse through the tree of function units.

By default, the first function unit – normally a Multiplexer – manages the pointers of the succeeding and dynamically created function units. This has the advantage, that the destructor of the first function unit frees the memory of the complete object tree. Is this behaviour undesired, because of working with static objects, then this option must be disabled by use of the method **SetManageChildMode(false)**.

6.7.1. Constructor

A pointer to the reader object of type **FEDM_ISCReader** and the type of the function unit are passed with the constructor. The reader object must represent a reader which is physically connected with the function unit.

6.7.2. Implemented data containers

Data container	Description
TmpData	For general temporary protocol data

The size of the data containers is determined statically in the class constructor. All data containers are initialized in the constructor with 0x00.

6.7.3. Implemented lists

List	Description
m_ChildList	List with pointer to function units which are physically connected at the output or outputs of the function unit.

The use of the list is only possible if the function unit supports the connection of successive function units. The multiplexer ID ISC.ANT.MUX is e. g. such a type of function unit.

6.7.4. Methods (public)

Method	Description
SendProtocol	The central communication method. For more details see section 6.7.6. Examples for using the method SendProtocol
AddChild	Method to add a successive function unit to the list.
DelChild	Method to remove a successive function unit from the list.
DelChildList	Method to remove all successive function units from the list.
GetChild	Method to get a pointer to a successive function unit from the list.
SetFUType	Method to set the type of the function unit. This action removes all successive function units from the list.
GetFUType	Get the type of the function unit.
SetManageChildMode	Set the management mode for childs. By default, the parent destroys the childs. If the application should control the free of memory, the use of this method with the parameter <code>bDeleteInternal=false</code> disables the internal management.
GetData	Overloaded method for reading a parameter value from a data container. The invocation is passed to the class <code>FEDM_Base</code> after the data container type (memory type constant) has been determined from the access constant. <code>GetData</code> supports the following data types: <code>bool</code> , <code>BOOL</code> , <code>UCHAR</code> , <code>UCHAR-Array</code> , <code>UNIT</code> , <code>__int64</code> , <code>CString</code> resp. <code>AnsiString</code> , <code>STL-string</code> and <code>C-string</code> .
SetData	Overloaded method for writing a parameter value to a data container. The invocation is passed to the class <code>FEDM_Base</code> after the data container type (memory type constant) has

Method	Description
	been determined from the access constant. SetData supports the following data types: bool, BOOL, UCHAR, UCHAR-Array, UNIT, __int64, CString resp. AnsiString, STL-string and C-string..
GetLastError	Returns the last error code.
GetLastStatus	Returns the last status byte.
GetErrorText	Gets a text corresponding to a sent error code. The error code may also come from the function collection sector ID FEISC or the underlying communication library.
GetStatusText	Gets a text corresponding to the sent status byte.

6.7.5.Important initializations

Before using the protocol method for the first time, some initializing must be performed:

Function Unit	Set of address
ID ISC.DAT	SetPara(FEDM_ISC_FU_TMP_DAT_ADR, ucAdr)
ID ISC.ANT.MUX	SetPara(FEDM_ISC_FU_TMP_MUX_ADR, ucAdr)
ID ISC.ANT.UMUX	

6.7.6.Examples for using the method SendProtocol

SendProtocol is of key importance to the protocol transfer. For this reason an example is shown for each control byte which is intended to show which data are to be stored in data containers with which access constants before each protocol transfer and which data are available after the protocol transfer.

All the access constants are listed in the file FEDM_ISCFunctionUnitID.h and should be studied carefully in conjunction with the explanation of the protocol data found in the system manual.

For reasons of clarity the processing of the return values of the methods is not shown here. Of course it should always be included in applications.

[Control byte] Protocol		Example
ID ISC.DAT	[0xC0] Get Firmware Version	UCHAR ucFirmware[7]; // buffer for firmware informations SendProtocol (0xC0); GetData (FEDM_ISC_FU_TMP_SOFTVER, ucFirmware, 7);
	[0xC1] CPU Reset	SendProtocol (0xC1);
	[0xC2] Set Capacities	SetData (FEDM_ISC_FU_TMP_DAT_ANT_VAL_C1, (UCHAR)0xAB); // capacity 1 SetData (FEDM_ISC_FU_TMP_DAT_ANT_VAL_C2, (UCHAR)0x9F); // capacity 2 SendProtocol (0xC2);
	[0xC3] Get Antenna Values	UCHAR ucAntValues[6]; // buffer for tuning values SendProtocol (0xC3); GetData (FEDM_ISC_FU_TMP_DAT_ANT_VAL, ucAntValues, 6);
	[0xC4] Set Outputs	SetData (FEDM_ISC_FU_TMP_DAT_OUT, (UCHAR)1); // switch output 1 SendProtocol (0xC4);
	[0xC5] Re-Tuning	SendProtocol (0xC5);
	[0xC6] Start Tuning	SendProtocol (0xC6);
	[0xC8] Store Settings	SendProtocol (0xC8);
	[0xC9] Detect	SendProtocol (0xC9);
	[0xCA] Set Address	UCHAR ucAdr = 2; // new address SetData (FEDM_ISC_FU_TMP_DAT_NEW_ADR, ucAdr); // new address for function unit SendProtocol (0xCA); // new address becomes valid SetData (FEDM_ISC_FU_TMP_DAT_ADR, ucAdr); // set new address for communication
	[0xCB] Set Mode	SetData (FEDM_ISC_FU_TMP_DAT_MODE, (UCHAR)1); // mode 1 SendProtocol (0xCB);

ID ISC.ANT.MUX	[0xDC] Detect	SendProtocol (0xDC);
	[0xDD] Select Channel	SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH1, (UCHAR)1); // set output 1 for input 1 SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH2, (UCHAR)8); // set output 8 for input 2 SendProtocol (0xDD);
	[0xDE] CPU Reset	SendProtocol (0xDE);
	[0xDF] Get Firmware Version	UCHAR ucFirmware[7]; // buffer for firmware informations SendProtocol (0xDF); GetData (FEDM_ISC_FU_TMP_SOFTVER, ucFirmware, 7);
ID ISC.ANT.UMUX	[0xDC] Detect/Get Power	UCHAR ucPower[5]; // buffer for power information SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // always 0 SendProtocol (0xDC); GetData (FEDM_ISC_FU_TMP_UMUX_POWER, ucPower, 5); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);
	[0xDD] Select Channel	SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // always 0 SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH1, (UCHAR)1); // select output 1 SendProtocol (0xDD); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);
	[0xDE] CPU Reset	SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // always 0 SendProtocol (0xDE); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);
	[0xDF] Get Firmware Version	UCHAR ucFirmware[7]; // buffer for firmware information SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // immer auf 0 SendProtocol (0xDF); GetData (FEDM_ISC_FU_TMP_SOFTVER, ucFirmware, 7); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);

6.8. FedmlscPeopleCounter

The class **FedmlscPeopleCounter** represents an external unit of type People-Counter ID **ISC.ANT1690/600-GPC** resp. ID **ISC.ANT1700/740-GPC** connected at RS485 port at the Reader. The class is derived from the base class **FedmlscPeripheralDevice**. For further information please refer to the People-Counter's system manual H01011-0e-ID-B. Additional information can also be found in the mounting instructions.

Up to three People-Counter can be connected to each Reader. They can be identified by their individual bus address. After power-on or after the command [0x64] System Reset (ACC) the Reader starts a detection process. An application can request all detected People-Counter with the method **FEDM_ISCReaderModule::ReadReaderInfo()** or with command [0x66] Reader Info with mode 0x61. After that the method **FEDM_ISCReader::GetPeripheralDevices()** returns a sorted list of People-Counter objects of type **FedmlscPeripheralDevice** which must be casted to **FedmlscPeopleCounter**.

6.8.1. Methods (public)

Method	Description
SetOutput	Set 1, 2 or 3 digital outputs
SetCounter	Set of all 4 counters
GetCounter	Return of all 4 counter values

6.8.2. Example for using the class

The following example demonstrates the use of the class for Readers working in Host-Mode or Buffered-Read-Mode.

```

unsigned int uiCounter1 = 0;
unsigned int uiCounter2 = 0;
unsigned int uiCounter3 = 0;
unsigned int uiCounter4 = 0;

FEDM_PD_MAP* pPDMap = NULL;

FEDM_PD_MAP_ITOR itor;

FedmlscPeopleCounter* pPeopleCounter = NULL;

// request detected People-Counter
FEDM_ISC_READER_INFO* pInfo = m_Reader.GetReaderInfo(); // get pointer to info struct
if (! pInfo->blsMode0x61Read)
{
    m_Reader.SetData(FEDM_ISC_TMP_READER_INFO_MODE, (unsigned char)0x61)
    int back = m_Reader.SendProtocol(0x66);
    if(back)
        return; // some problems
}

pPDMap = m_Reader.GetPeripheralDevices();

// 1. important test

```



```

if(pPDMMap == NULL)
    return; // no People-Counter connected or not activated in theReader configuration

// People-Counter withbusaddress 1
itor = pPDMMap->find(1);

// 2. important test
if(itor == pPDMMap->end())
    return; // no People-Counter withbusaddress 1 found

// 3. important test
if(dynamic_cast<FedmlscPeopleCounter*>(itor->second) == NULL)
    return; // it's not a People-Counter class

pPeopleCounter = (FedmlscPeopleCounter*)itor->second;

// request counter values

pPeopleCounter->GetCounter(uiCounter1, uiCounter2, uiCounter3, uiCounter4);

```

Important Note: The sortedpointer listis managed by thereader class FEDM_ISCReader. Creation and destroying of instances of FedmlscPeripheralDevice or derivations are unter control ofthe reader class. Applications must not modify the sorted list nor free the memory allocated for a list member.

6.8.3.Example for automatic notification

If the Reader is working in Notification-Mode, the counter values can be sentby TCP/IP-Notification to theapplication.The Reader's configuration must be set accordingly: s. parameter inthe namespace

OperatingMode.NotificationMode.GatePeopleCounter.Transmission.Destination

The realization in anapplication is very easy and without use of the People-Counter class. Add atask in thefunction library FEISC and provide acallback function. The counter values will be supplied by the callback function.

If you want to combine the notification of counter values with the notification of tag data, an application must start two tasks in two different reader objects. For counter notification start the task as shown in the example below. For tag data notification use the method StartAsyncTask of reader class FEDM_ISCReaderModule.

```

BOOL CPeopleCounterSampleDlg::OnInitDialog()
{
    FEISC_TASK_INIT TaskInit;

    memset(&TaskInit, 0, sizeof(TaskInit)); // very important initialization

    TaskInit.cbFct2 = cbsFct; // callback fundction
    TaskInit.uiFlag = FEISC_TASKCB_2;
    TaskInit.pAny = this; // pAny is reflected as 1st parameter in callback function
    TaskInit.iPortNr = 10005; // listener port
    TaskInit.uiTimeout = 10; // timeout in s, for waiting of next part of a protocol
    TaskInit.uiChannelType = FEISC_TASK_CHANNEL_TYPE_NEW_TCP;
    TaskInit.bKeepAlive = true; // enabled keep-alive option is recommended
    TaskInit.uiKeepAliveIdleTime = 500;
    TaskInit.uiKeepAliveProbe = 5; // applicable only for Linux, ignored by Windows

```

```
TaskInit.uiKeepAliveIntervalTime = 500;
int iReaderHnd =FEISC_NewReader(0);

int iBack = FEISC_StartAsyncTask( iReaderHnd,
                                FEISC_TASKID_PEOPLE_COUNTER_EVENT,
                                &TaskInit,
                                NULL);
}

void CPeopleCounterSampleDlg::cbsFct(
    void* pAny,           // [in] pointer to anything (from struct _FEISC_TASK_INIT)
    int iReaderHnd,       // [in] reader handle of FEISC
    int iTaskID,          // [in] task identifier from FEISC_StartAsyncTask(..)
    int iError,           // [in] OK (=0), error code (<0) or status byte from reader (>0)
    unsignedchar ucCmd,   // [in] reader command
    unsignedchar* ucRspData, // [in] response data
    int iRspLen,          // [in] length of response data
    char* cRemotelP,      // [in] ip address of the reader
    int iLocalPort )      // [in] local port number which has received the notification
{
    if(pAny == NULL)
        return;

    if(ucCmd != 0x77)
        return;

    if(iRspLen < 17)
        return;

    ((CPeopleCounterSampleDlg*)pAny)->cbFct(ucRspData, iRspLen);
}

void CPeopleCounterSampleDlg::cbFct(unsignedchar* ucRspData, int iRspLen)
{
    unsignedint uiCnt1 = ucRspData[3] | ucRspData [2] << 8 | ucRspData[1] << 16 | ucRspData[0] << 24;
    unsignedint uiCnt2 = ucRspData[7] | ucRspData [6] << 8 | ucRspData[5] << 16 | ucRspData[4] << 24;
    unsignedint uiCnt3 = ucRspData[11] | ucRspData [10] << 8 | ucRspData[9] << 16 | ucRspData[8] << 24;
    unsignedint uiCnt4 = ucRspData[15] | ucRspData [14] << 8 | ucRspData[13] << 16 | ucRspData[12] << 24;

    m_sEntryCounter1.Format("%u", uiCnt1);
    m_sExitCounter1.Format("%u", uiCnt2);
    m_sDiff1.Format("%d", uiCnt1-uiCnt2);

    m_sEntryCounter2.Format("%u", uiCnt3);
    m_sExitCounter2.Format("%u", uiCnt4);
    m_sDiff2.Format("%d", uiCnt3-uiCnt4);

    ::PostMessage(this->GetSafeHwnd(), WM_USER_NEW_DATA, 0, 0);
}
```

7.Appendix

7.1.Supported OBID® Readers

Reader	Notes
ID ISC.M02	
ID ISC.MR/PR100	all communication ports
ID ISC.PRH100/PRH101 / PRH102	all communication ports
ID ISC.MR/PR101	all communication ports
ID ISC.MR102	all communication ports
ID ISC.PRH102	all communication ports
ID ISC.PRHD102	all communication ports
ID ISC.MR200	all communication ports
ID ISC.LR200	
ID ISC.LR1002	all communication ports
ID ISC.LR2000	all communication ports
ID ISC.LR2500-A	all communication ports
ID ISC.LR2500-B	all communication ports
ID ISC.MRU200	all communication ports
ID ISC.LRU1000	all communication ports
ID ISC.LRU2000	all communication ports
ID ISC.LRU3000	all communication ports
ID CPR.02	
ID CPR.M02	all communication ports
ID CPR.04	all communication ports
ID CPR30.xx	all communication ports
ID CPR40.xx	all communication ports
ID CPR44.xx	all communication ports
ID CPR46.xx	all communication ports
ID CPR47.xx	all communication ports
ID CPR50.xx	all communication ports
ID CPR52.xx	all communication ports
ID MAX50.xx	all communication ports

7.2. Supported Transponders

The support of transponders depends on the implemented reader firmware. Please refer to the system manual of the reader.

The list below collects the transponder types, which are well-established during the development time of the library.

Transponder	Value	Notes
I-CODE 1	0x00	HF-Transponder
Tag-it	0x01	HF-Transponder
ISO15693	0x03	HF-Transponder
ISO14443-A	0x04	HF-Transponder
ISO14443-B	0x05	HF-Transponder
EPC	0x06	HF-Transponder (EPC-Types 1..4)
I-CODE UID	0x07	HF-Transponder
Jewel	0x08	HF-Transponder
ISO 18000-3M3	0x09	HF-Transponder
STMicroelectronics SR176	0x0A	HF-Transponder
STMicroelectronics SRIxx	0x0B	HF-Transponder
Microchip MCRFxxx	0x0C	HF-Transponder
Innovatron (ISO 14443B')	0x10	HF-Transponder
ASK CTx	0x11	HF-Transponder
ISO18000-6-A	0x80	UHF-Transponder
ISO18000-6-B	0x81	UHF-Transponder
EM4222	0x83	UHF-Transponder
EPC Class1 Generation 2	0x84	UHF-Transponder
EPC Class0/0+	0x88	UHF-Transponder
EPC Class1 Generation 1	0x89	UHF-Transponder

7.3. TCP-Status

Information concerning the status can be found with the Internet when searching for *Transmission Control Protocol*

TCP-Status	Value
CLOSED	1
LISTEN	2
SYN_SENT	3
SYN_RCVD	4
ESTABLISHED	5
FIN_WAIT1	6
FIN_WAIT2	7
CLOSE_WAIT	8
CLOSING	9
LAST_ACK	10
TIME_WAIT	11

7.4.List of constants

All constants listed here are defined in FEDM_ISC.h or FEDM_ISCFunctionUnitID.h.

7.4.1. Internal Constants

Constant	Description
FEDM_ISC_MAX_EEDATA_MEM	Size of data container EEData
FEDM_ISC_MAX_RAMDATA_MEM	Size of data container RAMData
FEDM_ISC_MAX_TMPDATA_MEM	Size of data container TmpData
FEDM_ISC_BRM_TABLE_RxDB_SIZE	Size of data buffer for Receive-Datablocks for each transponder
FEDM_ISC_ISO_TABLE_TxDB_SIZE	Size of data buffer for Transmit Datablocks for each transponder
FEDM_ISC_ISO_TABLE_RxDB_SIZE	Size of data buffer for Receive Datablocks for each transponder
FEDM_ISC_ISO_TABLE_EPC_BANK_SIZE	Size of data buffer for Transmit Datablocks in EPC memory bank for each transponder (nur EPC Class1 Gen2)
FEDM_ISC_ISO_TABLE_TID_BANK_SIZE	Size of data buffer for Transmit Datablocks in TID memory bank for each transponder (nur EPC Class1 Gen2)
FEDM_ISC_ISO_TABLE_RES_BANK_SIZE	Size of data buffer for Transmit Datablocks in RESERVED memory bank for each transponder (nur EPC Class1 Gen2)
FEDM_ISC_ISO_TABLE_SEC_STATUS_SIZE	Number of Security-Bytes for each Transponder (1 Byte for each data block)
FEDM_ISC_FU_MAX_TMPDATA_MEM	Size of data container TmpData for FEDM_ISCFunctionUnit

7.4.2. General Constants

Constant	Description
FEDM_ISC_TYPE_...	Reader Type according to the protocol [0x65] Software Version
FEDM_ISC_NAME_...	Reader Name according to the readers system manual
FEDM_ISC_NAME_..._UC	Reader Name in Unicode according to the readers system manual
FEDM_ISC_TR_TYPE_...	Transponder Type according to the appendix of the readers system manual
FEDM_ISC_EPC_TYPE_...	EPC-Type; (EPC = Electronic Product Code)
FEDM_ISC_FU_TYPE_...	Type of function units
FEDM_ISC_ISO_MODE_...	ISO-Command modes respectively mode flags
FEDM_ISC_ISO_BANK_...	Identifier for memory bank of EPC Class1 Gen2 transponder
FEDM_ISC_ISO_MFR_...	Manufacturer code for [0xB1] ISO15693 Host-Commands

7.4.3. Constants for uiTableID

Constant	Description
FEDM_ISC_BRM_TABLE	Table-ID for BRM-Table
FEDM_ISC_ISO_TABLE	Table-ID for ISO-Table

7.4.4. Constants for uiDataID

Constant	Description/Use																																
FEDM_ISC_DATA_TRTYPE	Transponder type <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex		X		X																													
FEDM_ISC_DATA_SNR	Serial number <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X		X	X	X	SetTableData					X	X	X	FindTableIndex					X	X	X
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X		X	X	X																										
SetTableData					X	X	X																										
FindTableIndex					X	X	X																										
FEDM_ISC_DATA_RxDB FEDM_ISC_DATA_RxDB_EPC_BANK FEDM_ISC_DATA_RxDB_TID_BANK FEDM_ISC_DATA_RxDB_RES_BANK	Data blocks from receive protocol Note: Use GetTableData und SetTableData (only ISO-Table) for data blocks !! <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData			X			X	X	FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X			X	X																										
SetTableData			X			X	X																										
FindTableIndex																																	
FEDM_ISC_DATA_TxDB FEDM_ISC_DATA_TxDB_EPC_BANK FEDM_ISC_DATA_TxDB_TID_BANK FEDM_ISC_DATA_TxDB_RES_BANK	Data blocks for send protocol Note: Use GetTableData und SetTableData (only ISO-Table) for data blocks !! <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData			X			X	X	FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X			X	X																										
SetTableData			X			X	X																										
FindTableIndex																																	
FEDM_ISC_DATA_TIMER	Timer value from [0x21] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X	X				SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X	X																													
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_RxCB	Configuration data block from receive protocol Note: Use GetTableData und SetTableData (only ISO-Table) for data blocks !! <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData			X			X	X	FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X			X	X																										
SetTableData			X			X	X																										
FindTableIndex																																	

Constant	Description/Use																																
FEDM_ISC_DATA_TxCB	<div>Configuration data block for send protocol</div> <div>Note: Use GetTableData und SetTableData (only ISO-Table) for data blocks !!</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData			X			X	X	FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X			X	X																										
SetTableData			X			X	X																										
FindTableIndex																																	
FEDM_ISC_DATA_AFI	<div>AFI from [0xB0] [0x2B] Get System Information</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData		X		X		X	X	FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData		X		X		X	X																										
FindTableIndex		X		X																													
FEDM_ISC_DATA_DSFDID	<div>DSFDID from received data [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData		X		X		X	X	FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData		X		X		X	X																										
FindTableIndex		X		X																													
FEDM_ISC_DATA_TRINFO	<div>Transponder Info (only for ISO14443A Transponder) from received data [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex		X		X																													
FEDM_ISC_DATA_OPTINFO	<div>Optional Info (only for ISO14443A Transponder) from received data [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_PROTONFO	<div>Protocol Info (only for ISO14443B Transponder) from received data [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_FSCI	<div>Max. Frame Size (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	

Constant	Description/Use																																				
<div>FEDM_ISC_DATA_FWI</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Frame Waiting Time (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_DSI</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Devisor Send Integer (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_DRI</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Devisor Receive Integer (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_NAD</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Node Address (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_CID</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Card Identifier (only for ISO14443-4 Transponder) from received data [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_SEC_STATUS</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Security Status from received data [0xB0] [0x23] Read Multiple Blocks</div> <div>Note: Use GetTableData und SetTableData (only ISO-Table) for data blocks !!</div> <div><table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData			X			X	X	FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X			X	X																														
SetTableData			X			X	X																														
FindTableIndex																																					

Constant	Description/Use																																				
FEDM_ISC_DATA_BLOCK_SIZE	Block size from received data [0xB0] [0x23] Read Multiple Blocks or [0x22] Read Buffer																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X				SetTableData		X		X		X	X	FindTableIndex							
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X																																	
SetTableData		X		X		X	X																														
FindTableIndex																																					
FEDM_ISC_DATA_MEM_SIZE	Memory size from [0xB0] [0x2B] Get System Information																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X			X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_IC_REF	IC-Reference from [0xB0] [0x2B] Get System Information																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex		X		X			
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex		X		X																																	
FEDM_ISC_DATA_DB_ADR	Data block addresse from received data [0x21] Read Buffer																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_DBN	Number of data blocks from receive data [0x21] Read Buffer or [0x22] Read Buffer																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X	X		X	X	SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_IS_BLOCK_SIZE_SET	Flag, whether block size was set with [0xB0] [0x23] Read Multiple Blocks																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData	X	X	X	X				SetTableData	X	X		X				FindTableIndex	X						
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData	X	X	X	X																																	
SetTableData	X	X		X																																	
FindTableIndex	X																																				
FEDM_ISC_DATA_IS_SELECTED	Flag, wether transponder is in selected mode. Is set with [0xB0][0x25] Select																																				
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData	X	X	X	X				SetTableData	X	X		X				FindTableIndex	X						
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData	X	X	X	X																																	
SetTableData	X	X		X																																	
FindTableIndex	X																																				

Constant	Description/Use																																
FEDM_ISC_DATA_IS_ISO14443_4_INFO	Flag, whether transponder info data are read with [0xB2] [0x2B] ISO14443-4 Transponder Info <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData	X	X	X	X				SetTableData	X	X		X				FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData	X	X	X	X																													
SetTableData	X	X		X																													
FindTableIndex																																	
FEDM_ISC_DATA_EPC	EPC; (EPC = Electronic Product Code) from receive data [0xB0] [0x01] Inventory or from [0x22] Read Buffer with variable length The EPC is a string in the format: "xx.xxxxxx.xxxxxx.xxxxxx" (Header.DomainManager.ObjectClass.Seriennummer) <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData						X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData						X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_EPC_TYPE	EPC-Type; (EPC = Electronic Product Code) from received data [0xB0] [0x01] Inventory. The EPC-Type is extracted from the field EPC-Header. <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X				SetTableData								FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X																													
SetTableData																																	
FindTableIndex		X		X																													
FEDM_ISC_DATA_EPC_HEADER	Field EPC-Header; (EPC = Electronic Product Code) from received data [0xB0] [0x01] Inventory <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData				X		X	X	SetTableData								FindTableIndex				X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData				X		X	X																										
SetTableData																																	
FindTableIndex				X																													
FEDM_ISC_DATA_EPC_DOMAIN	Field EPC-DomainManager; (EPC = Electronic Product Code) from received data [0xB0] [0x01] Inventory <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData					X	X	X	SetTableData								FindTableIndex					X	X	X
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData					X	X	X																										
SetTableData																																	
FindTableIndex					X	X	X																										
FEDM_ISC_DATA_EPC_OBJECT	Field EPC-ObjectClass; (EPC = Electronic Product Code) from received data [0xB0] [0x01] Inventory <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData					X	X	X	SetTableData								FindTableIndex					X	X	X
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData					X	X	X																										
SetTableData																																	
FindTableIndex					X	X	X																										

Constant	Description/Use																																
FEDM_ISC_DATA_EPC_SNR	Field EPC-Serial Number; (EPC = Electronic Product Code) from received data [0xB0] [0x01] Inventory <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td><td>X</td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData					X	X	X	SetTableData								FindTableIndex					X	X	X
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData					X	X	X																										
SetTableData																																	
FindTableIndex					X	X	X																										
FEDM_ISC_DATA_SNR_LEN	Length of serial number from receive protocol [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>Cstring</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	Cstring	string	GetTableData		X	X	X		X	X	SetTableData		X		X		X	X	FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	Cstring	string																										
GetTableData		X	X	X		X	X																										
SetTableData		X		X		X	X																										
FindTableIndex																																	
FEDM_ISC_DATA_ANT_NR	Antenna number from receive protocol [0x21] Read Buffer, [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex		X		X			
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex		X		X																													
FEDM_ISC_DATA_DATE	Date field from receive protocol [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_INPUT	Input byte from receive protocol [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_STATE	Status byte from receive protocol [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData		X	X	X		X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData		X	X	X		X	X																										
SetTableData																																	
FindTableIndex																																	
FEDM_ISC_DATA_MAC_ADR	Status byte from receive protocol [0x22] Read Buffer <table><tr><th></th><th>bool</th><th>UCHAR</th><th>UCHAR[]</th><th>UINT</th><th>__int64</th><th>CString</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData								FindTableIndex							
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																										
GetTableData			X			X	X																										
SetTableData																																	
FindTableIndex																																	

Constant	Description/Use				
FEDM_ISC_DATA_ALL <table><tr><td>BRM-Table</td><td>ISO-Table</td></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	Parameter for the method ResetTable to initialize all table items
BRM-Table	ISO-Table				
X	X				

7.5. Revision history

V4.03.00

- Support for new Reader: **ID CPR46.xx**
- Support for new Transponder types: **Innovatron (ISO 14443B')** und **ASK CTx**
- **New TagHandler classes** for Innovatron (ISO 14443B') und ASK CTx
- **Secured data transmission** with Serial Port and USB
- Support for Gate People Counter events in Notification-Mode
- Buffered-Read-Mode table supports direction information from Gate People Counter
- Update of namespaces and access constants for reader configuration
- Rename of Namespaces:

Old	New
OperatingMode::xxMode::DataSource::MifareAppID	OperatingMode::xxMode::DataSource::Mifare::Classic::AppID
OperatingMode::xxMode::DataSource::MifareKeyAddress	OperatingMode::xxMode::DataSource::Mifare::Classic::KeyAddress
OperatingMode::xxMode::DataSource::MifareKeyType	OperatingMode::xxMode::DataSource::Mifare::Classic::KeyType

Note: xxMode stands for NotificationMode or ScanMode

V4.02.00

- Update of namespaces and access constants for reader configuration
- Support for new Reader: **ID ISC.LR1002**
- **TagHandler class for ISO 14443-4 Mifare DESFire with FlexSoft- and SAM-Crypto:** Bugfix in the method SetConfiguration with values of 1 and 2 in Parameter option
- **ISO 14443-3 and -4 Transponder:** Optimized Select-Algorithm in the method TagSelect.
- **EPC Class 1 Gen 2:**
 1. Support for non-addressed mode.
 2. When UID = EPC + TID is configured: return of an errorcode (-158) before executing of a tag command in addressed mode, if UID contains no TID.
- **TagHandler class for EPC Class 1 Gen 2:**
 5. ISO-Errorcode is a new class member.
 6. Check of length of EPC and Password in all relevant methods.
 7. Method GetTidOfUid returns TID even if length of EPC is zero.
- Class **FEDM_ISCReaderModule:** Rename of the method GetNonAddressedTagHandler in CreateNonAddressedTagHandler.
- Windows:

1. First release of 64-Bit version
2. Dynamic binding to Log-Manager

- First Release for Mac OS X, V10.7.3 or higher

V4.00.07

- Update of namespaces and access constants for reader configuration
- Bugfixes in TagHandler class for EPC Class1 Gen2 in methods ReadCompleteBank, ReadMultipleBlocks and WriteMultipleBlocks
- TagHandler class for EPC Class1 Gen2: new method Lock with simplified parameter list

V4.00.02

- Update of namespaces and access constants for reader configuration
- Check for double UIDs in themethodFEDM_ISCReaderModule::TagInventory
- TagHandler for EPC Class1 Gen2: new method ReadCompleteBank
- Bugfix in the method FEDM_ISCReaderModule::ReadCompleteConfiguration for ID ISC.LR2500-A and ID ISC.LRU3000
- Only for Windows: Dependency from MFCand CRT librariesaccording MS11-025¹

V4.00.00

- Update of namespaces and access constants for reader configuration
- Support for new Reader: **ID ISC.LR2500-A**
- Support for UIDs up to 96 Bytes
- Support for UHF-Configuration UID = EPC + TID
- The organization of the Reader configuration for **ID ISC.LRU3000** above CFG63 is modified with firmwareversion from V2.0.0 and no longercompatiblewith the previous version. This version of FEDM adds the necessary adaptations and is therefore no longer compatible for firmwareversionsless than V2.0.0. The Reader classes do not check of compatibility. This must be done on application-side.

The table below summarizes the compatibilities:

LRU3000-Firmware	use SDK-Version	use ISOStart-Version	XML-Configuration file
< 2.00.00	<= 3.03.01	<= 8.03.02	must be created with ISOStart <= 8.03.02

¹ Microsoft Security Bulletin Article-IDs: 2538218, 2538243 and 2542054 from Juni 14, 2011

LRU3000-Firmware	use SDK-Version	use ISOStart-Version	XML-Configuration file
>= 2.00.00	>= 4.00.00	>= 9.00.00	must be created with ISOStart >= 9.00.00

- The shared use of a TCP/IP connection from different reader objects is no longer supported. ConnectTCP returns with error code -157, if another reader object tries to connect to a Reader with the same IP-Address and Port, which is still connected
- Disconnect of Reader class **FEDM_ISCReaderModule** can return a positive value, if in case of a TCP/IP connection the closing was not successful. The positive return value represents the last status of the connection. It is recommended to view each code line, which call this method.
- New methods in the Reader class **FEDM_ISCReaderModule**: *GetTcpConnectionState*, *GetNonAddressedTagHandler*, *Convert_EPC_C1_G2_TagHandler*
- Support for [0x74] Input Event with Notification-Mode for the Reader **ID CPR50** and **ID MAX50**
- New TagHandler-Classes for IDS SL13A (ISO 15693) and IDS SL900A (EPC Class1 Gen2)
- TagHandler-Class for EPC Class1 Gen2 with new methods: *GetProtocolControl*, *GetEpcOfUid*, *GetTidOfUid*
- The structure **struct _FEDM_TASK_INIT** is extended with new parameters for the Keep-Alive option inside the Notification-Task. In consequence, the new parameter *bKeepAlive* must be set to false or, which is the better approach, initialize the complete structure with 0 (e.g. with *memset*). It is recommended to view each code line, which uses this structure.
- FEDM V4.00.00 requires version numbers of dependent DLLs/SOs as follows:

DLL/SO	Version
FECOM	from 3.00.00
FEUSB	from 4.00.00
FETCP	from 2.00.00
FEISC	from 7.00.00
FETCL	from 2.00.00
FEFU	from 2.00.00

- Automatically set of the right OBID Protocol-Frame: the use of the methods *FindBaudRate()* for serial connections with the following call of *ReadReaderInfo(GetProtocolFrameSupport())* or *ReadReaderInfo()* for USB and TCP connections ensures the set of the right OBID-Protocol-Frame (Standard or Advanced). This is important, because Readers in the future will no longer support the Standard Protocol-Frame and an application will run into a communication timeout when a Standard Protocol-Frame is used.

V3.03.01

- Update of namespaces and access constants for reader configuration
- Support for new Reader: **ID ISC.MRU102**

V3.03.00

- The API of the most DLLs are modified. All applications must be recompiled.
- Update of namespaces and access constants for reader configuration
- Support for new Reader: **ID ISC.LR2500-B**, **ID ISC.MR102**, **ID CPR30.xx** und **ID ISC.CPR52.xx**

V3.02.00

- The API of the most DLLs are modified. All applications must be recompiled.
- Update of namespaces and access constants for reader configuration
- New class for support of external Units like **People-Counter**

V3.01.00

- The API of the most DLLs are modified. All applications must be recompiled.
- Release of TagHandler classes to simplify tag communication with certain tag types, especially ISO 14443 and ISO 15693
- Support for secured data transmission with **ID CPR50**, **ID MAX50** and **ID ISC.LRU3000**
- Update of namespaces and access constants for reader configuration

V3.00.14

- Support for new Reader type: **ID MAX50.xx**
- Support for new Reader type: **ID ISC.LRU3000**
- 2nd Beta-Release of TagHandler classes to simplify tag communication with certain tag types, especially ISO 14443 and ISO 15693
- Update of namespaces and access constants for reader configuration
- New utility classes **FedmlscReport_ReaderInfo** and **FedmlscReport_ReaderDiagnostic** creates report streams from informationdata or state data (have a look to ISOStart since V8.01.02)

V3.00.07

- Support for new Reader type: **ID CPR50.xx**
- Beta-Release of TagHandler classes to simplify tag communication
- Update of namespaces and access constants for reader configuration

- The method `SendProtocol(0x72)` use internally modified definitions of the constants `FEDM_ISC_TMP_0x72_OUT_TYPE_1...FEDM_ISC_TMP_0x72_OUT_TYPE_8`: Up to the previous release they addresses one bit. Now they addresses three bits. Thus, the OUT-TYPE 'Relay' must be set to 0x04 instead of 0x01 ([6.4.3. Examples for using the method SendProtocol](#)). This is applied to all reader types which supports the command [0x72] Set Output.

V3.00.02

- New method in **FEDM_ISCReaderModule**
 - `ReadReaderDiagnostic`
- Improvements in methods of **FEDM_ISCReaderModule**
 - `ApplyConfiguration`
 - `ReadCompleteConfiguration`
 - `WriteCompleteConfiguration`
- New struktur **FEDM_ISC_READER_DIAGNOSTIC**
- Update of namespaces and access constants for reader configuration

V3.00.00

- The main modifications are documented in Part A (H10102-xe-ID-B)
- Support for new reader: **ID ISC.MRU200**, **ID ISC.PRHD102**, **ID CPR40.xx**
- The following older reader types are no longer supported: ID ISC.M01, ID ISC.LR100
- Support for UHF-Multiplexer **ID ISC.ANT.UMUX**
- Support for transponder type EPC Class1 Gen2 HF
- New high-level methods in **FEDM_ISCReaderModule**
 - `ApplyConfiguration`
 - `ReadCompleteConfiguration`
 - `WriteCompleteConfiguration`
 - `ResetCompleteConfiguration`
- New method `EvalLibDependencies` in class **FEDM_ISCReader** for detecting version conflicts with dependent library files
- Collecting of all access constants for reader configuration in namespaces improves the clearness
- New overloaded methods `Get/SetConfigPara` in class **FEDM_ISCReader** for modifying reader configuration parameters
- Writing of reader configuration is only possible for previous read configuration blocks except, if the reader configuration is load by a XML file.

V2.06.00

- Small modifications for the reader ID ISC.LRU2000
- New function in the class **FEDM_ISCReaderModule**: ReadReaderInfo
- New struktur **FEDM_ISC_READER_INFO**

V2.05.06

- The Linux library is compiled with GCC 3.3.3 under SuSE Linux 9.1
- The sources are ready to compile with Visual Studio 2005
- Support for the new protocol [0x6B] Centralized RF Synchronization

V2.05.01

- New function **TriggerAsyncTask** in the Reader class **FEDM_ISCReaderModule**
- Modified licence agreement

V2.05.00

- New advanced Reader Class FEDM_ISCReaderModule with high-level functions and support for Notification Mode¹
- Support of the new UHF-Reader ID ISC.LRU2000
- Extensions for the UHF-Reader ID ISC.LRU1000 concerning the configuration

V2.04.00

- New common constants for the UHF-Reader LRU1000:

Constant	Comment
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK_LGT	Constants for Selection Mask in the reader configuration for the transponder type EPC Class 1 Gen 1
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_LGT	Constants for Selection Mask in the reader configuration for the transponder type EPC Class 1 Gen 2
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE_TRUNC	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE_BANK	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MSB	

¹ the Notification Mode not available in every Reader

Constant	Comment
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_LGT	Constants for Selection Mask in the reader configuration for the transponder type ISO18000-6-B
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_MODE	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK	

V2.03.05

- Support of the new HF-Reader ID ISC.LR2000
- Extensions for the UHF-Reader ID ISC.LRU1000 concerning the configuration
- Support for the new transponder types: HF-Transponder Innovision Jewel and UHF-Transponder EPC Class0/0+
- Integration of new options for the Advanced Buffered Read Mode:
 1. Input and Status informations can be transfered in the trigger mode
 2. Extension of TR-DATA in the reader configuration and in the response data of protocol [0x22] Read Buffer
- Extensions for ISO 14443A-Transponder:
 1. [0xB2][0x30] Mifare Value Commands
 2. [0xB0][0x25] Select supports Card Information
- New global function **FEDM_ConvHexUCharToTwoAscii**
- Support for the new protocol [0x72] Set Output

- New common constants:

Constant	Comment
FEDM_ISC_TMP_B0_MODE_CINF	Flag Card Information in Mode-Byte for [0xB0][0x25] Select
FEDM_ISC_TMP_B0_MODE_WR_NE	Flag Write-Erase in Mode-Byte for [0xB0][0x24] Write Multiple Blocks
FEDM_ISC_TMP_B0_RSP_FORMAT	Format Byte in response protocol of [0xB0][0x25] Select, if CINF-Flag is set
FEDM_ISC_TMP_B2_REQ_MF_CMD	Parameter for [0xB2][0x30] Mifare Value Commands
FEDM_ISC_TMP_B2_REQ_OP_VALUE	
FEDM_ISC_TMP_B2_REQ_DEST_ADR	
FEDM_ISC_TMP_ADV_BRM_TRDATA2	2. Byte of TR-DATA in response protocol of [0x22] Read Bufer
FEDM_ISC_TMP_ADV_BRM_TRDATA2_...	Flags in 2. Byte of TR-DATA in response protocol of [0x22] Read Bufer
FEDM_ISC_TMP_0x72_OUT...	Constants for [0x72] Set Output

- Modified common constants:

Old Constant	New Constant
FEDM_ISC_TMP_ADV_BRM_TRDATA1	FEDM_ISC_TMP_ADV_BRM_TRDATA
FEDM_ISC_TMP_ADV_BRM_TRDATA1_...	FEDM_ISC_TMP_ADV_BRM_TRDATA_...

V2.03.00

- Modifications for ISO 14443A Transponder.
- Modifikation in der Zugriffskonstanten FEDM_ISC_TMP_B0_REQ_UID. Nähere Einzelheiten in [6.5.1. Anomaly of the addressed mode](#)¹
- New options for EPCglobal Class1 Gen2 transponder (e.g. [0xB3] EPC Command)
- Support of protocols with enhanced options (variable UID length, bank number, 2 byte block address, access password)
- Extension of the table class FEDM_BRMTblItem for the Buffered Read Mode for EPC transponder
- The blocksize of the byte array TmpData for temporary data is changed from 16 to 32.¹
- The table constants, beginning with FEDM_ISC_DATA_... are completely renumbered.¹
- Some old access constants addresses a new memory space.¹
- New pre-processor definition **_FEDM_XML_SUPPORT** for including the XML serialization classes. This option was not necessary in previous versions. In front of the re-compilation of a project, this definition must be set, if the XML serialization classes are used.

¹ Applications, which loads the FEDM library dynamically, must completely re-compiled.

5. New pre-processor definition **_FEDM_MFC_SUPPORT** for including the MFC classes (basically CString and CArchive). This option was not necessary in previous versions. In front of the re-compilation of a project, this definition must be set, if the MFC classes are used.

- Function ResetTable has a little modified behaviour.
- New overload function SetTableSize. This function allows the dimensioning of the size of some data arrays deviating from the given values.
- New common constants:

Constant	Comment
FEDM_ISC_TMP_B0_MODE_EXT_ADR	Flag in mode byte for extended address mode in [0xB0] send protocol
FEDM_ISC_TMP_B0_MODE_UID_LF	Flag in mode byte for UID length in [0xB0] send protocol
FEDM_ISC_TMP_B0_REQ_BANK	Bank in [0xB0] send protocol
FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG	Access flag in Bank in [0xB0] send protocol
FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR	Bank number in Bank in [0xB0] send protocol
FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT	2 byte address in [0xB0] send protocol
FEDM_ISC_TMP_B1_REQ_TI_PASSWORD	Password for [0xB1] Cust and Proprietary Commands for transponder from Texas Instruments
FEDM_ISC_TMP_B3_ISO_ERROR	ISO Error Code in return protocol of [0xB3] Command
FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH	Length of password (in byte) for [0xB0] send protocol
FEDM_ISC_TMP_B0_ACCESS_PW	Password for [0xB0] send protocol
FEDM_ISC_TMP_B3_CMD	Subcommand for [0xB3] send protocol
FEDM_ISC_TMP_B3_MODE	Mode byte for [0xB3] send protocol
FEDM_ISC_TMP_B3_MODE_EXT_ADR	Flag in mode byte for extended address mode in [0xB3] send protocol
FEDM_ISC_TMP_B3_MODE_EPC_LF	Flag in mode byte for EPC length in [0xB3] send protocol
FEDM_ISC_TMP_B3_MODE_ADR	Address mode in mode byte in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_EPC_LEN	EPC length in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_BANK	Bank in [0xB3] Send protocol
FEDM_ISC_TMP_B3_REQ_BANK_ACCESS_FLAG	Access flag in Bank in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_BANK_BANK_NR	Bank number in Bank in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_DB_ADR_EXT	2 byte address in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_DB_ADR	1 byte address in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_DBN	Number of data blocks in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_DB_SIZE	Blocksize in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_TR_TYPE	Transponder type in [0xB3] send protocol
FEDM_ISC_TMP_B3_REQ_EPC	EPC in [0xB3] send protocol
FEDM_ISC_TMP_B3_KILL_PW_LENGTH	Length of Kill Password in [0xB3] send protocol
FEDM_ISC_TMP_B3_KILL_PW	Kill Password in [0xB3] send protocol
FEDM_ISC_TMP_B3_LOCK_DATA_LENGTH	Length of Lock Data in [0xB3] send protocol

Constant	Comment
FEDM_ISC_TMP_B3_LOCK_DATA	Lock Data in [0xB3] send protocol
FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH	Length of Password in [0xB3] send protocol
FEDM_ISC_TMP_B3_ACCESS_PW	Password in [0xB3] send protocol
FEDM_ISC_TMP_B3_RSP_DB_ADR_E	Address of data block when signaling ISO error in [0xB3] return protocol

- Modified common Constants:

Old Constant	New Constant
FEDM_ISC_FU_TMP_MUX_CH_IN1	FEDM_ISC_FU_TMP_MUX_OUT_CH1
FEDM_ISC_FU_TMP_MUX_CH_IN2	FEDM_ISC_FU_TMP_MUX_OUT_CH2
FEDM_ISC_TMP_FIRMWARE_VERSION	FEDM_ISC_TMP_READER_INFO

V2.02.00

- Remove of the table classes FEDM_ISOTabItem and FEDM_BRMTabItem from the reader class file and store them in own files.
- Support of EPC und I-CODE UID transponder in the Buffered Read Mode for the reader ID ISC.LR200.
- New constants for configuration parameters for HF-Reader ID ISC.M02, ID ISC.MR/PR/PRH100 and ID ISC.MR101 in FEDM_ISCReaderID.h.

Constant	Comment
FEDM_ISC_EE_SCAN_END_USER1	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_END_USER2	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_END_USER3	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER1	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER2	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER3	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER4	Parameter for Scan-Mode
FEDM_ISC_EE_SCAN_LEN_USER	Parameter for Scan-Mode

V2.01.00

- Support for the new reader ID ISC.MR101 (serial and USB version).
- Support for external function units.
- New constants for the UHF-Reader in FEDM_ISCReaderID_LRU1000.h.
- New general constants:

Constant	Comment
----------	---------

Constant	Comment
FEDM_ISC_TMP_DIAG_DATA	This identifier is valid for all reader diagnostic modes and substitutes the below listed removed constants for the modes 0x01, 0x02, 0x03.

- Modified general constants:

Old Constant	New Constant
FEDM_ISCLR_TMP_DIAG_MODE	FEDM_ISC_TMP_DIAG_MODE

- Removed general constants:

Constant	Comment
FEDM_ISCLR_TMP_DIAG_0x01_DATA	Substituted through FEDM_ISC_TMP_DIAG_DATA
FEDM_ISCLR_TMP_DIAG_0x02_DATA	Substituted through FEDM_ISC_TMP_DIAG_DATA
FEDM_ISCLR_TMP_DIAG_0x03_DATA	Substituted through FEDM_ISC_TMP_DIAG_DATA
FEDM_ISC_TMP_EPC_DESTROY_LEN	Removed, because of internal calculation of the length of EPC/UID based on the destroy mode and the header of the EPC.

- Some minor bug fixes.

V2.00.00

- Support for UHF-Transponder ISO18000-6-A, EM4222, EPC Gen 2, EPC Class 1
- Support of datablocks read with Advanced Buffered Read Mode.
- Modification for the Buffered Read Mode: The datatype of the number of blocks DBN is changed to **unsigned int**. Thus, you cannot read the table value with the function GetTableData(idx, FEDM_BRM_TABLE, FEDM_ISC_DATA_DBN, ...) for **unsigned char**. Potentially you must adapt your program code.
- New Constants in FEDM_ISCReaderID_LRU1000.h:

New	Comment
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_A	Flag for tag driver ISO18000-6-A in the reader configuration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_D	Flag for tag driver EM4222 in the reader configuration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_E	Flag for tag driver EPC Gen 2 in the reader configuration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_J	Flag for tag driver EPC Class 1 in the reader configuration
FEDM_ISC_LRU1000_EE_PER_MODE	Persistence mode in the reader configuration block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT1	Reset time for antenna 1 in the reader configuration block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT2	Reset time for antenna 2 in the reader configuration block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT3	Reset time for antenna 3 in the reader configuration block CFG16

New	Comment
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT4	Reset time for antenna 4 in the reader configuration block CFG16

V1.09.11

- Support for HF-Transponder I-Code UID
- Remove of all access constants which represents the location RAMDATA_MEM. This reduces the number of access constants dramatically. The access to values in RAMDATA_MEM can be realized instead of with the function FEDM_MdfyMemID and the access constants for EEDATA_MEM.
- New Constant in FEDM_ISCReaderID.h:

New	Comment
FEDM_ISC_EE_RF_TAG_DRV_H	Flag for Tag-Driver I-Code UID in the Reader-Configuration

V1.09.10

- Supports the new reader ID ISC.LRU1000 (configuration constants in FEDM_ISCReaderID_LRU1000.h)
- Supports the new reader ID ISC.MR200 (configuration constants in FEDM_ISCReaderID_MR200.h)
- Support for Advanced Protocol Frames with two length bytes.
- Expansions in the class FEDM_BRMTblItem for the new protocol [0x22] Read Buffer
- New configuration constants for ID ISC.LR200 (configuration constants moved to FEDM_ISCReader_LR200.h)
- New protocols: [0x18] Destroy EPC, [0x22] Read Buffer, [0x64] System Reset, [0x66] Firmware Version, [0x87] Set System Date, [0x88] Get System Date
- Introduces the following new constants in FEDM_ISCReader.h:

New	Comment
FEDM_ISC_DATA_ANT_NR	antenna number in table
FEDM_ISC_DATA_SNR_LEN	length of serial number in table
FEDM_ISC_DATA_DATE	date in table

- Introduces the following new constants in FEDM_ISCReaderID.h:

New	Comment
FEDM_ISC_TMP_SOFTVER_RX_BUF	field length of receive buffer in [0x65] Software Version
FEDM_ISC_TMP_SOFTVER_TX_BUF	field length of send buffer in [0x65] Software Version
FEDM_ISCLR_TMP_BRM_TRDATA_ANT	antenna number in protocol data

New	Comment
FEDM_ISC_TMP_EPC_DESTROY_MODE	transfer parameter in [0x18] Destroy EPC
FEDM_ISC_TMP_EPC_DESTROY_LEN	
FEDM_ISC_TMP_EPC_DESTROY_PASSWORD	
FEDM_ISC_TMP_DESTROY_EPC	transfer parameter EPC in [0x18] Destroy EPC Note: the EPC can be longer than the blocksize of 16.The example in. 6.4.3. Examples for using the method SendProtocol demonstrate the right use of the constant.
FEDM_ISC_TMP_SYSTEM_RESET_MODE	transfer parameter in [0x64] System Reset
FEDM_ISC_TMP_ADV_BRM_SETS	transfer parameter in [0x22] Read Buffer
FEDM_ISC_TMP_ADV_BRM_TRDATA	
FEDM_ISC_TMP_ADV_BRM_TRDATA_SNR	
FEDM_ISC_TMP_ADV_BRM_TRDATA_DB	
FEDM_ISC_TMP_ADV_BRM_TRDATA_ANT	
FEDM_ISC_TMP_ADV_BRM_TRDATA_TIME	
FEDM_ISC_TMP_ADV_BRM_TRDATA_DATE	
FEDM_ISC_TMP_ADV_BRM_RECSETS	
FEDM_ISC_TMP_ADV_BRM_VALID_TIME	
FEDM_ISC_TMP_FIRMWARE_VERSION_MODE	transfer parameter in [0x66] Firmware Version
FEDM_ISC_TMP_FIRMWARE_VERSION_SW_MAJOR	
FEDM_ISC_TMP_FIRMWARE_VERSION_SW_MINOR	
FEDM_ISC_TMP_FIRMWARE_VERSION_SW_DEV	
FEDM_ISC_TMP_DATE_CENTURY	transfer parameter in [0x87] Set System Date and [0x88] Get System Date
FEDM_ISC_TMP_DATE_YEAR	
FEDM_ISC_TMP_DATE_MONTH	
FEDM_ISC_TMP_DATE_DAY	
FEDM_ISC_TMP_DATE_TIMEZONE	
FEDM_ISC_TMP_DATE_HOUR	
FEDM_ISC_TMP_DATE_MINUTE	
FEDM_ISC_TMP_DATE_MILLISECOND	

- Remove of the following constants in FEDM_ISCReaderID.h:

New	Comment
FEDM_ISCLR_EE_SYSG_SYS_MODE_BRM	configuration constant is replaced by FEDM_ISCLR_EE_SYSG_SYS_MODE_OP_MODE ersetzt
FEDM_ISCLR_EE_SYSG_SYS_MODE_SCAN	configuration constant is replaced by FEDM_ISCLR_EE_SYSG_SYS_MODE_OP_MODE ersetzt

V1.09.00

- Serializing the reader configuration in XML format using the new classes FEDM_XMLBase and FEDM_XMLReaderCfgDataModul. Knowledge of the classes is not necessary.
- Expansions in the class FEDM_ISOTabItem for EPC (Electronic Product Code)
- Supports the new reader ID ISC.M02
- Full support of the data types bool, __int64 and STL-string with new overloaded functions GetTableData, SetTableData and FindTableIndex.
- Introduces the following new constants in FEDM_ISCReader.h:

New	Comment
FEDM_ISC_DATA_SAK	Select Acknowledge (only for ISO14443A Transponders)
FEDM_ISC_DATA_EPC	EPC; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_TYPE	EPC-Typ; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_HEADER	Field EPC Header; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_DOMAIN	Field EPC-DomainManager; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_OBJECT	Field EPC-ObjectClass; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_SNR	Field EPC-Serial Number; (EPC = Electronic Product Code)
FEDM_ISC_TYPE_...	Reader typ according to the field SW-TYPE of the protocol [0x65] Software Version
FEDM_ISC_NAME_...	Reader Name according to the system manual
FEDM_ISC_NAME_..._UC	Reader Name in Unicode according to the system manual
FEDM_ISC_TR_TYPE_...	Transponder type according to the system manual
FEDM_ISC_EPC_TYPE_...	EPC-Type, listed in FEDM_ISCReader.h; (EPC = Electronic Product Code)

V1.07.00

- Support of new protocols: [0xB1] and [0xB2]
- New Constants in FEDM_ISCReaderID:

New	Comment
FEDM_CPR_EE_BLOCK2 and all special configuration constants for the EEPROM of the reader	configuration constants for the reader ID CPR.02
FEDM_CPR_RAM_BLOCK2 and all special configuration constants for the RAM of the reader	
FEDM_ISC_TMP_B1_CMD	specifys the 0xB1-Command
FEDM_ISC_TMP_B1_MODE	specifys the mode for the 0xB1-Command
FEDM_ISC_TMP_B1_MODE_ADR	specfiys the address mode for the 0xB1-Command
FEDM_ISC_TMP_B1_MFR	manufacturer code for the 0xB1-Command

New	Comment
FEDM_ISC_TMP_B1_REQ_UID	UID in request protocol of 0xB1-Commando
FEDM_ISC_TMP_B1_ISO_ERROR	received ISO error code
FEDM_ISC_TMP_B2_CMD	specifys the mode for the 0xB2-Command
FEDM_ISC_TMP_B2_MODE	specfiys the address mode for the 0xB2-Command
FEDM_ISC_TMP_B2_MODE_ADR	specfiys the address mode for the 0xB1-Command
FEDM_ISC_TMP_B2_REQ_UID	UID in request protocol of 0xB1-Commando
FEDM_ISC_TMP_B2_REQ_DB_ADR	parameter for [0xB2] [0xB0] Authent Mifare protocol
FEDM_ISC_TMP_B2_REQ_KEY_ADR	
FEDM_ISC_TMP_B2_REQ_KEY_TYPE	
FEDM_ISC_TMP_B2_REQ_KEY_ADR_TAG	parameter for [0xB2] [0xB1] Authent my-d protocol
FEDM_ISC_TMP_B2_REQ_KEY_ADR_SAM	
FEDM_ISC_TMP_B2_REQ_AUTH_COUNTER_ADR	
FEDM_ISC_TMP_B2_REQ_KEY_AUTH_SEQUENCE	
FEDM_ISC_TMP_B2_ISO_ERROR	received ISO error code

V1.06.00

- Rename of Constants in FEDM_ISCReaderID:

Old	New
FEDM_ISCPRH_EE_SCAN_DBN	FEDM_ISCPRH_EE_SCAN_D_LGT

- New Constants in FEDM_ISCReaderID:

New	Comment
FEDM_ISC_EE_BLOCK0 .. 7 and all special cofiguration constants for the EEPROM of the reader	Common configuration constants for the following readers: ID ISC.PR100 ID ISC.MR100 ID ISC.PRH100 ID ISC.PR100-U ID ISC.MR100-U ID CPR.02 ID CPR.M02
FEDM_ISC_RAM_BLOCK0 .. 7 and all cspecial cofiguration constants for the RAM of the reader	

- Error corrections for Selected Mode of ISO-Host Commands
- Support of GNU C-Compiler under Linux.

V1.05.00

- The size of receive buffer for datablocks in Buffered-Read-Mode is increased from 128 to 1024 Byte.

V1.04.00

- Support for all Reader types in the *i-scan* family
- Integration of the ISO host commands for all Readers in the *i-scan* family is now finished. This means that transponders conforming to ISO 15693 having different block sizes are supported.
- The internal table ISCTable has been divided into two tables (m_ISOTable for data exchange via ISO host commands and m_BRMTable for data from long-range Readers of type ISC.LRxxx in Buffered Read Mode). This eliminates the GetTableType function.
- In expanding the port in the abstract basic class, all table functions had to have the handover parameter uiTableID (table type constant) added.
- Some name changes and expansions for the access constants.
- Support for Multi-Job-Poll has been eliminated.
- The protocols [0x11], [0x14], [0x15], [0x16] and [0x17] for the ID ISC.M01 are no longer supported.
- The data containers SN_Mem, PubMem and ConfMem are no longer used (pertains only to the Reader ISC.M01)

V1.03.00

- Along with expansion of the interface in the abstract base class all the table functions had to have the transfer parameter uiTableID (table type constant) added.
- Some name changes to and expansions of the access constants.

V1.00.00

- First release version