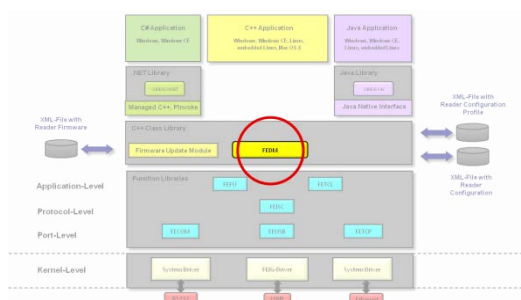


C++ Klassenbibliothek ID FEDM

Version 4.06.06

Teil B.ISC

Software-Support für
OBID i-scan® und OBID® classic-pro



Betriebssystem	Ausführung		Anmerkungen
	32-Bit	64-Bit	
Windows XP	X	(X)	bei 64-Bit nur mit 32-Bit Laufzeitsystem
Windows Vista / 7 / 8	X	X	
Windows CE	X	-	
Linux	X	X	
Android	X		Auf Anfrage
Apple Mac OS X	-	X	ab V10.7.3, Architektur x86_64

Hinweis

© Copyright 2001-2014 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
eMail: obid-support@feig.de

Alle früheren Ausgaben verlieren mit diesem Handbuch ihre Gültigkeit.
Die Angaben in diesem Handbuch können ohne vorherige Ankündigung geändert werden.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung ihres Inhalts sind nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlung verpflichtet zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder Gebrauchsmuster-Eintragung vorbehalten.

Die Zusammenstellung der Informationen in diesem Handbuch erfolgt nach bestem Wissen und Gewissen. FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung für die Richtigkeit und Vollständigkeit der Angaben in diesem Handbuch. Insbesondere kann FEIG ELECTRONIC GmbH nicht für Folgeschäden aufgrund fehlerhafter oder unvollständiger Angaben haftbar gemacht werden. Da sich Fehler, trotz aller Bemühungen nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung dafür, dass die in diesem Dokument enthaltenen Informationen frei von fremden Schutzrechten sind. FEIG ELECTRONIC GmbH erteilt mit diesem Dokument keine Lizenzen auf eigene oder fremde Patente oder andere Schutzrechte.

Die in diesem Handbuch gemachten Installationsempfehlungen gehen von günstigsten Rahmenbedingungen aus. FEIG ELECTRONIC GmbH übernimmt keine Gewähr für die einwandfreie Funktion einer OBID®-Anlage in systemfremden Umgebungen.

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.

Android is a trademark of Google Inc.

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.

Lizenzvertrag über die Nutzung der Software

Dies ist ein Vertrag zwischen Ihnen und der FEIG ELECTRONIC GmbH (nachfolgend "FEIG") über die Nutzung der überlassenen Klassenbibliothek ID FEDM und die vorliegende Dokumentation, nachfolgend Lizenzmaterial genannt. Mit der Installation und Benutzung der Software erklären Sie sich mit allen Bestimmungen dieses Vertrages ausnahmslos und ohne Einschränkung einverstanden. Wenn Sie mit den Bestimmungen dieses Vertrages nicht oder nicht vollständig einverstanden sind, dürfen Sie das Lizenzmaterial nicht installieren oder anderweitig benutzen. Das überlassene Lizenzmaterial ist Eigentum der FEIG ELECTRONIC GmbH und ist international urheberrechtlich geschützt.

§1 Vertragsgegenstand und Vertragsumfang

1. FEIG gewährt Ihnen das Recht, das überlassene Lizenzmaterial zu installieren und zu den nachstehenden Bedingungen zu nutzen.
2. Sie dürfen sämtliche Bestandteile des Lizenzmaterials auf einer Festplatte oder einem sonstigen Speichermedium installieren. Die Installation und Nutzung darf auch auf einem Netzwerk-Fileserver erfolgen. Sie dürfen Sicherheitskopien des Lizenzmaterials anfertigen.
3. FEIG gewährt Ihnen das Recht die dokumentierte Klassenbibliothek für die Entwicklung eigener Anwendungsprogramme oder Programmbibliotheken zu verwenden und Sie dürfen die Laufzeitdateien FedmlscCoreVCxx.dll¹, FedmlscCoreCE.dll, libFedmlscCore.x.y.z.dylib² und libFedmlscCore.so.x.y.z² ohne Abgabe von Lizenzgebühren vertreiben, unter der Voraussetzung, dass diese Anwendungsprogramme oder Programmbibliotheken dazu dienen, Geräte und / oder Anlagen anzusteuern oder zu betreiben, die von FEIG entwickelt und / oder vertrieben werden.

§2. Schutz des Lizenzmaterials

1. Das Lizenzmaterial ist geistiges Eigentum von FEIG und seinen Lieferanten. Es ist gemäß Urheberrecht, internationalen Verträgen und einschlägigen Gesetzen des Landes geschützt, in dem sie genutzt wird. Struktur, Organisation und Code der Software sind wertvolles Geschäftsgeheimnis und vertrauliche Information von FEIG und seinen Lieferanten.
2. Sie verpflichten sich, die Klassenbibliothek sowie die Dokumentation nicht zu ändern.
3. Soweit FEIG im Lizenzmaterial Schutzvermerke, wie Copyright-Vermerke und andere Rechtsvorbehalte angebracht hat, sind Sie verpflichtet, diese unverändert beizubehalten sowie in alle von Ihnen hergestellten vollständigen oder teilweisen Kopien in unveränderter Form zu übernehmen.
4. Die Weitergabe von Lizenzmaterial ist weder vollständig noch auszugsweise gestattet, solange dazu keine explizite anderslautende Vereinbarung zwischen Ihnen und FEIG getroffen wurde. Nicht betroffen von dieser Regelung sind solche Anwendungsprogramme oder Programmbibliotheken, die gem. §1 Absatz 3. dieser Vereinbarung erstellt und vertrieben werden, solange die Weitergabe des Sourcecodes der Klassenbibliothek ausgeschlossen bleibt.

§3 Gewährleistung und Haftungsbeschränkungen

1. Sie stimmen mit FEIG darüber überein, dass es nicht möglich ist, EDV-Programme so zu entwickeln, dass sie für alle Anwendungsbedingungen fehlerfrei sind. FEIG weist Sie ausdrücklich darauf hin, dass die Installation eines neuen Programms bereits vorhandene Software beeinflussen kann, und zwar auch solche Software, die nicht gleichzeitig mit der neuen Software ausgeführt wird. FEIG haftet in keinem Fall für direkte oder indirekte Schäden, für Folgeschäden oder Sonderschäden, Einschließlich entgangenen Geschäftsgewinn oder entgangener Einsparungen. Wenn Sie sicherstellen wollen, dass es zu keinerlei Beeinflussung eines bereits installierten Programms kommt, dürfen Sie die vorliegende Software nicht installieren.
2. FEIG weist ausdrücklich darauf hin, dass mit der Software irreversible Einstellungen und Anpassungen an Geräten vorgenommen werden können, wodurch diese Geräte zerstört oder unbrauchbar gemacht werden können. FEIG übernimmt für derartiges Handeln unabhängig davon ob dies bewußt oder unbewußt erfolgte keinerlei Gewährleistung.
3. FEIG liefert Ihnen die Software "wie besehen" ohne jegliche Gewährleistung. FEIG kann für die Leistung oder die Ergebnisse, die Sie durch die Nutzung der Software erzielen, nicht garantieren. FEIG übernimmt keine Gewährleistung oder Garantie dafür, dass keine Schutzrechte Dritter verletzt werden, auch nicht dafür, dass die Software für irgendeinen bestimmten Zweck geeignet ist.
4. FEIG weist ausdrücklich darauf hin, dass das Lizenzmaterial nicht für den Einsatz mit oder in medizinischen Geräten oder für Geräte für lebenserhaltende Maßnahmen konzipiert ist, bei denen ein Fehler eine Gefahr für menschliches Leben oder für die gesundheitliche Unversehrtheit zur Folge haben kann.

¹ xx repräsentiert die Versionsnummer der gelinkten MFC-Bibliothek

² x.y.z repräsentiert die aktuelle Versionsnummer

Der Anwender des Lizenzmaterials ist dafür verantwortlich, geeignete Maßnahmen zu ergreifen um Gefahren, Schäden oder Verletzungen zu vermeiden.

§4 Schlußbestimmungen

1. Dieser Vertrag enthält die vollständigen Lizenzbestimmungen und ersetzt alle eventuell vorangegangenen Regelungen und Absprachen. Änderungen und Ergänzungen bedürfen der Schriftform.
2. Sollte eine der in diesem Vertrag enthaltenen Bestimmungen unwirksam sein oder werden, so wird die Gültigkeit der übrigen Bestimmungen hierdurch nicht berührt. Beide Vertragsparteien verpflichten sich, die unwirksame Bestimmung durch eine solche wirksame Bestimmung zu ersetzen, die dem wirtschaftlichem Zweck der zu ersetzenden Bestimmung am nächsten kommt.
3. Dieser Vertrag unterliegt dem Recht der Bundesrepublik Deutschland. Gerichtsstand ist Frankfurt a. M.

Inhalt:

Lizenzvertrag über die Nutzung der Software	3
Inhalt:	5
Anmerkungen zur Dokumentation dieser Bibliothek.....	8
1. Übersicht	9
2. Änderungen gegenüber der Vorversion	11
3. Installation.....	12
3.1. 32- und 64-Bit Windows XP/Vista/7/8	12
3.2. Windows CE.....	14
3.3. 32- und 64-Bit Linux	15
3.4. 64-Bit Mac OS X.....	16
3.5. Quellcode.....	17
3.6. Abhängigkeiten	19
4. Einbindung in das Anwendungsprogramm.....	20
4.1. Unterstützte Entwicklungsumgebungen	20
4.2. Der schnelle Weg	20
4.2.1. Einbindung in Visual Studio.....	20
4.2.2. Einbindung in Xcode	24
4.3. Manuelle Einbindung	25
4.3.1. Include-Dateien.....	25
4.3.2. Quellcode-Dateien	25
4.3.3. Notwendige Projekt-Einstellungen.....	26
4.3.4. Windows	26
4.3.5. Linux	26
4.3.6. Optionale Projekt-Einstellungen	26
5. Installation auf dem Zielrechner	27
5.1. 32-Bit Bibliotheken für 32- und 64-Bit Windows	27
5.2. 64-Bit Bibliotheken für 64-Bit Windows	28
5.3. 32- und 64-Bit Linux	29
5.4. 64 Bit Mac OS X	29
6. Klassenbeschreibung.....	30

6.1. FEDM_ISCReader	30
6.1.1. Implementierte Datencontainer	30
6.1.2. Implementierte Tabellen.....	30
6.1.3. Methoden (public)	31
6.2. FEDM_ISCReaderModule.....	35
6.2.1. Methoden (public)	35
6.3. TagHandler-Konzept	38
6.4. Mit den Leserklassen arbeiten	39
6.4.1. Notwendige Initialisierungen	39
6.4.2. Verwaltung der Leserkonfiguration.....	41
6.4.3. Beispiele für die Verwendung der Methode SendProtocol.....	44
6.4.4. Asynchrone Tasks zur Entlastung von Applikationen	53
6.4.5. Die Serialisierung im XML-Format.....	57
6.5. Tabelle für ISO Host Commands	59
6.5.1. Besonderheiten des addressed mode	60
6.5.2. Beispiele für die Verwendung der Tabelle mit [0xB0] Protokollen.....	61
6.5.3. Beispiele für die Verwendung der Tabelle mit [0xB3] Protokollen.....	70
6.6. Tabelle für den Buffered Read Mode	73
6.6.1. Beispiele für die Verwendung der Tabelle	73
6.7. FEDM_ISCFunctionUnit	76
6.7.1. Konstruktor	77
6.7.2. Implementierte Datencontainer	77
6.7.3. Implementierte Listen.....	77
6.7.4. Methoden (public)	77
6.7.5. Notwendige Initialisierungen	78
6.7.6. Beispiele für die Verwendung der Methode SendProtocol.....	79
6.8. FedmlscPeopleCounter	81
6.8.1. Methoden (public)	81
6.8.2. Beispiel für die Verwendung der Klasse.....	81
6.8.3. Beispiel für automatische Notifikation.....	82
7. Anhang	84
7.1. Unterstützte OBID® Leser	84
7.2. Unterstützte Transponder.....	86
7.3. TCP-Status.....	87
7.4. Liste der Konstanten.....	88
7.4.1. Interne Konstanten.....	88
7.4.2. Allgemeine Konstanten	88
7.4.3. Konstanten für uiTableID	89
7.4.4. Konstanten für uiDataID	90

7.5. Änderungshistorie.....	97
-----------------------------	----

Anmerkungen zur Dokumentation dieser Bibliothek

Dieses Handbuch beschreibt eine Software-Bibliothek, die Ihnen auch als kommentierter Sourcecode vorliegt. Aus diesem Grund wird darauf verzichtet, mehr zu dokumentieren als unbedingt für das Verständnis der Funktionsweise und der Verwendung der Klassen notwendig ist. Es wird vorausgesetzt, dass der Anwender dieser Bibliothek den Sourcecode lesen und sich in die Details mit Hilfe dieser Dokumentation, der Header-Dateien und den eingefügten Kommentaren selbst zurechtfinden lernt.

Zum Verständnis der internen Programmabläufe müssen immer auch die Systemhandbücher der eingesetzten OBID®-Leser und der OBID®-Funktionsbibliotheken herangezogen werden.

FEIG ELECTRONIC GmbH verzichtet darauf, Informationen zu OBID®-Lesern in verschiedenen Handbüchern mehrfach darzustellen oder Querverweise auf bestimmte Seitenzahlen eines anderen Dokumentes einzubauen. Dies ist durch die ständige Aktualisierung der Handbücher notwendig und vermeidet Irrtümer durch Informationen in veralteten Dokumenten. Dem Anwender dieser Bibliothek sei deshalb empfohlen, sich ständig zu vergewissern, dass er die aktuellen Handbücher vorliegen hat. Diese kann er selbstverständlich jederzeit bei FEIG ELECTRONIC GmbH anfordern.

Wichtige Hinweise:

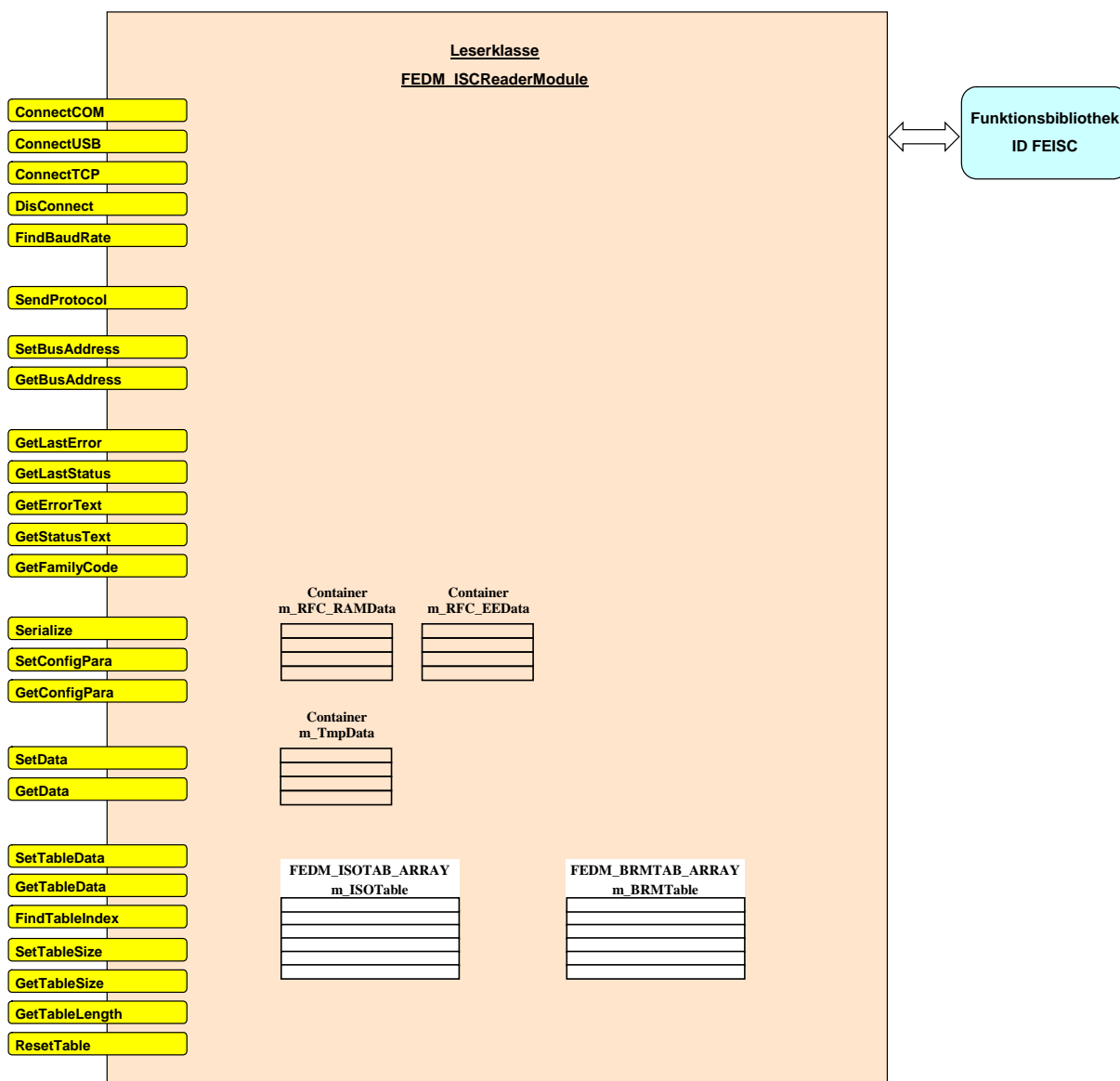
Sie dürfen diese Bibliothek nur verwenden, wenn Sie zuvor den umseitig abgedruckten Lizenzbestimmungen zugestimmt haben.

Sourcecode ist von jedermann veränderbar. Arbeiten Sie deshalb nur mit Bibliotheken, die Sie direkt von FEIG ELECTRONIC GmbH erhalten haben. Unabhängig davon ist die Weitergabe des Sourcecodes verboten.

1. Übersicht

Der Support für OBID i-scan® und OBID® classic-pro Leser besteht aus der von den Basisklassen FEDM_DataBase und FEDM_ISCReader abgeleiteten Leserklasse FEDM_ISCReaderModule und zwei in dieser Leserklasse integrierten Tabellen, die als Array von Klassen des Typs FEDM_ISOTabItem, bzw FEDM_BRMTabItem angelegt sind. Diese Dokumentation ist der zweite Teil zur Dokumentation der Klassenbibliothek ID FEDM, deren Konzept und Basisklassen im Dokument H10102-xd-ID-B beschrieben ist.

Das Komponenten-Diagramm zeigt in der Übersicht die wichtigsten Methoden und Datencontainer sowie die Tabellen.



Die Leserklasse unterstützt alle OBID i-scan® und OBID® classic-pro Leser, deshalb sind nicht alle Optionen der Klasse mit allen Lesertypen nutzbar.

Ergänzt wird die Leserkasse durch Klassen für externe Funktionseinheiten: FEDM_ISCFunctionUnit, FedmIsScExternalIO und FedmIsScPeopleCounter.

Für die effiziente Programmierung der Transponder-Kommunikation wurde das Konzept *TagHandler-Klassen* entwickelt, die eine Sammlung von Proxy-Klassen für eine große Anzahl von Transpondertypen darstellt. Jede TagHandler-Klasse bietet ein direktes Kommunikations-API für den identifizierten Transpondertyp an (s. Beispiel EPC Class 1 Gen 2).

FedmIsScTagHandler_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
+FedmIsScTagHandler_EPC_Class1_Gen2(uiTabIndex : unsigned int, uiTagHandlerType : unsigned int, pTabItem : FEDM_ISOTabItem *) +FedmIsScTagHandler_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

Beispiel: TagHandler-Klasse für EPC Class 1 Gen 2

2. Änderungen gegenüber der Vorversion

- TagHandler-Support für ISO 14443 Transponder mit 10-Byte UID
- Bugfix für EPC Class1 Gen2 Transponder mit Extended PC
- Android Support auf Anfrage
- ISO 15693: [0x2C] Get Multiple Block Security Status im extended addressed mode: bugfix für empfangene Daten oberhalb der Blockadresse 255
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

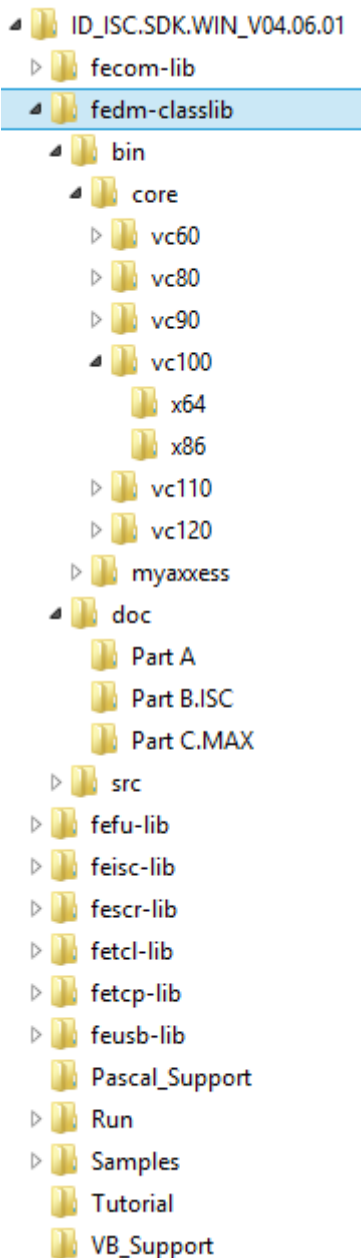
Bitte beachten Sie auch die Änderungshistorie im Anhang.

3. Installation

Das Supportpaket wird in der Regel mit einem Software Development Kit (SDK) ausgeliefert. Kopieren Sie das SDK unter Beibehaltung der Verzeichnisstruktur in ein Verzeichnis Ihrer Wahl.

Die Dateien dieses Supportpakets finden sich im Verzeichnis fedm-classlib.

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



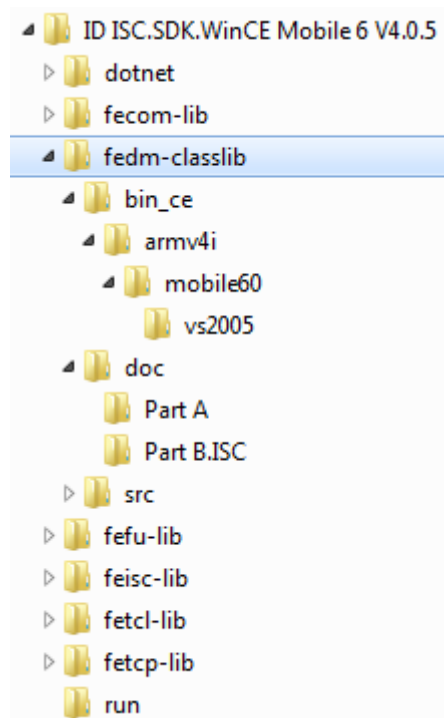
Wenn eigene Projekte nicht im SDK-Verzeichnis angelegt werden sollen, dann empfiehlt sich folgende Vorgehensweise:

- Kopieren Sie alle benötigten DLLs aus dem Run-Verzeichnis in das Verzeichnis des Anwendungsprogramms.
- Referenzieren Sie auf alle benötigten LIB-Dateien direkt im SDK-Verzeichnis (empfohlen) oder Kopieren Sie die LIBs aller benötigten Bibliotheken in das Projekt- oder LIB-Verzeichnis.
- Referenzieren Sie auf die Verzeichnisse der Headerdateien im SDK-Verzeichnis (empfohlen) oder Kopieren Sie Headerdateien aller benötigten Bibliotheken in das Projekt- oder INCLUDE-Verzeichnis. Hinweis: die Verzeichnisstruktur in src muss erhalten bleiben.
- Zusätzlich werden Dateien der myAXXESS-Bibliothek installiert. Die Beschreibung dieser ergänzenden Bibliothek finden Sie im Dokument Teil C.MAX (H90080-xe-ID-B.DOC).

Nach der Installation findet man im Verzeichnis fedm-classlib folgende Dateien:

Dateien im Unterverzeichnis src	Beschreibung
FedmlscCore.h	enthält alle Includes und Präprozessor-Definitionen
Dateien im Unterverzeichnis bin\core\vcxx\x86	Beschreibung
FedmlscCoreVC60.dll/.lib	mit Visual Studio 6 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 6.0
FedmlscCoreVC60d.dll/.lib	mit Visual Studio 6 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 6.0
FedmlscCoreVC80.dll/.lib	mit Visual Studio 2005 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 8.0
FedmlscCoreVC80d.dll/.lib	mit Visual Studio 2005 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 8.0
FedmlscCoreVC90.dll/.lib	mit Visual Studio 2008 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 9.0
FedmlscCoreVC90d.dll/.lib	mit Visual Studio 2008 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 9.0
FedmlscCoreVC100.dll/.lib	mit Visual Studio 2010 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 10.0
FedmlscCoreVC100d.dll/.lib	mit Visual Studio 2010 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 10.0
FedmlscCoreVC110.dll/.lib	mit Visual Studio 2012 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 11.0
FedmlscCoreVC110d.dll/.lib	mit Visual Studio 2012 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 11.0
FedmlscCoreVC120.dll/.lib	mit Visual Studio 2013 kompilierte 32-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 12.0
FedmlscCoreVC120d.dll/.lib	mit Visual Studio 2013 kompilierte 32-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 12.0
Dateien im Unterverzeichnis bin\core\vcxx\x64	Beschreibung
FedmlscCoreVC100.dll/.lib	mit Visual Studio 2010 kompilierte 64-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 10.0
FedmlscCoreVC100d.dll/.lib	mit Visual Studio 2010 kompilierte 64-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 10.0
FedmlscCoreVC110.dll/.lib	mit Visual Studio 2012 kompilierte 64-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 11.0
FedmlscCoreVC110d.dll/.lib	mit Visual Studio 2012 kompilierte 64-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 11.0
FedmlscCoreVC120.dll/.lib	mit Visual Studio 2013 kompilierte 64-Bit Bibliothek (Release-Version), kompatibel zur MFC-Version 12.0
FedmlscCoreVC120d.dll/.lib	mit Visual Studio 2013 kompilierte 64-Bit Bibliothek (Debug-Version), kompatibel zur MFC-Version 12.0

3.2. Windows CE



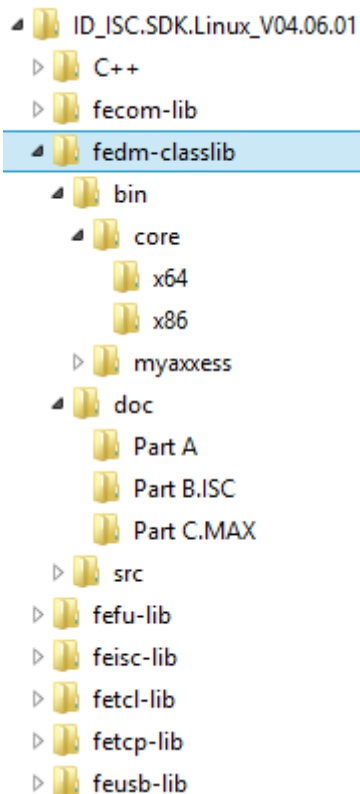
Wenn eigene Projekte nicht im SDK-Verzeichnis angelegt werden sollen, dann empfiehlt sich folgende Vorgehensweise:

- Kopieren Sie alle benötigten DLLs aus dem Run-Verzeichnis in das Verzeichnis des Anwendungsprogramms.
- Referenzieren Sie auf alle benötigten LIB-Dateien direkt im SDK-Verzeichnis (empfohlen) oder Kopieren Sie die LIBs aller benötigten Bibliotheken in das Projekt- oder LIB-Verzeichnis.
- Referenzieren Sie auf die Verzeichnisse der Headerdateien im SDK-Verzeichnis (empfohlen) oder Kopieren Sie Headerdateien aller benötigten Bibliotheken in das Projekt- oder INCLUDE-Verzeichnis. Hinweis: die Verzeichnisstruktur in src muss erhalten bleiben.

Nach der Installation findet man im Verzeichnis fedm-classlib folgende Dateien:

Dateien im Unterverzeichnis src	Beschreibung
FedmlscCore.h	enthält alle Includes und Präprozessor-Definitionen
Dateien im Unterverzeichnis bin_ce\...	Beschreibung
FedmlscCoreCE.dll/.lib	mit Visual Studio kompilierte Bibliothek, kompatibel zur gewählten Plattform

3.3. 32- und 64-Bit Linux



Zur Installation gibt es zwei Optionen:

Option 1: Falls eine install.sh im SDK-Verzeichnis vorliegt, führen Sie diese aus (./install.sh). Damit werden alle Bibliotheken in das Verzeichnis /usr/lib bzw. /usr/lib64 kopiert und alle symbolischen Links angelegt. Die Headerdatei können Sie in ein Verzeichnis Ihrer Wahl kopieren.

Option 2: Kopieren Sie die Dateien dieses Supportpakets in Verzeichnisse Ihrer Wahl und Erzeugen Sie symbolische Links auf die Bibliotheksdatei libFedmlscCore.so.x.y.z¹ im Verzeichnis /usr/lib bzw. /usr/lib64 durch folgende Aufrufe:

```
cd /usr/lib (für 64 Bit : /usr/lib64)
```

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

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

```
ldconfig
```

Nach der Installation findet man im Verzeichnis fedm-classlib folgende Dateien:

Dateien im Unterverzeichnis src	Beschreibung
FedmlscCore.h	enthält alle Includes und Präprozessor-Definitionen
Dateien im Unterverzeichnis bin/core/x86	Beschreibung
libFedmlscCore.so.x.y.z	mit GCC kompilierte Bibliothek für Linux

x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei

Die kompilierte Version wurde gegen die LibC V6 und LibStdC++ V6 gelinkt

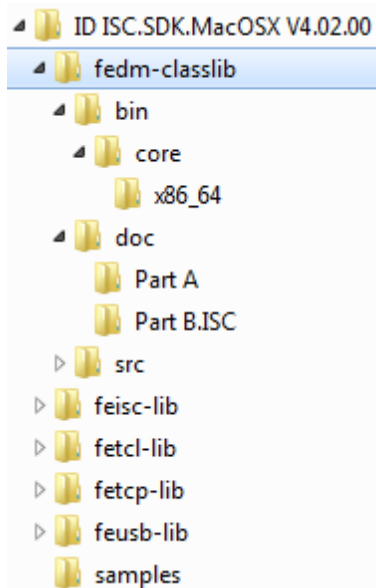
Anmerkung:

x86 : Die Bibliothek wurde unter SuSE Linux 11.1 mit der GNU Compiler Collection V4.3.2 erstellt.

X64: Die Bibliothek wurde unter SuSE Linux 11.2 mit der GNU Compiler Collection V4.4.1 erstellt.

¹ x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei

3.4. 64-Bit Mac OS X



Zur Installation gibt es zwei Optionen:

Option 1: Falls eine `install.sh` im SDK-Verzeichnis vorliegt, führen Sie diese aus (`./install.sh`). Damit werden alle Bibliotheken in das Verzeichnis `/usr/local/lib` kopiert und alle symbolischen Links angelegt. Die Headerdatei können Sie in ein Verzeichnis Ihrer Wahl kopieren.

Option 2: Kopieren Sie die Dateien dieses Supportpakets in Verzeichnisse Ihrer Wahl und Erzeugen Sie symbolische Links auf die Bibliotheksdatei `libFedmIscCore.x.y.z.dylib`¹ im Verzeichnis `/usr/local/lib` durch folgende Aufrufe:

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

```
In -s libFedmIscCore.x.y.z.dylib libFedmIscCore.x.dylib
```

```
In -s libFedmIscCore.x.dylib libFedmIscCore.dylib
```

Anmerkung: Die Bibliothek wurde unter Mac OS X V10.7.3 mit Xcode V4.3.2 erstellt. Die Bibliothek ist mit der Architektur `x86_64` kompatibel.

¹ x.y.z repräsentiert die Versionsnummer der Bibliotheksdatei

3.5. Quellcode

Mit der Installation wird der Quellcode der Bibliothek in Unterverzeichnisse von src kopiert.

src\impl\core	Beschreibung
FEDM.h, FEDM_ISC.h and FEDM_Xml.h	enthalten Includes, Konstanten und Macros
FEDM_Base.h/.cpp	Basisklasse für Leserklasse
FEDM_DataBase.h/.cpp	von FEDM_Base abgeleitete, abstrakte Basisklasse für Leserklasse
FEDM_Functions.h/.cpp	globale Funktionen
FEDM_XmlBase.h/.cpp	Basisklasse für Serialisierung im XML-Format
FEDM_XmlReaderCfgDataModul.h/.cpp	spezialisierte Klasse für Serialisierung der Leserkonfiguration im XML-Format
FEDM_XmlReaderCfgProfileModul.h/.cpp	spezialisierte Klasse für Serialisierung einer Profilkonfiguration im XML-Format

src\impl\core\i_scan	Beschreibung
FEDM_ISCReader.h/.cpp	von FEDM_DataBase abgeleitete, spezialisierte Leserklasse
FEDM_ISCReaderModule.h/.cpp	von FEDM_ISCReader abgeleitete Leserklasse mit High-Level Methoden
FEDM_ISCReaderInfo.h	Struktur mit Informationen zum Leser
FEDM_ISCReaderDiagnostic.h	Struktur mit Zustandsdaten vom Leser
FEDM_TabItem.h	Basisklasse für Tabellen
FEDM_BRMTabItem.h/.cpp	Tabellenklasse für Buffered Read Mode
FEDM_ISOTabItem.h/.cpp	Tabellenklasse für Host-Commands
FEDM_ISCReaderID.h	Zugriffskonstanten für temporäre Kommunikationsparameter
FEDM_ISCReaderConfig.h/.cpp	Dateien mit Zugriffskonstanten für Leserkonfigurationen

src\impl\core\i_scan\classic_pro	Beschreibung
FEDM_ISCReaderConfig_XXX.cpp	Dateien mit Zugriffskonstanten für Leserkonfigurationen

src\impl\core\i_scan\function_unit	Beschreibung
FEDM_ISCFunctionUnit.h/.cpp	spezialisierte Klasse für Funktionseinheiten (Multiplexer, Dynamischer Antennentuner)
FEDM_ISCFunctionUnitID.h	Zugriffskonstanten für temporäre Kommunikationsparameter für Funktionseinheiten

src\impl\core\i_scan\peripheral_devices	Beschreibung
FedmlscPeripheralDevice.h/.cpp	Basisklasse für externe Geräte (z.B. People-Counter)
FedmlscPeopleCounter.h/.cpp	von FedmlscPeripheralDevice abgeleitete People-Counter-Klasse
FedmlscExternalIO.h/.cpp	von FedmlscPeripheralDevice abgeleitete External-IO Klasse

src\impl\core\i_scan>tag_handler	Beschreibung
FedmlscTagHandler.h/.cpp	TagHandler-Basisklasse

src\impl\core\i_scan\tag_handler	Beschreibung
FedmlscTagHandler_Includes.h	Include-Datei
FedmlscTagHandler_XXX.h/.cpp	spezialisierte TagHandler-Klassen

src\impl\core\i_scan\utility	Beschreibung
FedmlscReport_ReaderInfo.h/.cpp	Reportklasse für wichtige Informationen zum Leser
FedmlscReport_ReaderDiagnostic.h/.cpp	Reportklasse mit wichtigen Zustandsdaten vom Leser

3.6. Abhängigkeiten

Die Klassenbibliothek FEDM ist von den Funktionsbibliotheken für die Kommunikations-Schnittstellen (FECOM, FEUSB, FETCP) und die Leserfamilie (FEISC, FEFU, FETCL) abhängig. Diese Bibliotheken sind Bestandteil des jeweiligen SDK und müssen auch auf dem Zielsystem installiert werden.

Die Klasse FEDM_ISCReader enthält die Methode *EvaLibDependencies* zur Verifikation der Versionsnummern der abhängigen Funktionsbibliotheken. Es wird empfohlen, diese Methode in der Applikation nach dem Programmstart einmal aufzurufen.

4. Einbindung in das Anwendungsprogramm

4.1. Unterstützte Entwicklungsumgebungen

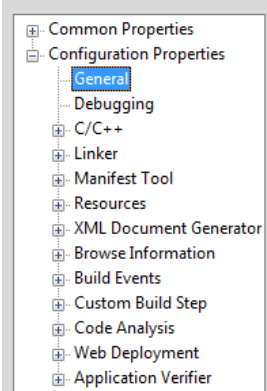
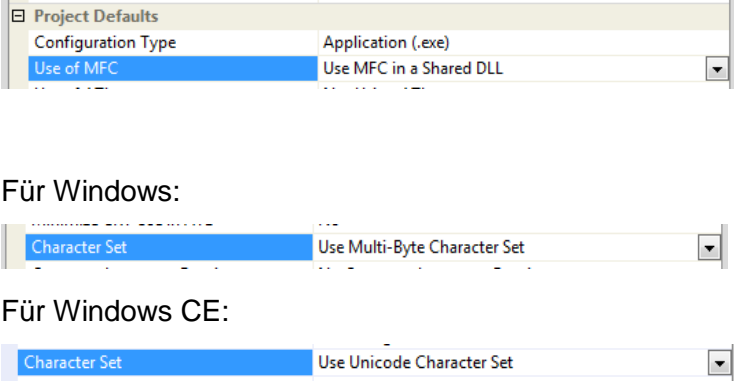
Betriebssystem	Entwicklungsumgebung	Unterstützung
Windows XP / Vista / 7 / 8	Visual Studio 6	auf Anfrage
	Visual Studio 2005 / 2008 / 2010 / 2012 / 2013	ja, ab Professional Version
	Borland C++ Builder	auf Anfrage
	Embarcadero C++ Builder	auf Anfrage
Windows CE	eMbedded Visual C++ 4	nein
	Visual Studio 2005 / 2008	ja, ab Professional Version
Linux	GCC	ja, für 32-Bit Projekte
Mac OS X	GCC	ja, für Projekte mit x86_64 Architektur
	Xcode ≥ V4.3.2	ja, für Projekte mit x86_64 Architektur

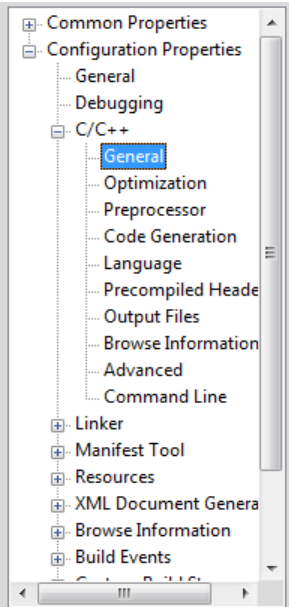
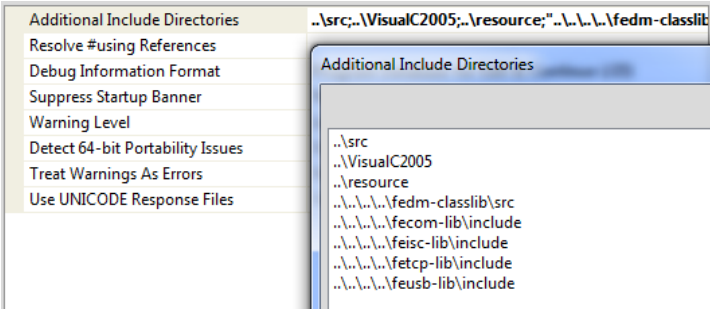
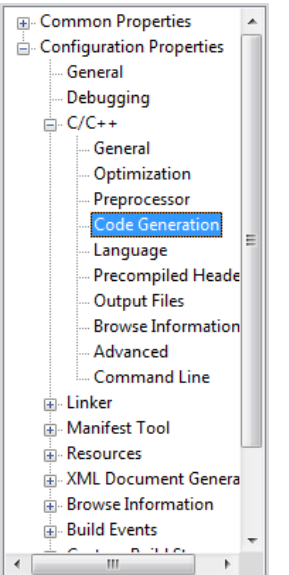
4.2. Der schnelle Weg

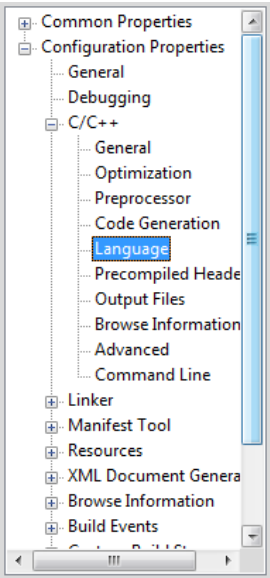

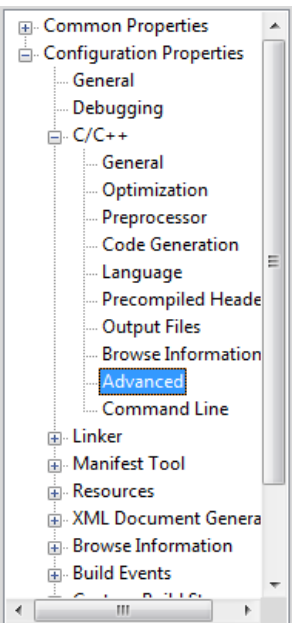
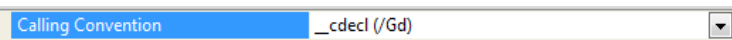
Bei Verwendung der kompilierten Bibliotheken sind alle Präprozessor-Definitionen in der Include-Datei FedmIscCore.h zusammengefasst und es ist ausreichend, ausschließlich diese Include-Datei einzubinden.

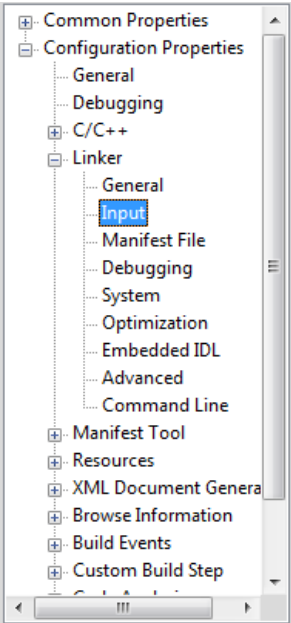
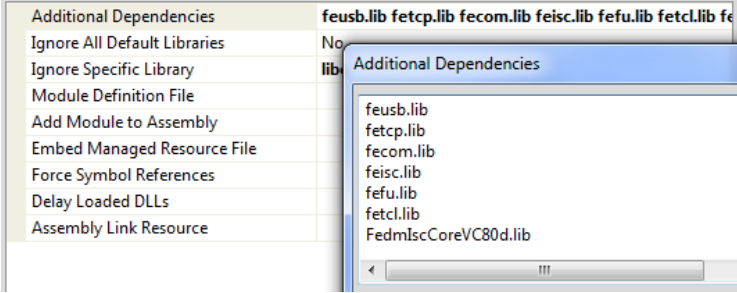
4.2.1. Einbindung in Visual Studio

Wichtige Einstellungen (Beispiel Visual Studio 2005):

Projekt Gruppe	Einstellung
	 <p>Für Windows:</p> <p>Für Windows CE:</p>

Projekt Gruppe	Einstellung				
	<p>Alle Verzeichnisse zu den Headerdateien müssen eingetragen werden.</p> 				
	<p>Für Release-Version:</p> <table border="1" data-bbox="635 929 1433 974"> <tr> <td>Runtime Library</td><td>Multi-threaded DLL (/MD)</td></tr> </table> <p>Für Debug-Version:</p> <table border="1" data-bbox="635 1108 1433 1153"> <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)				

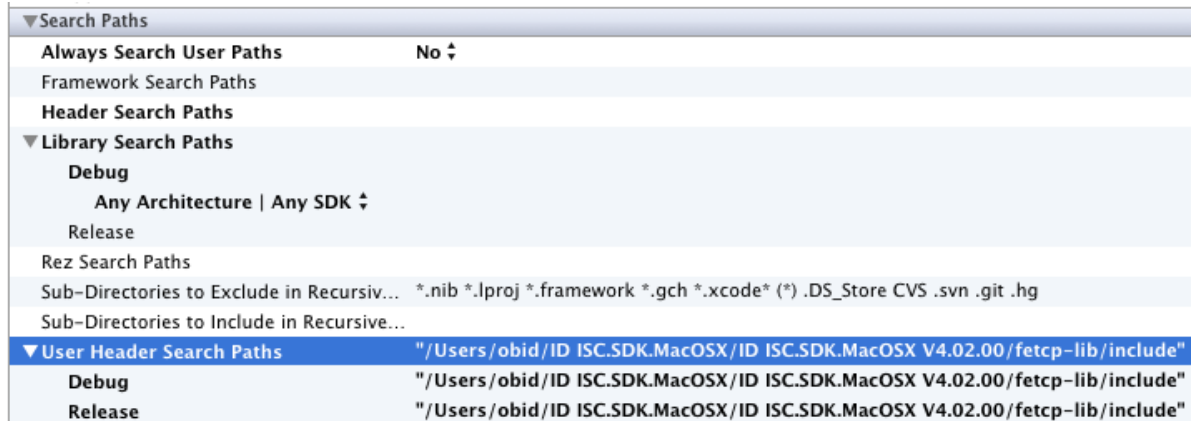
Projekt Gruppe	Einstellung
	
	

Projekt Gruppe	Einstellung
	<p>Die .lib-Dateien der abhängigen Basisbibliotheken und FedmIscCoreVCxx.lib (Release-Version) bzw. FedmIscCoreVCxxd.lib (Debug-Version) müssen eingebunden werden.</p> 

4.2.2. Einbindung in Xcode

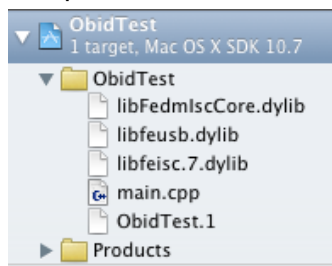
1. Pfad zu benötigten Headerdateien in den Projekteinstellungen (Kategorie Search Paths und dort für User Header Search Paths) hinzufügen

Beispiel:



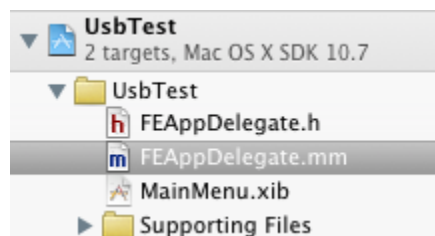
2. die benötigten DYLIB-Dateien per Drag-and-Drop dem Projekt hinzufügen

Beispiel:



3. In Objective-C Projekten müssen für alle Quelldateien, die C++ Klassen aus FEDM einbinden, die Dateierweiterung von .m in .mm geändert werden. Dadurch wird sichergestellt, dass der Compiler C++ Erweiterungen aktiviert.

Beispiel:



4.3. Manuelle Einbindung

Falls man den Quellcode der Bibliothek direkt in die Applikation einbinden will oder muss, sind nachfolgend die notwendigen Schritte beschrieben.

Die manuelle Einbindung bietet für Visual Studio Projekte die Option der statischen Bindung der MFC.

4.3.1. Include-Dateien

Fügen Sie die Include-Datei FedmlscCore.h dem Projekt hinzu. Diese setzt die nachfolgend beschriebenen Präprozessor-Definitionen und bindet weitere Include-Dateien aus dem Unterverzeichnis src ein.

Wichtig: Kommentieren Sie die Zeile `#define _FEDM_DLL` aus bzw. verwenden Sie `_FEDM_NO_DLL`. Die Präprozessor-Definition `_FEDM_DLL` darf nur gesetzt werden, wenn die vorkompilierte Bibliothek zur Laufzeit eingebunden wird

4.3.2. Quellcode-Dateien

Fügen Sie alle Quellcode-Dateien aus den Unterverzeichnissen

```
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
```

dem Projekt hinzu.

Zusätzlich verlangt die Entwicklungsumgebung MS Visual C++ folgende Projekteinstellungen:

Die *.cpp Dateien der Bibliothek FEDM, die zuvor dem Projekt hinzugefügt wurden, müssen von der automatischen Verwendung der vorkompilierten Headerdatei freigestellt werden. Dies stellt man im Register C/C++, in der Kategorie "Vorkompilierte Header" ein, indem man die Option "Vorkompilierte Header nicht verwenden" auswählt.

4.3.3. Notwendige Projekt-Einstellungen

Generell muss für jede verwendete Kommunikations-Bibliothek eine Präprozessor-Definition gesetzt werden (in FedmlscCore.h enthalten):

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

4.3.4. Windows

Unter Windows muss die Präprozessor-Definition **_FEDM_WINDOWS** gesetzt werden (in FedmlscCore.h enthalten).

4.3.5. Linux

Unter Linux muss die Präprozessor-Definition **_FEDM_LINUX** gesetzt werden (in FedmlscCore.h enthalten).

4.3.6. Optionale Projekt-Einstellungen

Für den XML-Support (Serialisierung der Konfigurationsdaten des Lesers in eine Datei) muss die Präprozessor-Definition **_FEDM_XML_SUPPORT** gesetzt werden (in FedmlscCore.h enthalten).

Zur Einbindung der TagHandler-Klassen muss die Präprozessor-Definition **_FEDM_TAG_HANDLER** gesetzt werden (in FedmlscCore.h enthalten). Einschränken kann man die Menge der TagHandler-Typen mit **_FEDM_NO_TAG_HANDLER_EPC_C1_G2**, **_FEDM_NO_TAG_HANDLER_ISO14443**, **_FEDM_NO_TAG_HANDLER_ISO15693** und **_FEDM_NO_TAG_HANDLER_ISO18000_3M3**.

Für Projekte, die die Microsoft Foundation Classes (MFC) verwenden und innerhalb der FEDM nutzen wollen (im Wesentlichen CString), muss die Präprozessor-Definition **_FEDM_MFC_SUPPORT** gesetzt werden (in FedmlscCore.h enthalten).

Für Projekte, die ausschließlich statische Bindungen der abhängigen Bibliotheken FECOM, FEUSB und FETCP bevorzugen, muss die Präprozessor-Definition **_FEDM_SUPPORT_SLINK** gesetzt werden (**nicht** in FedmlscCore.h enthalten).

Falls man die Bibliothek auf das nur Notwendige reduzieren möchte bei gleichzeitiger Verwendung der Include-Datei FedmlscCore.h, setzt man Präprozessor-Definitionen zum Ausschließen von nicht benötigten Teilen:

_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**

Alle genannten Präprozessor-Definitionen erlauben den Zuschnitt der Bibliothek nach eigenen Anforderungen.

5. Installation auf dem Zielrechner

Zusammen mit den Dateien der Applikation ist die Laufzeitdatei der Bibliothek FedmlscCore (falls dynamisch gebunden) und sind die Laufzeitdateien der Funktionsbibliotheken FECOM, FEUSB, FETCP, FEISC, FETCL und FEFU auf dem Zielrechner zu installieren.

5.1. 32-Bit Bibliotheken für 32- und 64-Bit Windows

Es wird empfohlen, die Bibliotheksdateien im Verzeichnis der Applikation zu halten. Damit vermeidet man Versionskonflikte mit späteren Installationen, die ebenfalls diese Bibliotheksdateien, aber möglicherweise in anderen Versionsständen, installieren.

Die Bibliotheksdateien FedmlscCoreVCxxx.dll hängen von neueren C/C++ Laufzeit-Bibliotheken ab, die in der Regel auf dem Zielrechner nicht vorhanden sind. Sie müssen somit installiert werden. Mit dem Visual Studio werden sogenannte Merge-Module mitgeliefert, die man in ein Setup-Projekt aufnehmen kann und die die MFC-Bibliotheken installieren. Die nachfolgende Tabelle listet für die jeweilige FedmlscCore-Bibliothek das zu installierende Merge-Modul auf.

Bibliotheksdatei	MFC-Version	Merge-Module
FedmlscCoreVC60.dll	Version 6.0	MFC-Bibliothek ist ab Windows 2000 installiert
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
FedmlscCoreVC120.dll	Version 12.0 (12.0.21005.1)	Microsoft_VC120_MFC_x86.msm Microsoft_VC120_CRT_x86.msm

Eine Alternative ist die Installation der Visual C++ Laufzeitbibliotheken über das Internet-Portal von Microsoft. Dort findet sich für jede MFC-Version eine Datei vcredist_x86.exe zum herunterladen.

¹ Microsoft Sicherheitsbulletin Artikel-ID: 2538218 vom 14. Juni 2011

² Microsoft Sicherheitsbulletin Artikel-ID: 2538243 vom 14. Juni 2011

³ Microsoft Sicherheitsbulletin Artikel-ID: 2542054 vom 14. Juni 2011

- 1. Hinweis:** Die Datei vcredist_x86.exe muss mindestens die oben angegebene Versionsnummer enthalten.
- 2. Hinweis:** Merge-Module müssen mit Windows-Update aktualisiert werden
- 3. Hinweis:** Die Debug-Versionen von FEDM, gekennzeichnet durch ein zusätzliches **d** am Ende des Bibliotheksnamens (z.B. FedmlscCoreVC80d.dll) können nicht auf einen Kundenrechner installiert werden, weil die Merge-Module keine Debug-Versionen der MFC installieren.

5.2. 64-Bit Bibliotheken für 64-Bit Windows

Es wird empfohlen, die Bibliotheksdateien im Verzeichnis der Applikation zu halten. Damit vermeidet man Versionskonflikte mit späteren Installationen, die ebenfalls diese Bibliotheksdateien, aber möglicherweise in anderen Versionsständen, installieren.

Die Bibliotheksdatei FedmlscCoreVC1xx.dll hängt von neueren C/C++ Laufzeit-Bibliotheken ab, die in der Regel auf dem Zielrechner nicht vorhanden sind. Sie müssen somit installiert werden. Mit dem Visual Studio werden sogenannte Merge-Module mitgeliefert, die man in ein Setup-Projekt aufnehmen kann und die die MFC-Bibliotheken installieren. Die nachfolgende Tabelle listet für die jeweilige FedmlscCore-Bibliothek das zu installierende Merge-Modul auf.

Bibliotheksdatei	MFC-Version	Merge-Module
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
FedmlscCoreVC120.dll	Version 12.0 (12.0.21005.1)	Microsoft_VC120_MFC_x64.msm Microsoft_VC120_CRT_x64.msm

Eine Alternative ist die Installation der Visual C++ Laufzeitbibliotheken über das Internet-Portal von Microsoft. Dort findet sich für jede MFC-Version eine Datei vcredist_x64.exe zum herunterladen.

- 1. Hinweis:** Die Datei vcredist_x64.exe muss mindestens die oben angegebene Versionsnummer enthalten.
- 2. Hinweis:** Merge-Module müssen mit Windows-Update aktualisiert werden
- 3. Hinweis:** Die Debug-Versionen von FEDM, gekennzeichnet durch ein zusätzliches **d** am Ende des Bibliotheksnamens (z.B. FedmlscCoreVC100d.dll) können nicht auf einen Kundenrechner installiert werden, weil die Merge-Module keine Debug-Versionen der MFC installieren.

¹ Microsoft Sicherheitsbulletin Artikel-ID: 2542054 vom 14. Juni 2011

5.3. 32- und 64-Bit Linux

Auf dem Zielrechner sind die gleichen Installationsvorgänge durchzuführen, wie unter [3.3. 32- und 64-Bit Linux](#) für die Installation auf dem Entwicklungsrechner. Allerdings dürfen nur die Laufzeitdateien *.so auf dem Zielrechner installiert werden.

Die Beschreibung der Installation der abhängigen Funktionsbibliotheken findet sich in den jeweiligen Handbüchern zu den Bibliotheken.

5.4. 64 Bit Mac OS X

Auf dem Zielrechner sind die gleichen Installationsvorgänge durchzuführen, wie unter [3.4. 64-Bit Mac OS X](#) für die Installation auf dem Entwicklungsrechner. Allerdings dürfen nur die Laufzeitdateien *.dylib auf dem Zielrechner installiert werden.

Die Beschreibung der Installation der abhängigen Funktionsbibliotheken findet sich in den jeweiligen Handbüchern zu den Bibliotheken.

6. Klassenbeschreibung

Die Klassenbibliothek ID FEDM ist einem ständigen Anpassungsprozeß ausgesetzt. Wir werden uns bemühen, den dokumentierten Stand beizubehalten. Änderungen sind trotzdem nicht auszuschließen.

6.1. FEDM_ISCReader

Die Klasse FEDM_ISCReader basiert auf der abstrakten Basisklasse FEDM_DataBase und erbt damit die allgemeine, typ-unabhängige Schnittstelle. Die abstrakten Schnittstellen-Methoden sind in FEDM_ISCReader implementiert, so dass mit einer Instanz dieser Klasse gearbeitet werden kann. Alternativ kann man, aufbauend auf dieser Klasse, seine eigene Leserklass ableiten und derart gestalten, dass man weitere Funktionalitäten hinzufügt oder Methoden überschreibt um ein anderes Verhalten implementieren zu können.

6.1.1. Implementierte Datencontainer

Datencontainer	Beschreibung
m_RFC_EEData	für Konfigurations-Parameter des Lesers (für alle OBID i-scan® und OBID® classic-pro Lesertypen) mit RF-Controller
m_RFC_RAMData	für temp. Konfigurations-Parameter des Lesers (für alle OBID i-scan® und OBID® classic-pro Lesertypen) mit RF-Controller
m_ACC_EEData	für zusätzliche Konfigurations-Parameter des Lesers mit AC-Controller (ID ISC.LRU2000)
m_ACC_RAMData	für zusätzliche temp. Konfigurations-Parameter des Lesers mit AC-Controller (ID ISC.LRU2000)
m_TmpData	für allgemeine temp. Protokolldaten (für alle OBID i-scan® und OBID® classic-pro Lesertypen)

Die Größe der Datencontainer wird statisch im Konstruktor der Klasse festgelegt. Alle Datencontainer werden im Konstruktor mit 0x00 initialisiert.

Alle nicht aufgeführten Datencontainer behalten die Länge 0 und sind nicht nutzbar.

6.1.2. Implementierte Tabellen

Tabelle	Beschreibung
m_ISOTable	unterstützt den Datenaustausch mit Transpondern über die ISO-Host-Commands.
m_BRMTable	Sammelt die Daten, die ein Long-Range-Reader vom Typ ISC.LRxxx im Buffered-Read-Modus liefert.

Die Größe der Tabellen muss vor der ersten Verwendung mit der Methode SetTableSize eingestellt werden! Die Größe wird bestimmt durch die maximale Anzahl an Transpondern, die sich gleichzeitig im Antennenfeld des Lesers befinden.

In Anwendungen, die den Buffered-Read-Mode des Lesers nicht nutzen, muss auch keine Tabellengröße für m_BRMTable festgelegt werden. Dasselbe gilt natürlich auch umgekehrt für die Tabelle m_ISOTable.

6.1.3. Methoden (public)

Methode	Beschreibung
EvalLibDependencies	Methode verifiziert die Kompatibilität mit abhängigen Funktionsbibliotheken.
SendProtocol	Die zentrale Kommunikationsmethode. Nähere Beschreibung in 6.4.3. Beispiele für die Verwendung der Methode SendProtocol .
GetLastProt	Methode zum Abrufen des letzten Sende- oder Empfangs-Protokolls. Mit sID="SEND" erhält man das letzte Sendeprotokoll Mit sID="SENDSTR" erhält man das letzte Sendeprotokoll mit vorangestelltem Datum/Uhrzeit Mit sID="REC" erhält man das letzte Empfangsprotokoll Mit sID="RECSTR" erhält man das letzte Empfangsprotokoll mit vorangestelltem Datum/Uhrzeit
FindBaudrate	Methode sucht an der durch den Port-Handle (wird im Leserobjekt in FEISC gespeichert) identifizierbaren Schnittstelle nach einem Leser und ermittelt die Baudrate und den Protokollrahmen. Diese Methode kann nicht mit dem USB-Port, bzw. TCP/IP-Port, genutzt werden.
SetReaderType	Setzt den Typ des Lesers, der von dem Laufzeitobjekt repräsentiert werden soll. Bevorzugt sollte allerdings die Methode ReadReaderInfo der Klasse FEDM_ISCReaderModule sofort nach dem Verbindungsaufbau aufgerufen werden. Diese setzt dann ebenfalls den Lesertyp.
SetPortHnd	Setzt den Port-, Device oder Socket-Handle als Parameter im Leserobjekt in der FEISC Bibliothek.
GetPortHnd	Ermittelt den Port-, Device oder Socket-Handle des Leserobjekts in der FEISC Bibliothek.
GetReaderName	Methoden zur Rückgabe der Readerbezeichnung
GetTagName	Gibt den Kurzstring zum Transponder zurück.
GetReaderInfo	Gibt einen Pointer auf die FEDM_ISC_READER_INFO-Struktur zurück.
GetReaderDiagnostic	Gibt einen Pointer auf die FEDM_ISC_READER_DIAGNOSTIC-Struktur zurück.
SetProtocolFrameSupport	Stellt den Protokolltyp für den Protokollverkehr ein (voreingestellt: Standard Protocol Frame). Übergabewerte sind: FEDM_PRT_FRAME_STANDARD FEDM_PRT_FRAME_ADVANCED
GetProtocolFrameSupport	Ermittelt den Protokolltyp für den Protokollverkehr.
DisableReadCfgBeforeWriteCfg	Mit dieser Methode kann man die Kontrolle vor dem Schreiben eines Konfigurationsblocks in den Leser abschalten, die dafür sorgt, dass erst ein Lesevorgang vor dem Schreibvorgang zu erfolgen hat.
EnableTagHandler	Mit dieser Methode schaltet man zur Laufzeit den TagHandler-Support ein.

Methode	Beschreibung
GetLastError	Ermittelt den letzten Fehlercode, der an der Stelle FEDM_ISC_TMP_LAST_ERROR im Datencontainer TempData hinterlegt ist.
GetLastStatus	Ermittelt das Statusbyte des letzten Protokolls, das an der Stelle FEDM_ISC_TMP_LAST_STATE im Datencontainer TempData hinterlegt ist.
GetErrorText	Ermittelt zum übergebenen Fehlercode einen Text. Der Fehlercode kann auch aus dem Bereich der Funktionssammlung ID FEISC oder der darunter liegenden Kommunikationsbibliotheken kommen. Die Sprache des Textes ist mit der Methode SetLanguage der Basisklasse FEDM_Base einstellbar.
GetStatusText	Ermittelt zum übergebenen Statusbyte einen Kurztext. Die Sprache des Textes ist mit der Methode SetLanguage der Basisklasse FEDM_Base einstellbar.
Serialize	Hauptmethode für Serialisierung. Ermöglicht das Serialisieren der Containerdaten in Dateien. (s. 6.4.4. Die Serialisierung im XML-Format).
GetCommandPara	<p>Die zentrale (überladene) Methode zum Lesen eines Command-Parameters von einem Datencontainer der Klasse.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter einen String des Parameternamens aus dem Namespace ReaderCommand entsprechend den Angaben im Systemhandbuch des Lesers.</p>
SetCommandPara	<p>Die zentrale (überladene) Methode zum Schreiben eines Command-Parameters in einen Datencontainer der Klasse.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter einen String des Parameternamens aus dem Namespace ReaderCommand entsprechend den Angaben im Systemhandbuch des Lesers.</p>
GetConfigPara	<p>Die zentrale (überladene) Methode zum Lesen eines Konfigurationsparameters von einem Datencontainer der Klasse.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter einen String des Parameternamens aus dem Namespace ReaderConfig entsprechend den Angaben im Systemhandbuch des Lesers, sowie die Angabe, ob der Konfigurationsparameter im Speicherort RAM oder EEPROM lokalisiert ist.</p>
SetConfigPara	<p>Die zentrale (überladene) Methode zum Schreiben eines Konfigurationsparameters in einen Datencontainer der Klasse.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter einen String des Parameternamens aus dem Namespace ReaderConfig entsprechend den Angaben im Systemhandbuch des Lesers, sowie die Angabe, ob der Konfigurationsparameter im Speicherort RAM oder EEPROM lokalisiert ist..</p>
TestConfigPara	Überprüft, ob der Konfigurationsparameter (String des Parameternamens aus dem Namespace ReaderConfig) für den eingestellten Lesertyp definiert ist.
GetTableData	<p>Die zentrale (überladene) Methode zum Lesen eines Wertes oder von Datenblöcken aus einer Tabelle.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p>

Methode	Beschreibung
	<p>Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet. Der Parameter uiDataID trägt eine Kennung für den zu schreibenden Wert. Alle Zugriffskennungen sind in der Datei FEDM_ISC.h aufgelistet. Welche Zugriffskennungen unterstützt werden, kann man dem Anhang 7.3.4. Konstanten für uiDataID entnehmen.</p> <p>Alternativ kann über die Methode GetISOTableItem bzw. GetBRMTTableItem direkt auf die Tabellenelemente zugegriffen werden.</p>
SetTableData	<p>Die zentrale (überladene) Methode zum Schreiben eines Wertes oder von Datenblöcken in die Tabelle m_Table.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UCHAR-Array, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet. Der Parameter uiDataID trägt eine Kennung für den zu lesenden Wert. Alle Zugriffskennungen sind in der Datei FEDM_ISC.h aufgelistet. Welche Zugriffskennungen unterstützt werden, kann man dem Anhang 7.3.4. Konstanten für uiDataID entnehmen.</p> <p>Alternativ kann über die Methode GetISOTableItem bzw. GetBRMTTableItem direkt auf die Tabellenelemente zugegriffen werden.</p>
FindTableIndex	<p>Die zentrale (überladene) Methode zum Ermitteln des Tabellen-Indexes anhand eines Wertes, ausgehend von einem Start-Index.</p> <p>Diese Variante unterstützt die Datentypen: bool, UCHAR, UINT, __int64, CString bzw. AnsiString und STL-string.</p> <p>Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet. Der Parameter uiDataID trägt eine Kennung für den gesuchten Wert. Alle Zugriffskennungen sind in der Datei FEDM_ISC.h aufgelistet. Welche Zugriffskennungen unterstützt werden, kann man dem Anhang 7.3.4. Konstanten für uiDataID entnehmen.</p>
SetTableSize	<p>Stellt die Größe der Tabellen m_ISOTable bzw. m_BRMTTable ein und initialisiert jede Tabellenzeile mit 0. Eine nachträgliche Änderung der Größe ist möglich. Dabei geht der alte Inhalt verloren. Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet.</p> <p>Eine zweite, überladene Implementierung erlaubt zusätzlich mit erweiterten Parametern die von der Voreinstellung abweichende Dimensionierung der Datenfelder RxDB und TxDB. Dadurch kann man den Speicherbedarf an die Erfordernisse der Applikation bzw. der Transponder genauer anpassen.</p>
GetTableSize	<p>Ermittelt die eingestellte Größe der Tabellen m_ISOTable bzw. m_BRMTTable. Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet.</p>
GetTableLength	<p>Ermittelt die Anzahl gültiger Tabelleneinträge in m_ISOTable, bzw. m_BRMTTable. Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet.</p>
ResetTable	<p>Setzt die Tabelle m_ISOTable, bzw. m_BRMTTable zurück. Die Methode erwartet als Parameter eine Tabellen-ID, die zwischen den Tabellen m_ISOTable und m_BRMTTable unterscheidet.</p> <p>Es werden nur die Variablen m_iISOTableLength, bzw. m_iBRMTTableLength auf 0 gesetzt (die Daten der Tabelle werden nicht gelöscht). Mit dem zusätzlichen Parameter uiDataFlags = FEDM_ISC_DATA_ALL werden alle Datenfelder mit 0 initialisiert.</p> <p>Voreinstellung: keine Initialisierung der Datenfelder.</p>

Methode	Beschreibung
GetPeripheralDevices	<p>Gibt eine sortierte Liste detektierter externer Funktionseinheiten (PeopleCounter, ExternalIO) zurück. Die Detektion muss zuvor mit dem Protokoll [0x66] Reader Info bei Verwendung des Mode-Bytes 0x61 oder mit der Methode ReadReaderInfo() der Klasse FEDM_ISCReaderModule erfolgen.</p> <p>Die sortierte Liste enthält keine in die Antennenleitung eingeschleifte Funktionseinheiten (Antennentuner, Multiplexer).</p>
GetISOTableItem	<p>Für die Betriebsart Host-Modus: Methode gibt den Pointer eines Tabelleneintrags (enthält Daten zu einem Transponder) zurück, mit dem direkt auf alle Variablen des Transponders zugegriffen werden kann.</p> <p>Diese Methode ersetzt GetTableData.</p>
GetBRMTableItem	<p>Für die Betriebsarten Buffered-Read-Mode und Notification-Mode: Methode gibt den Pointer eines Tabelleneintrags (enthält Daten zu einem Transponder) zurück, mit dem direkt auf alle Variablen des Transponders zugegriffen werden kann.</p> <p>Diese Methode ersetzt GetTableData.</p>

6.2. FEDM_ISCReaderModule

Die Klasse FEDM_ISCReaderModule basiert auf der Leserklasse FEDM_ISCReader und enthält High-Level-Methoden für die verschiedenen Schnittstellentypen und für den Notification Mode. Es wird empfohlen, diese Leserklasse in der eigenen Applikation zu verwenden.

Mit einer Instanz von FEDM_ISCReaderModule wird automatisch auch ein Reader-Objekt in der Funktionsbibliothek FEISC erzeugt.

6.2.1. Methoden (public)

Methode	Beschreibung
ConnectCOMM	Öffnet eine serielle Schnittstelle mit der Einstellung Baudrate=38400, Frame=8E1, Timeout=3000ms. Anm: Die Methode überprüft nicht, ob auch wirklich ein Leser angeschlossen ist. Dazu verwendet man die Methode FindBaudrate.
ConnectUSB	Öffnet einen USB-Kanal für einen USB-Leser. Die Device-ID des Lesers muss zuvor mit der Funktionsbibliothek FEUSB ermittelt werden. Alternativ übergibt man eine 0 zum Öffnen einer Verbindung mit dem zuerst gefundenen USB-Leser.
ConnectTCP	Öffnet eine Socketverbindung zu einem Leser.
DisConnect	Schließt eine bestehende Verbindung. Wenn eine TCP/IP-Verbindung geschlossen wurde, entspricht der Rückgabewert dem letzten Status der Verbindung. Wenn der Zustand TIME_WAIT ermittelt wird, dann gibt diese Funktion 0 zurück um anzuzeigen, dass die Verbindung korrekt geschlossen wurde. Siehe auch: 7.3. TCP-Status
IsConnected	Zeigt an, dass über das Reader-Objekt eine Connect-Methode aufgerufen wurde und keine weitere Verbindung möglich ist. Es wird nicht geprüft, ob der Leser noch angeschlossen ist und eine Kommunikation möglich ist.
GetTcpConnectionState	Ermittelt mit einer Kernelfunktion den Status einer TCP/IP-Verbindung. Es ist nicht möglich, durch permanentes Polling eine durch Strom- oder Netzwerkausfall unterbrochene Verbindung zu erkennen. Diese Methode ist hilfreich <u>nach</u> einem Kommunikationsfehler um die Ursache eingrenzen zu können. Siehe auch: 7.3. TCP-Status
ReaderAuthentication	Zwei überladenen Methoden zur Eröffnung einer Session zur verschlüsselten Datenübertragung mit einem Reader. Diese Methoden kapseln die Funktion FEISC_0xAE_ReaderAuthent. Weitergehende Informationen finden sich im Handbuch H9391-xx-ID-B zur Bibliothek FEISC unter dem Kapitel „Sicherheit in der Datenübertragung“.
ReadReaderInfo	Methode liest mit mehreren [0x66] Reader Info Protokollen alle wesentlichen Informationen vom Leser aus und speichert die Daten in der Struktur FEDM_ISC_READER_INFO. Zusätzlich wird intern der ermittelte Lesertyp gespeichert, den die Leserklasse für ein fehlerfreies Funktionieren dringend benötigt.

Methode	Beschreibung
ReadReaderDiagnostic	Methode liest mit mehreren [0x6E] Reader Diagnostic Protokollen alle wesentlichen Zustandsdaten vom Leser aus und speichert die Daten in der Struktur FEDM_ISC_READER_DIAGNOSTIC.
ReadCompleteConfiguration	Liest die gesamte Leser-Konfiguration aus und speichert diese im entsprechenden Datencontainer. EEPROM (bEEPROM=true) oder RAM (bEEPROM=false) werden adressiert.
WriteCompleteConfiguration	Schreibt die gesamte Leser-Konfiguration aus dem entsprechenden Datencontainer in den Leser. Diese Methode kann nur erfolgreich sein, wenn zuvor die gesamte Leser-Konfiguration gelesen wurde. EEPROM und RAM werden adressiert, wenn bEEPROM=true oder nur das RAM mit bEEPROM=false.
ResetCompleteConfiguration	Stellt im Leser die gesamte Konfiguration im EEPROM und RAM (bEEPROM=true) oder nur im RAM (bEEPROM=false) zurück auf Werkseinstellung.
ApplyConfiguration	Schreibt nur die modifizierten Konfigurationsblöcke in EEPROM und RAM (bEEPROM=true) oder nur RAM (bEEPROM=false) des Lesers.
TransferReaderCfgToXmlFile	Liest die gesamte Leser-Konfiguration aus, speichert diese im entsprechenden Datencontainer und schreibt anschließend die Daten in eine XML-Datei.
TransferXmlFileToReaderCfg	Öffnet eine XML-Datei, speichert die Konfigurationsdaten im entsprechenden Datencontainer und schreibt anschließend die gesamte Leser-Konfiguration in den Leser.
StartAsyncTask	Startet einen asynchronen Task für Notifications. Der Leser muss im Notification Mode oder im Host Mode mit Unterstützung für Notifications sein.
CancelAsyncTask	Beendet den asynchronen Task. Notification-Tasks müssen immer mit dieser Funktion beendet werden. Zur Vermeidung von Deadlocks wird der Task nicht beendet, wenn der Ausführungspfad des Tasks innerhalb der Callback-Funktion liegt. In diesem Fall kehrt die Funktion sofort mit dem Rückgabewert FEISC_ERR_THREAD_CANCEL_ERROR (-4084) zurück und die Applikation muß CancelAsyncTask solange aufrufen, bis der Rückgabewert nicht mehr -4084 ist. Applikations-seitig muss sichergestellt werden, dass die Callback-Funktion immer zuverlässig zurückkehrt.
TriggerAsyncTask	Triggert den mit der TaskID=FEDM_TASKID_EVERY_NEW_TAG gestarteten asynchronen Task, der nach dem Aufruf der Callback-Funktion auf diesen Trigger wartet.
SetPortHnd	Speichert den Port-Handle und stellt die Kommunikationsmode ein. Diese Methode muss verwendet werden, falls ein Kommunikationsport nicht mit einer der Connect-Methoden geöffnet wurde, sondern mit der in FECOM, FEUSB oder FETCP enthaltenen Open-Funktion.
SetPortPara	Konfiguriert den geöffneten Kommunikationskanal. Die Konfigurationsparameter sind der jeweiligen Dokumentation zu FECOM, FEUSB oder FETCP zu entnehmen.
GetPortPara	Gibt eine Einstellung des geöffneten Kommunikationskanals zurück. Die Konfigurationsparameter sind der jeweiligen Dokumentation zu FECOM, FEUSB oder FETCP zu entnehmen.
cbsTaskRsp1	Statische Callback-Funktion für Asynchronen Task. Nur zur internen Verwendung.
cbsTaskRsp2	Statische Callback-Funktion für Asynchronen Task. Nur zur internen Verwendung.

Methode	Beschreibung
Basierend auf TagHandler-Klassen optionale Methoden für Transponder-Kommunikation	
TagInventory	Methode führt einen Inventory auf der RF-Schnittstelle aus, sammelt alle ermittelten Transponder in der internen Tabelle und gibt eine Liste mit TagHandler-Klassen zurück.
TagSelect	Ein Select-Kommando wird abgesetzt und (für manche ISO 14443 Transponder) wird die TagHandler-Klasse angepasst. Optional kann für ISO 14443 konforme Reader ein Transpondertreiber angegeben werden.
GetTagHandler	Die TagHandler-Klasse eines Transponders wird anhand der Seriennummer identifiziert und zurückgegeben
GetSelectedTagHandler	Die TagHandler-Klasse des selektierten Transponders zurückgegeben
GetTagList	Gibt die Liste aller zuvor mit TagInventory erzeugten TagHandler zurück.
GetNonAddressedTagHandler	<p>Transponder, die den non-addressed Mode unterstützen, können ohne Inventory angesprochen werden. Um spezielle TagHandler benutzen zu können und wenn eine Applikation nur eine Sorte Transponder benutzt, für die es eine TagHandler-Klasse mit Chip-spezifischen Methoden gibt, muss man diesen TagHandler mit dieser Methode manuell erzeugen.</p> <p>Alle TagHandler-Typen sind in der Klasse FedmlscTagHandler angegeben, wobei nicht jeder für non-addressed Kommunikation spezifiziert ist. Nähere Informationen dazu kann nur das Datenblatt des Transponder-Herstellers geben.</p> <p>Wichtige Anmerkungen:</p> <ol style="list-style-type: none"> 1. Es kann immer nur ein TagHandler für non-addressed Kommunikation erzeugt werden. 2. Jeder Aufruf von CreateNonAddressedTagHandler zerstört den aktuellen NonAddressed-TagHandler! 3. Ein gemischter Betrieb von addressed und non-addressed Kommunikation ist nicht möglich.
Convert_EPC_C1_G2_TagHandler	Die Class1 Gen2 Spezifikation definiert keine Identifikationsmerkmale für Hersteller und Chiptypen, wie dies z.B. für ISO 15693 der Fall ist. Um dennoch spezielle TagHandler benutzen zu können und wenn eine Applikation nur eine Sorte UHF-Transponder benutzt, für die es eine TagHandler-Klasse mit Chip-spezifischen Methoden gibt, muss man diesen TagHandler nach einem TagInventory manuell erzeugen.

6.3. TagHandler-Konzept

Mit dem Konzept der TagHandler steht eine Sammlung von Transpondertyp-spezifischen Klassen zur Verfügung, die dem Programmierer eine effiziente Programmierung der Transponder-Kommunikation ermöglicht. TagHandler stehen **nur im Host-Modus** zur Verfügung und der Support muss zuvor mit der Methode EnableTagHandler der Klasse FEDM_ISCReader aktiviert werden.

Das Konzept basiert auf der automatischen Identifikation des Transpondertyps nach einem Inventory. Bei ISO 15693 konformen Transpondern werden die in der Seriennummer enthaltenen Hersteller-ID und Chip-ID ausgewertet. Bei ISO 14443 Transpondern wird nach dem obligatorischen Select die Card-Info ausgewertet oder, im Fall der Auswahl eines Transpondertreibers beim Select, kann der Typ des TagHandlers sofort ermittelt werden.

Alle TagHandler-Klassen sind von der Basisklasse FedmIscTagHandler abgeleitet. Darüber hinaus sind Beziehungen der einzelnen Transpondertypen untereinander durch Ableitung nachgebildet worden.

TagHandler-Klassen werden intern erzeugt, verwaltet und gelöscht. Nach jedem Aufruf von TagInventory (oder SendProtocol(0xB0) für Cmd=0x01) wird für jeden gespeicherten TagHandler geprüft, ob der zugehörige Transponder noch erkannt wurde und gegebenenfalls gelöscht. Deshalb hat ein TagHandler eine Lebensdauer von normalerweise nur einem Inventory-Zyklus.

Kleines Beispiel:

```
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. Mit den Leserklassen arbeiten

Applikationen, die im Host-Mode mit Transpondern kommunizieren, haben die Wahl zwischen der Tabellen-orientierten Programmierung oder der Verwendung der TagHandler-Klassen. Es wird empfohlen zuerst den Einsatz der TagHandler zu prüfen. Diese Klassen setzen auf der Tabellen-orientierten Transponderverwaltung auf und bieten das effizientere API. Nur für den Fall fehlender TagHandler-Klassen oder der Kommunikation im non-addressed Mode muss auf die (alte) Methode zurückgegriffen werden. Kapitel [6.5. Tabelle für ISO Host Commands](#) bietet dazu den geeigneten Einstieg.

6.4.1. Notwendige Initialisierungen

Vor der ersten Verwendung der Protokollmethode müssen einige Initialisierungen durchgeführt werden, wenn andere als voreingestellte Einstellungen gewünscht werden:

1. Busadresse Die Busadresse des Lesers ist in der Klasse mit 255 voreingestellt. Eine andere Adresse stellt man mit der Methode SetBusAddress ein.

Anm: Die Busadresse ist nur für die serielle Kommunikation von Bedeutung.
2. ReaderHandle **FEDM_ISCReader:** Der Handle eines Leserobjektes in der FEISC-Funktionsbibliothek muss immer mit der Methode SetReaderHnd (s. FEDM_DataBase) in einer Instanz der Leserklasse hinterlegt werden.

FEDM_ISCReaderModule: Bei Verwendung dieser Klasse wird ein Leserobjekt in der FEISC-Funktionsbibliothek automatisch im Konstruktor angelegt.
3. PortHandle **FEDM_ISCReader:** Der Handle einer Schnittstelle, die mit FECOM, FETCP oder FEUSB geöffnet wurde, muss im Leserobjekt der FEISC-Funktionsbibliothek hinterlegt werden. Dies kann man entweder bei der Erzeugung des Leserobjektes mit FEISC_NewReader tun oder nachträglich mit der Methode SetPortHnd. Die nachträgliche Änderung ist auch immer dann möglich, wenn zur Laufzeit die Schnittstelle gewechselt werden soll.

FEDM_ISCReaderModule: Kommunikations-Schnittstellen werden mit den Connect-Methoden geöffnet. Der Port-Handle wird intern gesetzt.
4. Lesertyp Der Lesertyp muss in der Klasse gesetzt werden. Dies kann auf zwei Arten erfolgen:
 1. Nach dem erfolgreichen Verbindungsaufbau mit dem Aufruf der Methode ReadReaderInfo der Klasse **FEDM_ISCReaderModule** (empfohlen).

2. Mit der Methode `SetReaderType`. Die Konstanten für die Lesertypen sind in den Datei `FEDM_ISC.h` enthalten.
5. Sprachunterstützung (optional) Fehlertexte sind in mehreren Sprachen abrufbar. Die Sprache kann man mit der Methode `SetLanguage` einstellen. Voreingestellt ist Englisch.
6. Tabellengröße Die integrierten Tabellen für den Buffered Read Mode (BRM) und den ISO Host Mode sind nicht initialisiert. Vor der ersten Kommunikation muss man die Größe der Tabellen mit der Methode `SetTableSize` einstellen. Die Größe wählt man gleich der maximalen Anzahl gleichzeitig im Antennfeld befindlicher Transponder.
- Man braucht nur die Größe der Tabelle setzen, die man auch verwendet.
7. TagHandler-Support (optional) Der Support für TagHandler-Klassen muss mit der Methode `EnableTagHandler` manuell aktiviert werden, sofern TagHandler-Klassen benutzt werden sollen.

6.4.2. Verwaltung der Leserkonfiguration

Jeder OBID i-scan® und OBID® *classic-pro* Leser wird durch Parameter gesteuert, die Blockweise gruppiert in einem EEPROM gespeichert und im Systemhandbuch zum jeweiligen Leser ausführlich beschreiben sind. Nach dem Einschalten bzw. Reset des Lesers werden alle Parameter in das RAM geladen, ausgewertet und in die Steuerung einbezogen.

Alle Parameter können mit einem Protokoll verändert werden, damit das Verhalten des Lesers an die Applikation angepasst werden kann. Idealerweise verwendet man das Programm ISOStart für diese Anpassung und normalerweise müssen in der Applikation keine Parameter mehr geändert werden. Trotzdem kann es vorkommen, dass aus einem Programm heraus ein oder mehrere Parameter umgestellt werden müssen. Dieses Kapitel soll Sie mit der Vorgehensweise unter Verwendung der Leserklasse an einem Beispiel vertraut machen.

Eine gemeinsame Eigenschaft aller Leser ist die Blockweise Gruppierung von thematisch verwandten Parametern zu je 14 Byte je Konfigurationsblock. Jeder Parameter kann nicht einzeln adressiert werden, sondern muss immer zusammen mit einem Konfigurationblock mit dem Protokoll [0x80] Read Configuration ausgelesen, dann modifiziert und schließlich wieder mit dem Protokoll [0x81] Write Configuration in den Leser geschrieben werden. Dieser Zyklus ist immer einzuhalten und wird von der Leserklasse FEDM_ISCReader auch kontrolliert. Das bedeutet, dass das Schreiben eines Konfigurationsblockes ohne vorheriges Lesen desselben Blocks nicht möglich ist, mit der Ausnahme, dass nach der Übernahme von Konfigurationsparametern aus einer XML-Datei das Schreiben in den Leser immer möglich ist.

Die Leserklasse verwaltet die Konfigurationsdaten in einem (public) Byte-Array m_RFC_EEData¹ für Daten aus dem EEPROM bzw. m_RFC_RAMData für Daten aus dem RAM des Lesers. Die Unterscheidung ist von Bedeutung, da Änderungen im RAM sofort zur Anwendung kommen, während Änderungen im EEPROM des Lesers erst nach einem Reset wirksam werden. Deshalb hat die Leserklasse für beide Konfigurationssätze eigene Byte-Arrays.

Am Beispiel des Konfigurationsblockes CFG2 des Lesers ID ISC.LR2000, der Parameter zur Einstellung der digitalen Ein- und Ausgänge enthält, soll erläutert werden, wie man gezielt einen Parameter mit Hilfe der Leserklasse modifiziert.

¹ RFC ist eine Abkürzung für RF-Controller

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

Dargestellt ist die Belegung des Konfigurationsblockes CFG2. Der Parameter IDLE-MODE belegt zwei Bytes und enthält Sub-Parameter für vier Relais und einen digitalen Ausgang. Jeder Ausgang kann für einen von vier Zuständen entsprechend der Tabelle konfiguriert werden. Da das Feld IDLE-MODE nicht grau unterlegt ist, kann die Modifikation im RAM des Lesers erfolgen.

Für die Änderung von REL1 Mode innerhalb von IDLE-MODE sind nun folgende Schritte notwendig:

```
// das Beispiel zeigt das Lesen, Modifizieren und Zurückschreiben eines Blocks der Leserkonfiguration
```

```
// Modifiziert wird der Idle-Mode des Relais 1
```

```
// m_Reader ist ein Objekt der Leserklasse FEDM_ISCReader oder FEDM_ISCReaderModule
```

```
unsigned char ucCfgAdr = 2;           // Adresse des Konfigurationsblocks
```

```
bool bEEProm = false;                // Konfigurationsdaten aus/in RAM des Lesers
```

```
unsigned int uIdleModeRel1           // Parameter IDLE-MODE für Relais 1
```

```
// Voreinstellungen für das nächste SendProtocol
```

```
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG, (UCHAR)0x00); // alles zurücksetzen
```

```
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG_ADR, ucCfgAdr); // Adresse setzen
```

```
m_Reader.SetData(FEDM_ISC_TMP_READ_CFG_LOC, bEEProm); // Speicherort auf RAM setzen
```

```
// Konfigurationsdaten lesen
```

```
m_Reader.SendProtocol(0x80);
```

```
uIdleModeRel1 = 3; // REL1 soll alternierend anziehen (Anm: Frequenz wäre im Parameter IDLE-FLASH einzustellen)
```

```
// neuen Wert in Leserklasse speichern (Ziel: RAM des Lesers)
```

```
m_Reader.SetConfigPara(ReaderConfig::DigitalIO::Relay::No1::IdleMode, uIdleModeRel1, false);
```

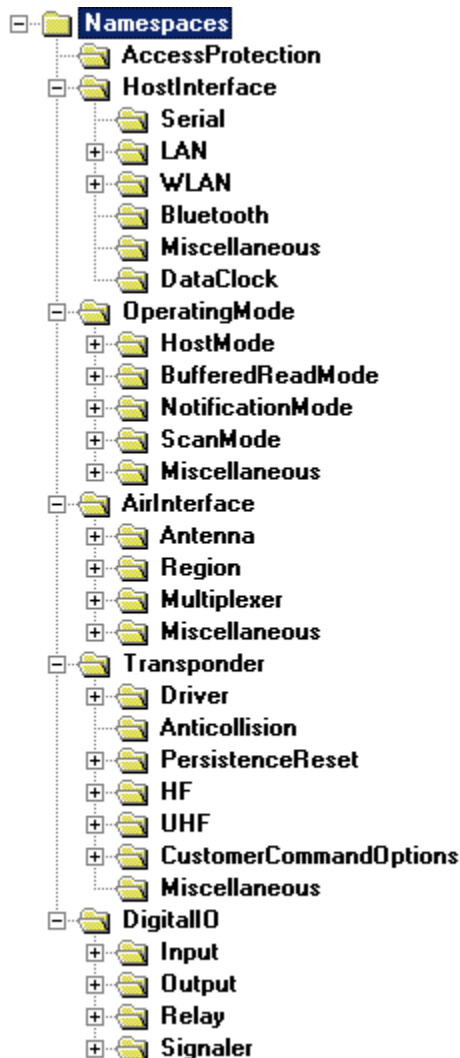
// Voreinstellungen für das nächste SendProtocol

```
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG, (UCHAR)0x00); // alles zurücksetzen  
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG_ADR, ucCfgAdr); // Adresse setzen  
m_Reader.SetData(FEDM_ISC_TMP_WRITE_CFG_LOC, bEEProm); // Speicherort auf EEPROM setzen
```

// Konfigurationsdaten zurückschreiben

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

Die Methoden GetConfigPara und SetConfigPara bekommen im ersten Übergabeparameter einen Parameternamen als String aus dem Namespace ReaderConfig übergeben. Dieser Namespace kapselt in weiteren internen Namespaces einheitlich alle Parameternamen aller OBID i-scan® und OBID® classic-pro Leser. Nachfolgend sind die Haupt- und die ersten Untergruppen dargestellt.



Dieses Schema hat den Vorteil, dass mit den heute verfügbaren Intellisense-Funktionen der IDEs die Auswahl des gewünschten Namens für einen Konfigurationsparameter schnell gefunden werden kann.

6.4.3. Beispiele für die Verwendung der Methode SendProtocol

Die Methode SendProtocol ist von zentraler Bedeutung für den Protokolltransfer. Aus diesem Grund wird für jedes Steuerbyte¹ ein Beispiel aufgeführt, das verdeutlichen soll, welche Daten mit welchen Zugriffskonstanten vor jedem Protokolltransfer in Datencontainer zu speichern sind und welche Daten nach dem Protokolltransfer zur Verfügung stehen. Mit manchen Protokollen können unterschiedliche Daten transferiert werden. In einem solchen Fall wird nur ein exemplarisches Beispiel aufgeführt.

Alle Zugriffskonstanten sind in der Datei FEDM_ISCReaderID.h aufgelistet und sollten eingehend zusammen mit der im Systemhandbuch zum Leser angeführten Erklärung der Protokolldaten studiert werden.

Auf die Auswertung der Rückgabewerte der Methode wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll	Beispiel
[0x18] Destroy	<pre> UCHAR ucMode = 0; // Modus UCHAR ucPW[3]; // Passwort UCHAR ucEPC[32]; // Puffer für EPC int iEpcLen = 0; // Anzahl Bytes in ucEPC // hole die Daten z.B. aus Eingabefeldern // ermittle die Anzahl Bytes in ucEPC SetData(FEDM_ISC_TMP_EPC_DESTROY_MODE, ucMode); 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; // Anzahl angeforderter Datensätze UCHAR ucTrData = 0; // Datensatzstruktur UCHAR ucRecSets = 0; // Anzahl Datensätze in Protokoll SetData(FEDM_ISCLR_TMP_BRM_SETS, ucDataSets); SendProtocol(0x21); // lese Daten von Transponder mit Buffered Read Mode GetData(FEDM_ISCLR_TMP_BRM_TRDATA, &ucTrData); GetData(FEDM_ISCLR_TMP_BRM_RECSETS, &ucRecSets); // Alle weiteren Transponderdaten sind in m_BRMTable enthalten. Beispiel für Datenzugriffe in // 6.6.1. Beispiele für die Verwendung der Tabelle </pre>

¹ nicht alle Steuerbytes werden von jedem Leser unterstützt. Weitere Informationen zu den unterstützten Steuerbytes entnimmt man dem Systemhandbuch des jeweiligen Lesers

[Steuerbyte] Protokoll	Beispiel
[0x22] Read Buffer	<pre> UINT uiDataSets = 1; // Anzahl angeforderter Datensätze UCHAR ucTrData = 0; // Datensatzstruktur UINT uiRecSets = 0; // Anzahl Datensätze in Protokoll SetData(FEDM_ISC_TMP_ADV_BRM_SETS, uiDataSets); SendProtocol(0x22); // lese Daten von Transponder mit Buffered Read Mode GetData(FEDM_ISC_TMP_ADV_BRM_TRDATA1, &ucTrData); GetData(FEDM_ISC_TMP_ADV_BRM_RECSETS, &uiRecSets); // Alle weiteren Transponderdaten sind in m_BRMTable enthalten. Beispiel für Datenzugriffe in // 6.6.1. Beispiele für die Verwendung der Tabelle </pre>
[0x31] Read Data Buffer Info	<pre> UINT uiTabSize = 0; // Größe des Datenpuffers UINT uiTabStart = 0; // Startadresse für ersten Datensatz UINT uiTabLen = 0; // Anzahl Datensätze in Datenpuffer 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; // RF-Controller (1 für AC-Controller) SetData(FEDM_ISC_TMP_SYSTEM_RESET_MODE, ucMode); SendProtocol(0x64); </pre>
[0x65] Get Software Version	<pre> string sSoftVer; // Software-Version als STL-String SendProtocol(0x65); GetData(FEDM_ISC_TMP_SOFTVER, sSoftVer); </pre>
[0x66] Get Reader Info	<pre> string sSoftVer; // Reader Info als STL-String SetData(FEDM_ISC_TMP_READER_INFO_MODE, (UINT)0); // identisch mit [0x65] //SetData(FEDM_ISC_TMP_READER_INFO_MODE, (UINT)1); // LRU1000: AC-Controller SendProtocol(0x66); GetData(FEDM_ISC_TMP_SOFTVER, sSoftVer); // identisch mit [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; // RF ON SetData(FEDM_ISC_TMP_RF_ONOFF, ucRF); SendProtocol(0x6A); </pre>

[Steuerbyte] Protokoll	Beispiel
[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; // minimaler Noise Level UINT uiNLAvg = 1000; // mittlerer Noise Level UINT uiNLMax = 1500; // maximaler 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; // minimaler Noise Level UINT uiNLAvg = 0; // mittlerer Noise Level UINT uiNLMax = 0; // maximaler 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; // Diagnose-Modus SetData (FEDM_ISC_TMP_DIAG_MODE, ucDiagMode); SendProtocol (0x6E);
[0x6F] Base Antenna Tuning	SendProtocol (0x6F); // der Long-Range-Reader wechselt in einen Spezialmodus // nur mit einem Reset kann dieser Modus verlassen werden
[0x71] Set Output	// Beispiel 1 aus Systemhandbuch ID ISC.M01 SetData (FEDM_ISCM_TMP_OUT_OS, (UINT)0); // OS-Bytes zurücksetzen SetData (FEDM_ISCM_TMP_OUT_OS_OUT1, (UCHAR)0x01); // Ausgang 1 aktivieren SetData (FEDM_ISCM_TMP_OUT_OS_LED_G, (UCHAR)0x10); // LED grün aus SetData (FEDM_ISCM_TMP_OUT_OS_LED_R, (UCHAR)0x01); // LED rot an SetData (FEDM_ISCM_TMP_OUT_OS_BEEPER, (UCHAR)0x11); // Piepser alternierend an SetData (FEDM_ISCM_TMP_OUT_OSF, (UINT)0); // OSF-Bytes zurücksetzen SetData (FEDM_ISCM_TMP_OUT_OSF_BEEPER, (UCHAR)0x01); // Piepser mit 4Hz SetData (FEDM_ISCM_TMP_OUT_OSTIME, (UINT)5); // 500ms Aktivzeit Piepser und LEDs SetData (FEDM_ISCM_TMP_OUT_OUTTIME, (UINT)3); // Ausgang 1 300ms aktiv SendProtocol (0x71);

[Steuerbyte] Protokoll	Beispiel
[0x72] Set Output	<pre>// Beispiel aus dem Systemhandbuch 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>// Beispiel für ID ISC.LR200 bool bIn1 = false; // Eingang 1 bool bIn2 = false; // Eingang 2 bool bDip1 = false; // Dip-Schalter 1 bool bDip2 = false; // Dip-Schalter 2 bool bDip3 = false; // Dip-Schalter 3 bool bDip4 = false; // Dip-Schalter 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; // Antennen-Spannung SendProtocol(0x75); GetData(FEDM_ISCM_TMP_ANTENNA_VALUE, &uiAntValue);</pre>

[Steuerbyte] Protokoll	Beispiel
[0x80] Read Configuration und [0x81] Write Configuration	<p>// das Beispiel zeigt das Lesen und Zurückschreiben eines Blocks der Leserkonfiguration</p> <pre> UCHAR ucCfgAdr = 1; // Adresse des Konfigurationsblocks bool bEEProm = true; // Konfigurationsdaten aus/in EEPROM des Lesers UCHAR ucBusAddress; // Busadresse des ISC.LR2000-Lesers aus Block 1 SetData(FEDM_ISC_TMP_READ_CFG, (UCHAR)0x00); // alles zurücksetzen SetData(FEDM_ISC_TMP_READ_CFG_ADR, ucCfgAdr); // Adresse setzen SetData(FEDM_ISC_TMP_READ_CFG_LOC, bEEProm); // Speicherort auf EEPROM setzen SendProtocol(0x80); // Konfigurationsdaten lesen GetConfigPara(ReaderConfig::HostInterface::Serial::BusAddress, &ucBusAdr); // z.B. Busadresse übernehmen SetData(FEDM_ISC_TMP_WRITE_CFG, (UCHAR)0x00); // alles zurücksetzen SetData(FEDM_ISC_TMP_WRITE_CFG_ADR, ucCfgAdr); // Adresse setzen SetData(FEDM_ISC_TMP_WRITE_CFG_LOC, bEEProm); // Speicherort auf EEPROM setzen SendProtocol(0x81); // Konfigurationsdaten zurückschreiben </pre>
[0x82] Save Configuration	<pre> SetData(FEDM_ISC_TMP_SAVE_CFG, (UCHAR)0x00); // alles zurücksetzen SetData(FEDM_ISC_TMP_SAVE_CFG_ADR, (UCHAR)0x01); // Adresse setzen SetData(FEDM_ISC_TMP_SAVE_CFG_MODE, TRUE); // alle Blöcke sichern SendProtocol(0x82); // Konfigurationsdaten von RAM in EEPROM sichern </pre>
[0x83] Set Default Configuration	<pre> SetData(FEDM_ISC_TMP_RESET_CFG, (UCHAR)0x00); // alles zurücksetzen SetData(FEDM_ISC_TMP_RESET_CFG_ADR, (UCHAR)0x01); // Adresse setzen SetData(FEDM_ISC_TMP_RESET_CFG_LOC, FALSE); // RAM wählen SetData(FEDM_ISC_TMP_RESET_CFG_MODE, FALSE); // nur Block 2 auf Default SendProtocol(0x83); // Konfigurationsdaten von Block 2 in RAM auf Default setzen </pre>
[0x85] Set System Timer	<pre> SetData(FEDM_ISCLR_TMP_TIME_H, (UINT)16); // 16 Uhr SetData(FEDM_ISCLR_TMP_TIME_M, (UINT)20); // 20 Minuten SetData(FEDM_ISCLR_TMP_TIME_MS, (UINT)2000); // 2000 Millisekunden SendProtocol(0x85); // Timer setzen </pre>
[0x86] Get System Timer	<pre> UINT uiHour = 0; // Stunden UINT uiMinute = 0; // Minuten UINT uiMilliSec = 0; // Millisekunden SendProtocol(0x86); // Timer auslesen GetData(FEDM_ISCLR_TMP_TIME_H, &uiHour); // Stunden übernehmen GetData(FEDM_ISCLR_TMP_TIME_M, &uiMinute); // Minuten übernehmen GetData(FEDM_ISCLR_TMP_TIME_MS, &uiMilliSec); // Millisekunden übernehmen </pre>
[0x87] Set System Date	<pre> SetData(FEDM_ISC_TMP_DATE_CENTURY, (UINT)20); // 20. Jahrhundert SetData(FEDM_ISC_TMP_DATE_YEAR, (UINT)4); // Jahr 04 im Jahrhundert 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); // z.Zt. ungenutzt SetData(FEDM_ISC_TMP_DATE_HOUR, (UINT)12); // Stunden SetData(FEDM_ISC_TMP_DATE_MINUTE, (UINT)00); // Minuten SetData(FEDM_ISC_TMP_DATE_MILLISECOND, (UINT)0); // Millisekunden (inkl. Sekunden) SendProtocol(0x87); // Datum und Uhrzeit setzen </pre>

[Steuerbyte] Protokoll	Beispiel
[0x88] Get System Date	<pre> UCHAR ucCentury = 0; // Jahrhundert UCHAR ucYear = 0; // Jahr im Jahrhundert UCHAR ucMonth = 0; // Monat UCHAR ucDay = 0; // Tag UCHAR ucTimezone = 0; // Zeitzone (z.Zt. ungenutzt) UCHAR ucHour = 0; // Stunde UCHAR ucMinute = 0; // Minute UINT uiMilliSec = 0; // Millisekunden SendProtocol(0x88); // Datum und Uhrzeit auslesen GetData(FEDM_ISC_TMP_DATE_CENTURY, &ucCentury); // Jahrhundert GetData(FEDM_ISC_TMP_DATE_YEAR, &ucYear); // Jahr im Jahrhundert GetData(FEDM_ISC_TMP_DATE_MONTH, &ucMonth); // Monat GetData(FEDM_ISC_TMP_DATE_DAY, &ucDay); // Tag im Monat GetData(FEDM_ISC_TMP_DATE_TIMEZONE, &ucTimezone); // z.Zt. ungenutzt GetData(FEDM_ISC_TMP_DATE_HOUR, &ucHour); // Stunden GetData(FEDM_ISC_TMP_DATE_MINUTE, &ucMinute); // Minuten GetData(FEDM_ISC_TMP_DATE_MILLISECOND, &uiMilliSec); // Millisekunden </pre>
[0x8A] Read Configuration und [0x8B] Write Configuration	<pre> // das Beispiel zeigt das Lesen und Zurückschreiben eines Blocks der Leserkonfiguration UINT uiCfgAdr = 1; // Adresse des Konfigurationsblocks UCHAR ucBusAddress; // Busadresse des ISC.LRU3000-Lesers aus 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); // alles zurücksetzen SetData(FEDM_ISC_TMP_0x8A_READ_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8A_READ_REQ_CFG_ADR, uiCfgAdr); // Konfigurations-Adresse SetData(FEDM_ISC_TMP_0x8A_READ_REQ_CFG_N, (UCHAR)1); // 1 Konfigurationsblock SendProtocol(0x8A); // Konfigurationsdaten lesen // z.B. Busadresse übernehmen GetConfigPara(ReaderConfig::HostInterface::Serial::BusAddress, &ucBusAdr); // Parameteränderung mit 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); // alles zurücksetzen SetData(FEDM_ISC_TMP_0x8B_WRITE_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8B_WRITE_CFG_ADR, uiCfgAdr); // Konfigurations-Adresse SetData(FEDM_ISC_TMP_0x8B_WRITE_CFG_N, (UCHAR)1); // 1 Konfigurationsblock SendProtocol(0x8B); // Konfigurationsdaten zurückschreiben </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); // alles zurücksetzen SetData(FEDM_ISC_TMP_0x8C_RESET_MODE_LOC, true); // EEPROM SetData(FEDM_ISC_TMP_0x8C_RESET_CFG_ADR, (UINT)1); // Konfigurations-Adresse SetData(FEDM_ISC_TMP_0x8C_RESET_CFG_N, (UCHAR)1); // 1 Konfigurationsblock SendProtocol(0x8C); // Konfigurationsdaten in CFG1 zurücksetzen </pre>
[0xA0] Reader Login	<pre> UCHAR ucPW[] = {0x00, 0x00, 0x00, 0x00}; // Passwort SetData(FEDM_ISCLR_TMP_READER_PW, ucPW, 4); // Passwort setzen SendProtocol(0xA0); // Passwort an Leser schicken </pre>
[0xA2] Write Mifare Keys	<pre> UCHAR ucKey[6]; </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre>// hole den Mifare-Key z.B. aus einem Eingabefeld 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(0xA2); // Mifare-Key an Leser schicken</pre>
[0xA3] Write DES/AES Keys	<pre>UCHAR ucKey[16]; // hole den Key z.B. aus einem Eingabefeld 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); // Key an Leser schicken</pre>
[0xB0] ISO Mandatory and Optional Commands	<pre>// das Beispiel zeigt den [0x01] Inventory // Alternative: Verwendung der Methode TagInventory und der TagHandler-Klassen SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x01); // Inventory SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // kein More-Flag SendProtocol(0xB0); // die Inventory-Daten liegen in m_ISOTable. Beispiel für Datenzugriffe in 6.5. Tabelle für ISO Host Commands</pre>
[0xB1] ISO15693 Customer and Proprietary Commands (TagHandler-Klassen bieten ein vollständiges und einfacheres API)	<pre>// das Beispiel zeigt den [0xA2] Set EAS // alle anderen 0xB1-Kommandos entsprechend string sSnr; // STL-String für Seriennummer UCHAR uclISOError = 0; // für ISO-Fehlercode SetData(FEDM_ISC_TMP_B1_CMD, (UCHAR)0xA2); // Set EAS SetData(FEDM_ISC_TMP_B1_MFR, (UCHAR) FEDM_ISC_ISO_MFR_PHILIPS); // Hersteller SetData(FEDM_ISC_TMP_B1_MODE, (UCHAR) FEDM_ISC_ISO_MODE_ADR); // addressed // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern SetData(FEDM_ISC_TMP_B1_REQ_UID, sSnr); int iStatus = SendProtocol(0xB1); if(iStatus == 0x95) { // hole ISO-Fehlercode GetData(FEDM_ISC_TMP_B1_ISO_ERROR, &uclISOError) }</pre>

[Steuerbyte] Protokoll	Beispiel
<p>[0xB2] ISO14443 Special Commands</p> <p>[0x2B] ISO14443-4 Transponder Info</p> <p>(TagHandler-Klassen bieten ein vollständiges und einfacheres 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) { // Tabellenindex des selektierten Transponders ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_IS_SELECTED, true); if(idx >= 0) { // hole Transponderdaten GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_FSCI, &ucFSCI) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_FWI, &ucFWI) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_DSI, &ucDSI) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_DRI, &ucDRI) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_NAD, &ucNad) GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_CID, &ucCid) } } </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB0] Authent Mifare</p> <p>(TagHandler-Klassen bieten ein vollständiges und einfacheres API)</p>	<pre> UCHAR ucDBAdr = 0; // Adresse des ersten Datenblocks UCHAR ucKeyType = 0; // Keytype für Authentifikation UCHAR ucKeyAdr = 0; // EEPROM-Adresse des Keys im Leser UCHAR ucKeyLocation = 0; // Location of the Authent-Key (0: Reader; 1: Protocol) string sKey = "000000000000"; // Authent-Key SetData(FEDM_ISC_TMP_B2_CMD, (UCHAR)0xB0); // Authent Mifare SetData(FEDM_ISC_TMP_B2_MODE, (UCHAR)0x00); // Byte löschen SetData(FEDM_ISC_TMP_B2_MODE_ADR, (UCHAR) FEDM_ISC_ISO_MODE_SEL); // selected SetData(FEDM_ISC_TMP_B2_MODE_KL, ucKeyLocation); SetData(FEDM_ISC_TMP_B2_REQ_KEY_TYPE, ucKeyType); SetData(FEDM_ISC_TMP_B2_REQ_DB_ADR, ucDBAdr); if(keyLocation == 0) reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR, ucKeyAdr); else reader.SetData(FEDM_ISC_TMP_ISO14443A_KEY, sKey); SendProtocol(0xB2); </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB1] Authent my-d</p> <p>(TagHandler-Klassen bieten ein vollständiges und einfacheres API)</p>	<pre> UCHAR ucKeyAdrTag = 5; // Adresse des Keys auf dem Transponder UCHAR ucKeyAdrSam = 2; // Adresse des Keys im Authentifikationsmodul UCHAR ucCntAdr = 3; // Adresse des Authentifikationszählers UCHAR ucAuthSeq = 0; // Authentifikationssequenz 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>

[Steuerbyte] Protokoll	Beispiel
<p>[0xB2] ISO14443 Special Commands</p> <p>[0x30] Mifare Value Commands</p> <p>(TagHandler-Klassen bieten ein vollständiges und einfacheres API)</p>	<pre> UCHAR ucMFCmd = 0x01; // Mifare Command UCHAR DBAdr = 0x05; // Datenblock-Adresse UCHAR ucOpValue[] = {0x00, 0x00, 0x00, 0x03}; // OP_VALUE UCHAR ucDestAdr = 0x05; // Zieladresse 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. Asynchrone Tasks zur Entlastung von Applikationen

Eine immer wiederkehrende Aufgabe von Applikationen ist die Inventarisierung von Transpondern im Antennenfeld des Lesers oder der Empfang von Notifications. Idealerweise sollten diese Aktionen im Hintergrund ablaufen und die Applikation dann informieren, wenn Transponder im Feld sind bzw. die Notification eingetroffen ist. Auch Kommunikationsvorgänge mit langer Antwortzeit sollten asynchron ablaufen.

Exakt diese Funktionalität kann u.a. mit der Methode **FEDM_ISCReaderModule::StartAsyncTask** realisiert werden. Intern wird dazu ein Thread gestartet, der das komplexe Handling mit dem Leser übernimmt und die Antwortdaten per Callback-Funktion an die Applikation liefert.

Asynchrone Tasks sind für mehrere Anwendungsfälle definiert, u.a. für Inventory im Host-Mode oder für den Empfang von Buffered-Read-Mode Daten im Notification Mode.

Asynchrone Tasks kann man für mehrere Leser gleichzeitig spezifizieren, sofern sie mit unterschiedlichen Reader-Objekten gestartet werden. Problematisch sind Leser am RS485-Bus. In einem solchen Fall kann man immer nur einen Leser gleichzeitig ‚beobachten‘, weil diese an derselben Schnittstelle angeschlossen sind.

In der nachfolgenden Tabelle sind die Besonderheiten der Tasks dargelegt:

Task	TaskID	Anmerkungen
Einmaliger Inventory	FEDM_TASKID_FIRST_NEW_TAG	<p>Ein Task kann nur gestartet werden, wenn folgende Option in der Firmware des Lesers integriert ist: Das Leserprotokoll [0xB0][0x01] Inventory muß in seinem Mode-Byte ein optionales NOTIFY-Flag unterstützen.</p> <p>Nach dem Empfang des Leserprotokolls innerhalb der vorgegebenen Zeit, beendet sich der Task selbständig. Kommt es zu einer Zeitüberschreitung, wird die Callback-Funktion aufgerufen und der Status 0x01 (kein Transponder im Lesefeld) übermittelt und der Task beendet. Im Fehlerfall wird der Task immer sofort beendet und die Callback-Funktion übergibt den Fehlercode.</p> <p>Unterstützt werden die drei Schnittstellen Seriell, USB und TCP/IP, wobei die Schnittstellen vor dem Starten des Tasks geöffnet sein müssen. Der selbständige Verbindungsaufbau per TCP/IP vom Leser oder einem geeigneten Konverter zur Übermittlung der Daten ist nicht möglich.</p> <p>Callback-Funktion in FEDM_TASK_INIT: cbFct1</p> <p>Die Daten sind in der internen ISOTable eingefügt und über GetISOTableItem verfügbar.</p>

Task	TaskID	Anmerkungen
Repetierender Inventory	FEDM_TASKID EVERY_NEW_TAG	<p>Es gelten die Bedingungen des einmaligen Inventory mit folgendem Unterschied:</p> <p>Der repetierende Inventory definiert eine zyklische Aufgabe, die nur durch FEDM_ISCReaderModule::CancelAsyncTask beendet werden kann. Ein Zyklus entspricht einem einmaligen Inventory und endet in einer Warteschleife, bis der nächste Zyklus von der Applikation durch den Aufruf von FEDM_ISCReaderModule::TriggerAsyncTask erneut angestoßen wird. Durch die Applikations-seitige Triggerung wird sichergestellt, dass eine Applikation Zeit für die Entgegennahme und Bearbeitung der Inventarisierungsdaten erhält.</p> <p>Callback-Funktion in FEDM_TASK_INIT: cbFct1</p> <p>Die Daten sind in der internen ISOTable eingefügt und über GetISOTableItem verfügbar.</p>
Empfang von Notifications	FEISC_TASKID_NOTIFICATION	<p>Ein Task sollte nur gestartet werden, wenn der Notification-Mode in der Firmware des Lesers intergriert und aktiviert ist. Unterstützt wird nur die Kommunikation über TCP/IP. Mögliche Verbindungsoptionen sind (s. Systemhandbuch zum Leser):</p> <ul style="list-style-type: none"> - Temporärer Verbindungsaufbau durch den Leser für die Dauer der Datenübertragung - Dauerhafter Verbindungsaufbau durch den Leser (in Planung) - Dauerhafter Verbindungsaufbau durch den Host (in Planung) <p>Der Task definiert eine endlose Aufgabe, die nur durch FEDM_ISCReaderModule::CancelAsyncTask beendet werden kann, bzw. im Fehlerfall während der Initialisierungsphase, nach dem Aufruf der Callback-Funktion, sofort beendet wird.</p> <p>Der Task wartet auf den Empfang der Buffered-Read-Mode Daten und ruft anschließend die Callback-Funktion auf. Nach der Rückkehr der Callback-Funktion können sofort wieder Daten vom Leser entgegengenommen werden.</p> <p>Bei Übertragungsfehlern wird die Callback-Funktion mit dem Fehlercode aufgerufen und anschließend die Empfangsprozeder fortgesetzt. Wenn die Keep-Alive Option aktiviert ist (empfohlen), dann wird eine Unterbrechung der Netzwerkverbindung erkannt, der empfangende Socket geschlossen und anschließend neu initialisiert. Dadurch ist sichergestellt, dass der RFID-Leser nach der Wiederherstellung der Verbindung erneut eine Verbindung aufbauen kann.</p> <p>Hinweis: je nach Einstellung des Lesers können in kürzesten Abständen sehr viele Daten vom Leser verschickt werden. Ohne Handshake-Mechanismen (s. Systemhandbuch zum Leser) können u.U. Daten verloren gehen, wenn der Host für die Quantität der Notifications nicht geeignet ist.</p> <p>Callback-Funktion in FEDM_TASK_INIT: cbFct1 oder cbFct2</p> <p>Die Daten sind in der internen BRMTable eingefügt und über GetBRMTableItem verfügbar.</p>

Das interne Task-Verhalten wird wesentlich durch die Struktur **FEDM_TASK_INIT** bestimmt, die mit **FEDM_ISCReaderModule::StartAsyncTask** übergeben wird. Sie enthält u.a. die für die Callback-Funktion notwendigen Parameter.

```
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;
```

Kernelement der Struktur ist die *union* mit Funktionszeigern. Die Auswahl der Callback-Funktion wird mit dem Parameter *uiFlag* vorgenommen. Der Parameter *pAny* kann für beliebige Daten verwendet werden und wird im ersten Parameter der Callback-Funktion zurückgegeben. C++ Programmierer können damit einen Zeiger des aufrufenden Objektes in die statisch deklarierte Callback-Funktion übertragen bekommen und so auf Klassenfunktionen zugreifen. *uiTimeout*

definiert die Zeitüberschreitung für einen Inventory-Zyklus bzw. die maximale Zeit zum Empfang eines Notification-Protokolls. Die Wertigkeit ist abhängig von den Vorgaben im Systemhandbuch des Lesers zum Protokoll [0xB0][0x01] Inventory in Sekunden.

Die Strukturvariablen *clPAdr* und *iPortAdr* sind ausschließlich für den Notification-Task vorgesehen.

Wichtiger Hinweis: vor der Verwendung der Struktur FEDM_TASK_INIT muss diese mit 0 initialisiert werden: z.B. mit `memset(myTaskInit, 0, sizeof(FEDM_TASK_INIT));`

6.4.5. Die Serialisierung im XML-Format¹

Mit der Standardisierung von XML (Extensible Markup Language) hat sich eine Beschreibungssprache für Dokumente durchgesetzt, die unabhängig der eingesetzten Computersprache und Betriebssysteme verwendet werden kann. Es ist folglich naheliegend, mit dieser Sprache die Struktur einer Leser-Konfigurationsdatei zu definieren. Nachfolgend ist der Inhalt einer XML-Datei dargestellt, die mit dem Programm ISOStart erstellt wurde:

```
<?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="00" 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="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="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="05" b12="04" 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"/>
  </data-array>
```

¹ nur verwendbar, wenn die Kompiler-Definition `_FEDM_XML_SUPPORT` gesetzt ist

```

<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"/>
<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>

```

Neben einigen Headerdaten finden sich in den Tags `<data-array name="Reader EEPROM-Parameter" blocks="16" size="16">` und `<data-array name="Reader RAM-Parameter" blocks="16" size="16">` die Leserparameter als Hex-Werte.

Mit der Serialize-Methode kann nun diese Datei erstellt, bzw. die Leserkonfiguration einer solchen Datei ausgelesen und in den internen Speicher EEData bzw. RAMData übernommen werden. Voraussetzung für die Generierung der Konfigurationsdatei ist, dass zuvor die Leserkonfiguration ausgelesen wurde.

Zum Erzeugen einer Leserkonfigurationsdatei verwendet man den Aufruf:

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

und zum Einlesen der Daten aus einer Leserkonfigurationsdatei den Aufruf:

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

6.5. Tabelle für ISO Host Commands

Hinweis: dieses Kapitel ist nur relevant, wenn auf TagHandler-Klassen verzichtet wird.

Mit den [0xB0] ISO-Commands können Daten mit mehreren im Antennenfeld des Lesers befindliche Transpondern ausgetauscht werden. Zur strukturierten Ablage dieser Transponderdaten ist innerhalb der Leserkasse FEDM_ISCReader die Tabelle m_ISOTable (protected) implementiert, die die Protokolldaten eines Transponders als einen Tabelleneintrag verwaltet. Wenn z.B. mit einem Protokoll Daten von drei Transpondern kommen, werden drei Tabelleneinträge mit den Transponderdaten gefüllt.

Die Tabelle hat voreingestellt eine Größe von 0. Sie muss deshalb vor der ersten Verwendung mit der Methode SetTableSize initialisiert werden. Die Größe wird bestimmt durch die maximale Anzahl an Transpondern, die sich gleichzeitig im Antennenfeld des Lesers befinden können.

Die Daten werden in die leere Tabelle immer beginnend ab dem Index 0 eingebaut. Die (protected) Variable m_iISOTableLength der Leserkasse gibt darüber Auskunft, wieviele Einträge gültig sind.

Der [0x0B] [0x01] Inventory hängt immer neue Transponderdaten an. Somit wächst die Anzahl der gültigen Tabelleneinträge kontinuierlich, bis vom Anwendungsprogramm die Methode ResetTable aufgerufen wird, oder bis die durch die Methode SetTableSize festgelegte maximale Größe der Tabelle erreicht wird. Üblicherweise löscht man deshalb die Tabelle vor jedem neuen Inventory.

Alle ISO Commands im Non-Addressed Mode nutzen immer nur den Tabellenindex 0. Wenn man ein Anwendungsprogramm nur für diesen Spezialmodus entwickelt, kann man die Tabellengröße auch auf 1 einstellen und so Speicherplatz sparen.

Der Zugriff auf die Tabellendaten erfolgt immer mit den Methoden GetTableData und SetTableData der Leserkasse FEDM_ISCReader. Als Parameter uiTableID wird immer FEDM_ISC_ISO_TABLE übergeben. Eine Auflistung und Erläuterung der Zugriffskonstante im Parameter uiDataID findet sich in [7.3.4. Konstanten für uiDataID](#).

Zum Finden bestimmter Tabelleneinträge, z. B. anhand der Seriennummer, steht die Methode FindTableIndex bereit. Weitere Abfragemethoden sind: GetTableSize und GetTableLength. Mit ResetTable löscht man die Tabelle. Optional kann man dabei auch mit dem Parameter uiDataFlags=FEDM_ISC_DATA_ALL alle Datenfelder mit 0 initialisieren. Alle genannten Methoden sind im Kapitel zur Leserkasse s. 6.1. FEDM_ISCReader beschrieben.

In der Tabelle m_ISOTable sind separate Datenpuffer für Datenblöcke aus Empfangs- (m_ucRxDB) und Datenblöcken aus Sendeprotokollen (m_ucTxDB) integriert. Das erlaubt die einfache Verifikation der geschriebenen mit den gelesenen Datenblöcken.

Nach einem [0xB0] [0x25] Select wird der betreffende Tabelleneintrag als selektiert markiert (m_bIsSelected wird auf true gesetzt). Alle nachfolgenden ISO Commands im Selected Mode suchen automatisch nach diesem Tabelleneintrag und wickeln den Datenaustausch darüber ab. Es kann immer nur ein Tabelleneintrag als selektiert markiert sein, weil auch immer nur ein Transponder im Antennenfeld des Lesers selektiert sein kann. Ein neuerlicher Select mit einer

anderen Seriennummer (UID) deselektiert deshalb automatisch den zuletzt selektierten Tabelleneintrag, bevor der neue Tabelleneintrag als selektiert markiert wird. Ein [0xB0] [0x26] Reset to Ready hebt die letzte Selektion sowohl des Transponders als auch der Tabelle auf.

Mit der Methode

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

kann der aktuell als selektiert markierte Tabellenindex ermittelt werden.

Der Speicherplatzbedarf für einen Tabelleneintrag beträgt standardmäßig 17496 Byte. Mit der Methode SetTableSize kann man allerdings einige Datenfelder eines Tabelleneintrags kleiner oder größer dimensionieren.

Wichtiger Hinweis: Die ISO-Tabelle unterstützt nicht die [0xB1] ISO15693 Custom and Proprietary Commands. Diese Commands sind mit den TagHandler-Klassen realisiert.

6.5.1. Besonderheiten des addressed mode

Fast alle Host-Commands können im addressed mode verwendet werden. In diesem Fall muss die Seriennummer – manchmal auch unified identifier (UID) genannt – in das Protokoll eingesetzt werden. In früheren Versionen der Bibliothek wurden nur UIDs mit einer Länge von 8 Byte unterstützt. Inzwischen ermöglicht ein Flag im Mode-Byte (UID_LF) abweichende UID-Längen. Ist das UID_LF Flag gesetzt, muss die Länge der UID im Protokoll eingetragen werden.

Nachfolgendes Beispiel zeigt exemplarisch einen [0xB0][0xB23] Read Multiple Blocks:

```
// setze UID für Addressed Mode (UID darf bis zu 32 Byte enthalten)
SetData(FEDM_ISC_TMP_B0_REQ_UID, sUid);
SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, ucUidLen);    // Anzahl Byte der UID

SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x23);        // Command Read Multiple Blocks
SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00);        // Mode-Byte zurücksetzen
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);    // ein Datenblock lesen
SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr);      // setze Datenblock-Adresse

SendProtocol(0xB0); // Kommunikation mit Leser/Transponder
```

6.5.2. Beispiele für die Verwendung der Tabelle mit [0xB0] Protokollen

Die nachfolgenden Beispiele demonstrieren exemplarisch den Datenaustausch mit der Tabelle FEDM_ISOTable. Auf die Auswertung der Rückgabewerte der Methoden wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll	Beispiel
[0x01] Inventory für HF-Transponder: - 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 für UHF-Transponder: - ISO18006-6-B - EM4222 - EPC Class0/0+ - EPC Class1 Gen1 - EPC Class1 Gen2	<pre> UCHAR ucTrType = 0; // für Transpondertyp string sSnr; // für Seriennummer (auch EPC) string sHeader; // für EPC Header string sDomain; // für EPC DomainManager-Feld string sObject; // für EPC ObjektClass-Feld string sEPC; // für EPC ("Header.DomainManager.ObjectClass.SerialNumber") SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x01); // Command Inventory SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // kein More-Flag // Tabelleninhalt komplett löschen (mit Option FEDM_ISC_DATA_ALL) oder // Tabellenlänge auf 0 setzen (ohne Option FEDM_ISC_DATA_ALL) ResetTable(FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_ALL); SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_ISO_TABLE); ++iCnt) { // hole Transpondertyp 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 UHF case 0x88: // EPC Class0/0+ case 0x89: // EPC Class1 Gen1 // hole Seriennummer als String GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); break; case 0x06: // EPC (Electronic Product Code) // hole EPC-Felder 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>

[Steuerbyte] Protokoll	Beispiel
	<pre> // oder hole EPC-Feld als kompletten String GetTableData(iCnt, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_EPC, sEPC); break; } } </pre>
[0x02] Stay Quiet	<pre> string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x22] Lock Multiple Blocks	<pre> /* <u>Achtung:</u> mit diesem ISO Command werden Datenblöcke unwiederruflich schreibgeschützt!! string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // einen Datenblock sperren SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, (UCHAR)0x00); // setze Datenblock-Adresse SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x23] Read Multiple Blocks (normale Adressierung)	<pre> UCHAR ucDB[32]; // Puffer für einen Datenblock (max. Blocksize 32) UCHAR ucDBAdr = 5; // Datenblock-Adresse 5 string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen // setze Seriennummer (genau 8 Byte) für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // ein Datenblock lesen SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr); // setze Datenblock-Adresse SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // hole die Größe des Datenblocks (Blocksize) 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, ucDBAdr, ucDB, 32); </pre>

[Steuerbyte] Protokoll	Beispiel
	// ... mach was mit dem Datenblock
[0x23] Read Multiple Blocks (erweiterte Adressierung)	<pre> UCHAR ucDB[32]; // Puffer für einen Datenblock (max. Blocksize 32) UINT uiDBAdr = 5; // Datenblock-Adresse 5 (zulässiger Wertebereich: 0..65535) UCHAR ucPwLen; // Länge des Access Passworts string sPw; // für Access Passwort string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen // ... Passwort z. B. aus Textfeld holen und in sPw speichern // setze Seriennummer (> 8 Byte zulässig) für Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, sSnr.length() / 2); // Länge der UID in Byte SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x23); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_EXT_ADR, true); // erweiterter Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_UID_LF, true); // nur falls von 8 abweichende UID-Länge SetData(FEDM_ISC_TMP_B0_REQ_BANK, (UCHAR)0x00); // Speicherbank zurücksetzen SetData(FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR, (UCHAR)0x03); // Bank User Memory SetData(FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG, true); // mit Access Passwort SetData(FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH, ucPwLen); // setze Länge Passwort SetData(FEDM_ISC_TMP_B0_ACCESS_PW, sPw); // setze Passwort SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT, uiDBAdr); // setze Datenblock-Adresse SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // ein Datenblock lesen SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // hole die Größe des Datenblocks (Blocksize) 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); // ... mach was mit dem Datenblock </pre>

[Steuerbyte] Protokoll	Beispiel
<p>[0x24] Write Multiple Blocks (normale Adressierung)</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muss zuvor ein [0x01] Inventory ausgeführt sein. Achtung: wenn noch kein [0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blockgröße auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blockgröße, muss diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) abfragen, ob die Blockgröße bereits mit [0x23] Read Multiple Blocks gelesen wurde. Mit dem folgenden Aufruf kann man die Blockgröße in der Tabelle setzen, wenn man explizit nur Schreiben will und die Blockgröße (z.B. 8) bekannt ist: SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); */ UCHAR ucDB[32]; // Puffer für einen Datenblock (max. Blockgröße 32) UCHAR ucDBAdr = 5; // Datenblock-Adresse 5 string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // ... Datenblock z. B. aus einem Textfeld holen und in ucDB[] speichern // Tabellenindex der Seriennummer ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // einen Datenblock schreiben SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, ucDBAdr); // setze Datenblock-Adresse // setze Blocksize auf 8 SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); // schreibe einen Datenblock in Tabelle SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, ucDBAdr, ucDB, 8); SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

[Steuerbyte] Protokoll	Beispiel
<p>[0x24] Write Multiple Blocks (erweiterte Adressierung)</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muss zuvor ein [0x01] Inventory ausgeführt sein. Achtung: wenn noch kein [0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blockgröße auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blockgröße, muss diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) abfragen, ob die Blockgröße bereits mit [0x23] Read Multiple Blocks gelesen wurde. Mit dem folgenden Aufruf kann man die Blockgröße in der Tabelle setzen, wenn man explizit nur Schreiben will und die Blockgröße (z.B. 8) bekannt ist: SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); */ UCHAR ucDB[32]; // Puffer für einen Datenblock (max. Blocksize 32) UINT uiDBAdr = 5; // Datenblock-Adresse 5 UCHAR ucPwLen; // Länge des Access Passworts string sPw; // für Access Passwort string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // ... Passwort z. B. aus Textfeld holen und in sPw speichern // ... Datenblock z. B. aus einem Textfeld holen und in ucDB[] speichern // Tabellenindex der Seriennummer ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // setze Seriennummer für Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_UID_LEN, sSnr.length() / 2); // Länge der UID in Byte SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x24); // Command Write Multiple Blocks SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_EXT_ADR, true); // erweiterter Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_UID_LF, true); // nur falls von 8 abweichende UID-Länge SetData(FEDM_ISC_TMP_B0_REQ_BANK, (UCHAR)0x00); // Speicherbank zurücksetzen SetData(FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR, (UCHAR)0x03); // Bank User Memory SetData(FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG, true); // mit Access Passwort SetData(FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH, ucPwLen); // setze Länge Passwort SetData(FEDM_ISC_TMP_B0_ACCESS_PW, sPw); // setze Passwort SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT, uiDBAdr); // setze Datenblock-Adresse SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x01); // einen Datenblock schreiben // setze Blocksize auf 8 SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); // schreibe einen Datenblock in Tabelle SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, uiDBAdr, ucDB, 8); SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

[Steuerbyte] Protokoll	Beispiel
[0x25] Select	<pre> CString sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x25] Select mit Option Card Information für ISO14443 Transponder	<pre> string sSnr; // für Seriennummer UCHAR ucFormat = 0; // Format Byte aus dem Antwortprotokoll // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_MODE_CINF, true); // CINF-Flag SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // das Format Byte ist in TmpData gespeichert GetData(FEDM_ISC_TMP_B0_RSP_FORMAT, &ucFormat); // Format // die Card Information sind in TmpData ab Index 2048 gespeichert UCHAR* ucCardInfo = &TmpData[2048]; // Adresse des ersten Bytes </pre>
[0x26] Reset to Ready	<pre> string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

[Steuerbyte] Protokoll	Beispiel
[0x27] Write AFI	<pre> string sSnr; // für Seriennummer UCHAR ucAFI = 0; // für AFI // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // ... AFI z. B. aus Eingabefeld holen und in ucAFI speichern // setze Seriennummer für Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); // Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // schreibe AFI in Tabelle 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x28] Lock AFI	<pre> string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x29] Write DSFID	<pre> string sSnr; // für Seriennummer UCHAR ucDSFID = 0; // für DSFID // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // ... DSFID z. B. aus Eingabefeld holen und in ucDSFID speichern // setze Seriennummer für Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); // Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // schreibe DSFID in Tabelle 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

[Steuerbyte] Protokoll	Beispiel
[0x2A] Lock DSFID	<pre> string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen 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; // für DSFID UCHAR ucAFI = 0; // für AFI UCHAR ucMemSize[] = {0, 0}; // für Memory-Size UCHAR ucICRef = 0; // für IC-Reference string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // hole AFI GetTableData(idx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_AFI, &ucAFI); // ... mach was mit AFI // ... hole alle anderen Daten nach dem gleichen Muster </pre>
[0x2C] Get Multiple Block Security Status	<pre> UCHAR ucSecStatus; // für Security Status string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_UID, sSnr); SetData(FEDM_ISC_TMP_B0_REQ_DBN, (UCHAR)0x05); // 5 Datenblöcke SetData(FEDM_ISC_TMP_B0_REQ_DB_ADR, (UCHAR)0x00); // setze 1. Datenblock-Adresse SetData(FEDM_ISC_TMP_B0_CMD, (UCHAR)0x2C); // Command Get Multiple Block // Security Status SetData(FEDM_ISC_TMP_B0_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // hole von den Blöcken 0..4 den Security Status for(int iCnt=0; iCnt<5; ++iCnt) </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre> { GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SEC_STATUS, iCnt, &ucSecStatus, 1); // ... mach was mit ucSecStatus } </pre>
<p>[0xA0] Read Config Block</p> <p>nur für I-Code 1</p>	<pre> UCHAR ucCB[4]; // Puffer für einen Datenblock (Blocksize ist immer 4) UCHAR ucCBAAdr = 0; // Datenblock-Adresse 0 string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, ucCBAAdr); // setze Datenblock-Adresse SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // hole den Datenblock GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_RxCB, ucCBAAdr, ucCB, 4); // ... mach was mit dem Datenblock </pre>
<p>[0xA1] Write Config Block</p> <p>nur für I-Code 1</p>	<pre> /* <u>Achtung:</u> Mit diesem ISO Command verändert man die Konfiguration eines Transponders und kann diesen so verändern, dass er unbrauchbar wird!! */ UCHAR ucCB[4]; // Puffer für einen Datenblock (Blocksize ist immer 4) UCHAR ucCBAAdr = 0; // Datenblock-Adresse 0 string sSnr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // ... Datenblock z. B. aus einem Textfeld holen und in ucCB[] speichern // Tabellenindex der Seriennummer ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sSnr); // setze Seriennummer für 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); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B0_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, ucCBAAdr); // setze Datenblock-Adresse // schreibe einen Datenblock in Tabelle SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxCB, ucCBAAdr, ucCB, 4); SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

6.5.3. Beispiele für die Verwendung der Tabelle mit [0xB3] Protokollen

Die nachfolgenden Beispiele demonstrieren exemplarisch den Datenaustausch mit der Tabelle FEDM_ISOTable. Auf die Auswertung der Rückgabewerte der Methoden wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll	Beispiel
[0x18] Kill für UHF-Transponder: - EPC Class1 Gen1 - EPC Class1 Gen2	<pre> /* Achtung: mit diesem ISO Command werden Transponder unwiederruflich zerstört!! string sEpc; // für EPC string sPw; // für Kill Password unsigned char ucEpcLen = 0; // Länge der EPC in Byte unsigned char ucPwLen = 0; // Länge des Kill Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // Tabellenindex der EPC ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // Länge der EPC ermitteln GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR_LEN, &ucEpcLen); // setze EPC für Addressed Mode SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // Länge der EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x18); // Command Kill SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_UID_LF, true); // von 8 abweichende UID-Länge SetData(FEDM_ISC_TMP_B3_KILL_PW_LENGTH, ucPwLen); // Länge Passwort SetData(FEDM_ISC_TMP_B3_KILL_PW, sPw); // Passwort SendProtocol(0xB3); // Kommunikation mit Leser/Transponder </pre>
[0x22] Lock Multiple Blocks für UHF-Transponder: - EPC Class1 Gen1 - EPC Class1 Gen2	<pre> /* Achtung: mit diesem ISO Command können Datenblöcke unwiederruflich schreibgeschützt werden!! string sEpc; // für EPC string sLockData; // für optionale Lockdaten string sPw; // für optionales Access Password unsigned char ucEpcLen = 0; // Länge der EPC in Byte unsigned char ucTrType = 0; // Transpondertyp unsigned char ucLockDataLen = 0; // Länge der Lockdaten unsigned char ucPwLen = 0; // Länge des optionalen Access Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern // ... Lockdaten z. B. aus Textfeld holen und in sLockData speichern, dito mit der Länge // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // Tabellenindex der EPC ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // Länge der EPC ermitteln GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR_LEN, &ucEpcLen); // Transpondertyp ermitteln GetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TRTYPE, &ucTrType); // setze EPC für Addressed Mode SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre> SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // Länge der EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x22); // Command Lock SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_EPC_LF, true); // von 8 abweichende UID-Länge SetData(FEDM_ISC_TMP_B3_REQ_TR_TYPE, ucTrType); // Transpondertyp SetData(FEDM_ISC_TMP_B3_LOCK_DATA_LENGTH, ucLockDataLen); // Länge Lockdaten SetData(FEDM_ISC_TMP_B3_LOCK_DATA, sLockData); // Lockdaten SetData(FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH, ucPwLen); // Länge Passwort if(ucPwLen > 0) SetData(FEDM_ISC_TMP_B3_ACCESS_PW, sPw); // Passwort SendProtocol(0xB3); // Kommunikation mit Leser/Transponder </pre>
<p>[0x24] Write Multiple Blocks</p> <p>für UHF-Transponder: - EPC Class1 Gen2</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muss zuvor ein [0x01] Inventory ausgeführt sein. Achtung: wenn noch kein [0xB0][0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blockgröße auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blockgröße, muss diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) abfragen, ob die Blockgröße bereits mit [0xB0][0x23] Read Multiple Blocks gelesen wurde. Mit dem folgenden Aufruf kann man die Blockgröße in der Tabelle setzen, wenn man explizit nur Schreiben will und die Blockgröße (z.B. 8) bekannt ist: SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)8); */ unsigned char ucDB[32]; // Puffer für Daten string sEpc; // für EPC string sPw; // für optionales Access Password unsigned char ucEpcLen = 0; // Länge der EPC in Byte unsigned char ucPwLen = 0; // Länge des optionalen Access Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern, dito mit Länge // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // ... Datenblock z. B. aus einem Textfeld holen und in ucDB[] speichern // Tabellenindex der EPC ermitteln int ildx = FindTableIndex(0, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_SNR, sEpc); // setze EPC für Addressed Mode SetData(FEDM_ISC_TMP_B3_REQ_UID, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, ucEpcLen); // Länge der EPC SetData(FEDM_ISC_TMP_B3_CMD, (UCHAR)0x24); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B3_MODE, (UCHAR)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B3_MODE_ADR, (UCHAR)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_UID_LF, true); // von 8 abweichende UID-Länge SetData(FEDM_ISC_TMP_B3_MODE_EXT_ADR, true); // 2 Byte DB-Adresse SetData(FEDM_ISC_TMP_B3_BANK, (UCHAR)0x00); // Speicherbank zurücksetzen SetData(FEDM_ISC_TMP_B3_BANK_BANK_NR, (UCHAR)0x01); // EPC Speicherbank SetData(FEDM_ISC_TMP_B3_BANK_ACCESS_FLAG, true); // mit Access Password SetData(FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH, ucPwLen); // setze Länge Passwort SetData(FEDM_ISC_TMP_B3_ACCESS_PW, sPw); // setze Passwort SetData(FEDM_ISC_TMP_B3_REQ_DB_ADR_EXT, (UINT)0); // setze Datenblock-Adresse SetData(FEDM_ISC_TMP_B3_REQ_DBN, (UCHAR)0x06); // sechs Datenblöcke schreiben SetData(FEDM_ISC_TMP_B3_REQ_DB_SIZE, (UCHAR)0x02); // Größe Datenblock // setze Blocksize in Tabelle auf 2 SetTableData(ildx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, (UCHAR)2); </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre>// schreibe Datenblöcke in Tabelle for(int iAdr=0; iAdr<6; ++iAdr) SetTableData(iIdx, FEDM_ISC_ISO_TABLE, FEDM_ISC_DATA_TxDB, iAdr, &ucDB[iAdr*2], 2); SendProtocol(0xB3); // Kommunikation mit Leser/Transponder</pre>

6.6. Tabelle für den Buffered Read Mode

Mit dem [0x21] Read Buffer (nur für Long-Range-Reader ISC.LR200) bzw. [0x22] Read Buffer können Daten von mehreren im Antennenfeld des Lesers befindliche Transpondern gelesen werden. Zur strukturierten Ablage dieser Transponderdaten ist innerhalb der Leserkasse FEDM_ISCReader die Tabelle m_BRMTable (protected) implementiert, die die Protokoll Daten eines Transponders als einen Tabelleneintrag verwaltet. Wenn z.B. mit einem Protokoll Daten von drei Transpondern kommen, werden drei Tabelleneinträge mit den Transponderdaten gefüllt.

Die Tabelle hat voreingestellt eine Größe von 0. Sie muss deshalb vor der ersten Verwendung mit der Methode SetTableSize initialisiert werden. Die Größe wird bestimmt durch die maximale Anzahl an Transpondern, die sich gleichzeitig im Antennenfeld des Lesers befinden können.

Die Daten werden in die Tabelle immer beginnend ab dem Index 0 eingebaut. Das bedeutet, dass mit jedem [0x21] Read Buffer (bzw. [0x22]) alte Tabellendaten überschrieben werden. Die (protected) Variable m_iBRMTableLength der Leserkasse gibt darüber Auskunft, wieviele Einträge gültig sind.

Der Zugriff auf die Tabellendaten erfolgt immer mit den Methoden GetTableData und SetTableData der Leserkasse FEDM_ISCReader. Als Parameter uiTableID wird immer FEDM_ISC_BRM_TABLE übergeben. Eine Auflistung und Erläuterung der Zugriffskonstante im Parameter uiDataID findet sich in [7.3.4. Konstanten für uiDataID](#).

Zum Finden bestimmter Tabelleneinträge, z. B. anhand der Seriennummer, steht die Methode FindTableIndex bereit. Weitere Abfragemethoden sind: GetTableSize und GetTableLength. Alle genannten Methoden sind im Kapitel zur Leserkasse s. 6.1. FEDM_ISCReader beschrieben.

Der Speicherplatzbedarf für einen Tabelleneintrag beträgt 1104 Byte.

6.6.1. Beispiele für die Verwendung der Tabelle

Die nachfolgenden Beispiele demonstrieren exemplarisch den Datenaustausch mit der Tabelle FEDM_BRMTable. Auf die Auswertung der Rückgabewerte der Methoden wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll	Beispiel
[0x21] Read Buffer	<pre>// das Beispiel zeigt das Lesen von Datensätzen mit Seriennummer, Datenblock und Timer-Wert UCHAR ucDataSets = 1; // Anzahl angeforderter Datensätze UCHAR ucRecSets = 0; // Anzahl Datensätze in Protokoll UCHAR ucDB[4]; // Puffer für einen Datenblock UCHAR ucAnt; // Antennennummer UINT uiTimer = 0; // für Timer-Wert __int64 i64Snr = 0; // für Seriennummer BOOL bSNR = FALSE; // Flag für Seriennummer in Datensatz BOOL bDB = FALSE; // Flag für Datenblock in Datensatz BOOL bANT = FALSE; // Flag für Antennennummer in Datensatz</pre>

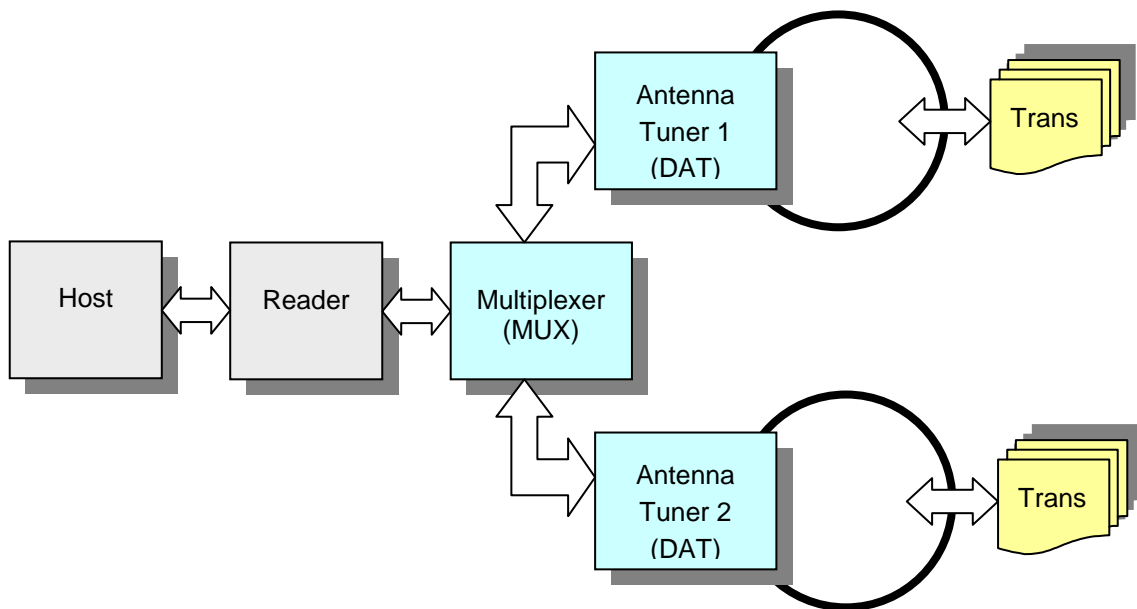
[Steuerbyte] Protokoll	Beispiel
	<pre> BOOL bTimer = FALSE; // Flag für Timer in Datensatz SetData(FEDM_ISCLR_TMP_BRM_SETS, ucDataSets); SendProtocol(0x21); // lese Daten von Transponder mit Buffered Read Mode GetData(FEDM_ISCLR_TMP_BRM_TRDATA_SNR, &bSNR); GetData(FEDM_ISCLR_TMP_BRM_TRDATA_DB, &bDB); GetData(FEDM_ISCLR_TMP_BRM_TRDATA_ANT, &bANT); GetData(FEDM_ISCLR_TMP_BRM_TRDATA_TIME, &bTimer); GetData(FEDM_ISCLR_TMP_BRM_RECSETS, &ucRecSets); // Alle Transponderdaten sind in der Tabelle enthalten for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_BRM_TABLE); iCnt++) { if(bSNR) // hole Seriennummer GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_SNR, &i64Snr); if(bDB) // hole Datenblock 1 GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_RxDB, 1, ucDB, 4); if(bANT) // hole Antennennummer GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_ANT_NR, ucAnt); if(bTIMER) // hole Timer-Wert GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_TIMER, &uiTimer); } </pre>
[0x22] Read Buffer	<pre> // das Beispiel zeigt das Lesen von Datensätzen mit Seriennummer, Datenblöcken, Timer-Wert, // Datumsfeld und Antennennummer UINT uiDataSets = 1; // Anzahl angeforderter Datensätze UINT uiRecSets = 0; // Anzahl Datensätze in Protokoll UCHAR ucAnt; // Antennennummer UINT uiTimer = 0; // für Uhrzeit-Wert UCHAR ucDate[5]; // für Datumsfeld UCHAR ucInput; // für Input-Byte UCHAR ucStatus; // für Status-Byte UCHAR ucSize; // für Blockgröße eines Datenblocks UINT uiDBN = 0; // für Anzahl der Datenblöcke string sSnr; // für Seriennummer string sDB; // für Datenblöcke BOOL bSNR = FALSE; // Flag (in TR-DATA1) für Seriennummer im Datensatz BOOL bDB = FALSE; // Flag (in TR-DATA1) für Datenblock im Datensatz BOOL bANT = FALSE; // Flag (in TR-DATA1) für Antennennummer im Datensatz BOOL bTime = FALSE; // Flag (in TR-DATA1) für Uhrzeit im Datensatz BOOL bDate = FALSE; // Flag (in TR-DATA1) für Datumsfeld im Datensatz BOOL bExt = FALSE; // EXTENSION-Flag (in TR-DATA1) im Datensatz: zeigt an, dass ein // zweites TR-DATA Byte im Protokoll enthalten ist, welches weitere // Flags enthält BOOL bInput = FALSE; // Flag (in TR-DATA2) für Input und Status im Datensatz SetData(FEDM_ISC_TMP_ADV_BRM_SETS, uiDataSets); SendProtocol(0x22); // lese Daten von Transponder mit 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); </pre>

[Steuerbyte] Protokoll	Beispiel
	<pre> 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); // Alle Transponderdaten sind in der Tabelle enthalten for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_BRM_TABLE); iCnt++) { if(bSNR) // hole Seriennummer GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_SNR, sSnr); if(bDB) // hole alle Datenblöcke { // hole Anzahl der Datenblöcke GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_DBN, &uiDBN); // hole die Blockgröße GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_BLOCK_SIZE, &ucSize); // hole Datenblöcke for(int i=0; i<uiDBN; ++i) { GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_RxDB, i, sDB); // tu was mit den Datenblöcken } } if(bANT) // hole Antennennummer GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_ANT_NR, ucAnt); if(bTime) // hole Uhrzeit GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_TIMER, &uiTimer); if(bDate) // hole Datumsfeld GetTableData(iCnt, FEDM_ISC_BRM_TABLE, FEDM_ISC_DATA_DATE, ucDate, 5); if(bExt && bInput) // hole Input- und 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

Die Klasse **FEDM_ISCFunctionUnit** repräsentiert eine externe HF- oder UHF-Funktionseinheit in der Antennen-Leitung des Lesers. Die Klasse hat keine Basisklasse. Zum Verständnis der Funktionseinheiten ist das Systemhandbuch H30701-xe-ID-B (HF) bzw. H80302-xe-ID-B (UHF) zwingend erforderlich. Nützliche Informationen finden sich auch in den Montageanleitungen zu den jeweiligen Funktionseinheiten.

In Anbetracht der Tatsache, dass eine Funktionseinheit immer einen Leser als Kommunikationsbrücke voraussetzt, kann auch die Klasse **FEDM_ISCFunctionUnit** nur instanziiert werden, wenn ein Leserobjekt vom Typ **FEDM_ISCReader** oder **FEDM_ISCReaderModule** vorhanden ist.



Die Abbildung verdeutlicht auch, dass externe Funktionseinheiten baumartig mit dem Leser verbunden sind. Diese Topologie wird in der Leserklasse durch eine Liste für nachfolgende Funktionseinheiten nachgebildet. Mit dieser Liste ist es möglich, durch den Baum, beginnend mit der ersten Funktionseinheit, zu traversieren.

Standardmäßig verwaltet die erste Funktionseinheit – i.d.R. ein Multiplexer – die Zeiger der dynamisch allozierten, nachfolgenden Funktionseinheiten. Das hat den Vorteil, dass der Destruktor des ersten Knotens den Speicher aller nachfolgenden Knoten freigibt. Ist dieses Verhalten nicht erwünscht, weil z.B. mit statisch angelegten Objekten gearbeitet wird, dann muss dieses Verhalten mit dem Methodenaufruf **SetManageChildMode(false)** abgeschaltet werden.

6.7.1. Konstruktor

Dem Konstruktor wird ein Zeiger auf das Leserobjekt vom Typ **FEDM_ISCReader** und der Typ der Funktionseinheit (Konstanten s. FEDM_ISCFunctionUnit.h) übergeben. Das Leserobjekt muss einen Leser repräsentieren, mit dem die Funktionseinheit physisch verbunden ist.

6.7.2. Implementierte Datencontainer

Datencontainer	Beschreibung
TmpData	für allgemeine temp. Protokolldaten

Die Größe der Datencontainer wird statisch im Konstruktor der Klasse festgelegt. Alle Datencontainer werden im Konstruktor mit 0x00 initialisiert.

6.7.3. Implementierte Listen

Liste	Beschreibung
m_ChildList	Liste für Funktionseinheiten, die physisch am Ausgang/den Ausgängen der Funktionseinheit angeschlossen sind.

Die Liste kann nur verwendet werden, wenn eine Funktionseinheit weitere Funktionseinheiten treiben kann. Der Multiplexer ID ISC.ANT.MUX ist z. B. solch eine Funktionseinheit.

6.7.4. Methoden (public)

Methode	Beschreibung
SendProtocol	Die zentrale Kommunikationsmethode. Nähere Beschreibung in 6.7.6. Beispiele für die Verwendung der Methode SendProtocol .
AddChild	Methode zum Hinzufügen einer neuen Funktionseinheit in die Liste.
DelChild	Methode zum Entfernen einer Funktionseinheit aus der Liste.
DelChildList	Methode zum Löschen aller Funktionseinheiten aus der Liste
GetChild	Methode zur Rückgabe eines Objektzeigers aus der Liste.
SetFUType	Setzt den Typ der Funktionseinheit Dabei wird der Inhalt der Liste der Funktionseinheiten gelöscht.
GetFUType	Ermittelt den Typ der Funktionseinheit.
SetManageChildMode	Stellt den Management Mode für Childs ein. Per Default zerstört der Parent seine Childs. Soll sich die Applikation um die Speicherverwaltung der Childs kümmern, schaltet man mit dem Parameter bDeleteInternal=false die interne Verwaltung ab.
GetData	Überladene Methode zum Lesen eines Parameterwertes aus einem Datencontainer. Der Aufruf wird an die Klasse FEDM_Base weitergeleitet, nachdem aus der Zugriffskonstanten der Typ des Datencontainers (Speichertyp-Konstante) ermittelt wurde. SetData unterstützt die Datentypen: bool, BOOL, UCHAR, UCHAR-Array, UINT, __int64, CString bzw AnsiString, STL-string und C-Zeichenkette.
SetData	Überladene Methode zum Schreiben eines Parameterwertes in einen Datencontainer. Der

Methode	Beschreibung
	Aufruf wird an die Klasse FEDM_Base weitergeleitet, nachdem aus der Zugriffskonstante der Typ des Datencontainers (Speichertyp-Konstante) ermittelt wurde. SetData unterstützt die Datentypen: bool, BOOL, UCHAR, UCHAR-Array, UINT, __int64, CString bzw AnsiString, STL-string und C-Zeichenkette.
GetLastError	Ermittelt den letzten Fehlercode
GetLastStatus	Ermittelt das Statusbyte des letzten Protokolls.
GetErrorText	Ermittelt zum übergebenen Fehlercode einen Text. Der Fehlercode kann auch aus dem Bereich der Funktionssammlung ID FEISC oder der darunter liegenden Kommunikationsbibliotheken kommen.
GetStatusText	Ermittelt zum übergebenen Statusbyte einen Kurztext.

6.7.5. Notwendige Initialisierungen

Vor der ersten Verwendung der Protokollmethode muss die Adresse der Funktionseinheit eingestellt werden:

Funktionseinheit	Adresseinstellung
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. Beispiele für die Verwendung der Methode SendProtocol

Die Methode *SendProtocol* ist von zentraler Bedeutung für den Protokolltransfer. Aus diesem Grund wird für jedes Steuerbyte ein Beispiel aufgeführt, das verdeutlichen soll, welche Daten mit welchen Zugriffskonstanten vor jedem Protokolltransfer im Datencontainer zu speichern sind und welche Daten nach dem Protokolltransfer zur Verfügung stehen.

Alle Zugriffskonstanten sind in der Datei FEDM_ISCFunctionUnitID.h aufgelistet und sollten eingehend zusammen mit den im Systemhandbuch zur Funktionseinheit angeführten Erklärung der Protokolldaten studiert werden.

Auf die Auswertung der Rückgabewerte der Methoden wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll	Beispiel
ID ISC.DAT	[0xC0] Get Firmware Version UCHAR ucFirmware[7]; // Puffer für Firmware Informationen 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); // Kapazität 1 SetData(FEDM_ISC_FU_TMP_DAT_ANT_VAL_C2, (UCHAR)0x9F); // Kapazität 2 SendProtocol(0xC2);
	[0xC3] Get Antenna Values UCHAR ucAntValues[6]; // Puffer für Tuningwerte SendProtocol(0xC3); GetData(FEDM_ISC_FU_TMP_DAT_ANT_VAL, ucAntValues, 6);
	[0xC4] Set Outputs SetData(FEDM_ISC_FU_TMP_DAT_OUT, (UCHAR)1); // Ausgang 1 wird geschaltet 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; // neue Adresse SetData(FEDM_ISC_FU_TMP_DAT_NEW_ADR, ucAdr); // neue Adresse für Funktionseinheit SendProtocol(0xCA); // neue Adresse wird wirksam SetData(FEDM_ISC_FU_TMP_DAT_ADR, ucAdr); // neue Adresse für nachfolgende Protokolle übernehmen
	[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); // Ausgang für Eingang 1 auf 1 SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH2, (UCHAR)8); // Ausgang für Eingang 2 auf 8 SendProtocol (0xDD);
	[0xDE] CPU Reset	SendProtocol (0xDE);
	[0xDF] Get Firmware Version	UCHAR ucFirmware[7]; // Puffer für Firmware Informationen SendProtocol (0xDF); GetData (FEDM_ISC_FU_TMP_SOFTVER, ucFirmware, 7);
ID ISC.ANT.UMUX	[0xDC] Detect/Get Power	UCHAR ucPower[5]; // Puffer für Power Informationen SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // immer auf 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); // immer auf 0 SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH1, (UCHAR)1); // selektiere Ausgang 1 SendProtocol (0xDD); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);
	[0xDE] CPU Reset	SetData (FEDM_ISC_FU_TMP_FLAGS, (UCHAR)0); // immer auf 0 SendProtocol (0xDE); GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, &ucUMuxStatus);
	[0xDF] Get Firmware Version	UCHAR ucFirmware[7]; // Puffer für Firmware Informationen 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

Die Klasse **FedmlscPeopleCounter** repräsentiert eine externe Funktionseinheit vom Typ People-Counter **ID ISC.ANT1690/600-GPC** bzw. **ID ISC.ANT1700/740-GPC** am RS485-Port des Lesers. Die Klasse ist von der Basisklasse **FedmlscPeripheralDevice** abgeleitet. Zum Verständnis des People-Counters ist das Systemhandbuch H01011-0e-ID-B zwingend erforderlich. Nützliche Informationen finden sich auch in den Montageanleitungen zum People-Counter.

Bis zu drei People-Counter können am Leser angeschlossen werden und sind durch ihre individuelle Busadresse identifizierbar. Der Leser führt nach dem Einschalten, bzw. nach einem [0x64] System Reset (ACC) einen Detektionsprozess durch. Eine Applikation kann mit der Methode **FEDM_ISCReaderModule::ReadReaderInfo()** die erkannten People-Counter abfragen und mit der Methode **FEDM_ISCReader::GetPeripheralDevices()** holt man sich anschließend eine nach der Busadresse sortierte Liste mit intern erzeugten People-Counter-Objekten.

6.8.1. Methoden (public)

Methode	Beschreibung
SetOutput	Setzt 1, 2 oder 3 digitale Ausgänge
SetCounter	Setzt alle 4 Zähler auf den übergebenen Stand
GetCounter	Gibt die Zählerstände der 4 Zähler zurück

6.8.2. Beispiel für die Verwendung der Klasse

Das nachfolgend gezeigte Beispiel ist für Reader im Host-Mode oder Buffered-Read-Mode geeignet. Die Initialisierungsphase mit der Abfrage der angeschlossenen People-Counter ist nur einmal erforderlich.

```
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;

// zuerst angeschlossene People-Counter abfragen
FEDM_ISC_READER_INFO* pInfo = m_Reader.GetReaderInfo(); // Zeiger auf Infostruktur holen
if (!pInfo->blsMode0x61Read)
{
    m_Reader.SetData(FEDM_ISC_TMP_READER_INFO_MODE, (unsigned char)0x61)
    int back = m_Reader.SendProtocol(0x66);
    if(back)
        return; // irgendein Problem aufgetreten
}

pPDMap = m_Reader.GetPeripheralDevices();
```

```

// 1. wichtiger Test
if (pPDMMap == NULL)
    return; // keine People-Counter angeschlossen oder in der Leser-Konfiguration nicht aktiviert

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

// 2. wichtiger Test
if (itor == pPDMMap->end())
    return; // keine People-Counter mit Busadresse 1 gefunden

// 3. wichtiger Test
if (dynamic_cast< FedmlscPeopleCounter*>(itor->second) == NULL)
    return; // keine People-Counter Klasse

pPeopleCounter = (FedmlscPeopleCounter*)itor->second;

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

```

Wichtiger Hinweis: Die sortierte Liste mit den Pointern bzw. die Pointer selbst stehen unter der Verwaltung der Leserkasse. Diese erzeugt die Objekte und löscht sie auch wieder. Deshalb dürfen die Pointer niemals vom Applikationsprogrammierer freigegeben werden.

6.8.3. Beispiel für automatische Notifikation

Wenn der Leser im Notification-Mode arbeitet, bietet es sich an, auch die Zählerstände per TCP/IP-Notification an die Applikation geliefert zu bekommen. Dazu muss der Leser entsprechend konfiguriert werden: s. Parameter im Konfigurationszweig

OperatingMode.NotificationMode.GatePeopleCounter.Transmission.Destination

Die Realisierung in der Applikation ist sehr einfach und erfolgt ohne die im Beispiel zuvor genannte People-Counter-Klasse. Dazu meldet man lediglich einen Task in der Funktionsbibliothek FEISC an und stellt eine Callback-Funktion bereit. Da jedes interne Reader-Objekt in der FEISC nur einen Task bearbeiten kann, ergibt sich möglicherweise ein Konflikt, wenn man auch die Transponderdaten per TCP/IP-Notification annehmen muss. In diesem Fall kann man ein neues Reader-Objekt in FEISC erzeugen und für den Empfang der Zählerstände nutzen.

```

BOOL CPeopleCounterSampleDlg::OnInitDialog()
{
    FEISC_TASK_INIT TaskInit;

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

    TaskInit.cbFct2 = cbsFct; // callback function
    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)
    unsigned char ucCmd,  // [in] reader command
    unsigned char* 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(unsigned char* ucRspData, int iRspLen)
{
    unsigned int uiCnt1 = ucRspData[3] | ucRspData[2] << 8 | ucRspData[1] << 16 | ucRspData[0] << 24;
    unsigned int uiCnt2 = ucRspData[7] | ucRspData[6] << 8 | ucRspData[5] << 16 | ucRspData[4] << 24;
    unsigned int uiCnt3 = ucRspData[11] | ucRspData[10] << 8 | ucRspData[9] << 16 | ucRspData[8] << 24;
    unsigned int 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. Anhang

7.1. Unterstützte OBID® Leser

Leser	Anmerkungen
ID ISC.M02	
ID ISC.MR/PR100	alle Schnittstellen
ID ISC.PRH100 / PRH101 / PRH102	alle Schnittstellen
ID ISC.MR/PR101	alle Schnittstellen
ID ISC.MR102	alle Schnittstellen
ID ISC.PRH102	alle Schnittstellen
ID ISC.PRHD102	alle Schnittstellen
ID ISC.PRH200	alle Schnittstellen
ID ISC.MR200	alle Schnittstellen
ID ISC.LR200	
ID ISC.LR1002	alle Schnittstellen
ID ISC.LR2000	alle Schnittstellen
ID ISC.LR2500-A	alle Schnittstellen
ID ISC.LR2500-B	alle Schnittstellen
ID ISC.MU02	
ID ISC.MRU102	alle Schnittstellen
ID ISC.MRU200	alle Schnittstellen
ID ISC.LRU1000	alle Schnittstellen
ID ISC.LRU1002	alle Schnittstellen
ID ISC.LRU2000	alle Schnittstellen
ID ISC.LRU3000	alle Schnittstellen
ID CPR.02	
ID CPR.M02	alle Schnittstellen
ID CPR.04	alle Schnittstellen
ID CPR30.xx	alle Schnittstellen
ID CPR40.xx	alle Schnittstellen
ID CPR44.xx	alle Schnittstellen
ID CPR46.xx	alle Schnittstellen
ID CPR47.xx	alle Schnittstellen
ID CPR50.xx	alle Schnittstellen

Leser	Anmerkungen
ID CPR52.xx	alle Schnittstellen
ID CPR60.xx	alle Schnittstellen
ID MAX50.xx	alle Schnittstellen
ID myAXXESS onTop-S	alle Schnittstellen

7.2. Unterstützte Transponder

Die Unterstützung von Transpondertypen ist zunächst einmal abhängig von der im Leser implementierten Firmware. Genaue Auskunft erhält man im Systemhandbuch des Lesers.

Nachfolgend sind die Transpondertypen aufgeführt, die zum Zeitpunkt der Entwicklung der Bibliothek bekannt waren.

Transponder	Typkennung	Anmerkung
I-CODE 1	0x00	HF-Transponder
Tag-it	0x01	HF-Transponder
ISO 15693	0x03	HF-Transponder
ISO 14443-A	0x04	HF-Transponder
ISO 14443-B	0x05	HF-Transponder
EPC	0x06	HF-Transponder (EPC-Typen 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
ISO 18000-6-A	0x80	UHF-Transponder
ISO 18000-6-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

Informationen zum einzelnen Status finden sich im Internet unter dem Suchbegriff *Transmission Control Protocol*

TCP-Status	Wert
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. Liste der Konstanten

Alle aufgeführten Konstanten sind in FEDM_ISC.h bzw. FEDM_ISCFunctionUnitID.h definiert.

7.4.1. Interne Konstanten

Konstante	Beschreibung
FEDM_ISC_MAX_RFC_EEDATA_MEM	Größe des Datencontainers EEData
FEDM_ISC_MAX_RFC_RAMDATA_MEM	Größe des Datencontainers RAMData
FEDM_ISC_MAX_TMPDATA_MEM	Größe des Datencontainers TmpData
FEDM_ISC_BRM_TABLE_RxDB_SIZE	Größe des Datenpuffers für Receive-Datenblöcke je Transponder
FEDM_ISC_ISO_TABLE_TxDB_SIZE	Größe des Datenpuffers für Transmit-Datenblöcke je Transponder
FEDM_ISC_ISO_TABLE_RxDB_SIZE	Größe des Datenpuffers für Receive-Datenblöcke je Transponder
FEDM_ISC_ISO_TABLE_SEC_STATUS_SIZE	Anzahl Security-Bytes je Transponder (1 Byte je Datenblock)
FEDM_ISC_ISO_TABLE_EPC_BANK_SIZE	Größe des Datenpuffers für Transmit/Receive-Datenblöcke der EPC-Bank je Transponder (nur EPC Class1 Gen2)
FEDM_ISC_ISO_TABLE_TID_BANK_SIZE	Größe des Datenpuffers für Transmit/Receive-Datenblöcke der TID-Bank je Transponder (nur EPC Class1 Gen2)
FEDM_ISC_ISO_TABLE_RES_BANK_SIZE	Größe des Datenpuffers für Transmit/Receive-Datenblöcke der RESERVED-Bank je Transponder (nur EPC Class1 Gen2)
FEDM_ISC_FU_MAX_TMPDATA_MEM	Größe des Datencontainers TmpData für FEDM_ISCFunctionUnit

7.4.2. Allgemeine Konstanten

Konstante	Beschreibung
FEDM_ISC_TYPE_...	Lesertyp entsprechend dem Protokoll [0x65] Software Version
FEDM_ISC_NAME_...	Leserbezeichnung entsprechend dem Systemhandbuch
FEDM_ISC_NAME_..._UC	Leserbezeichnung in Unicode entsprechend dem Systemhandbuch
FEDM_ISC_TR_TYPE_...	Kennung des Transpondertyps entsprechend dem Anhang im Systemhandbuch
FEDM_ISC_TR_NAME_...	Transponderbezeichnung entsprechend dem Anhang im Systemhandbuch
FEDM_ISC_EPC_TYPE_...	Kennung der EPC-Typen; (EPC = Electronic Product Code)
FEDM_ISC_FU_TYPE_...	Kennung der Typen der Funktionseinheiten
FEDM_ISC_ISO_MODE_...	ISO-Command Modes bzw. Mode-Flags
FEDM_ISC_ISO_BANK_...	Identifizier für Speicherbank des EPC Class1 Gen2 Transponders
FEDM_ISC_ISO_MFR_...	Manufacturer Code für [0xB1] ISO15693 Host-Commands

7.4.3. Konstanten für uiTableID

Konstante	Beschreibung
FEDM_ISC_BRM_TABLE	Tabellen-ID für BRM-Tabelle
FEDM_ISC_ISO_TABLE	Tabellen-ID für ISO-Tabelle

7.4.4. Konstanten für uiDataID

Konstante	Beschreibung/Verwendung																																				
<div>FEDM_ISC_DATA_TRTYPE</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table></div>	BRM-Table	ISO-Table	X	X	<div>Transpondertyp</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>X</td><td></td><td>X</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		X		X			
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex		X		X																																	
<div>FEDM_ISC_DATA_SNR</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table></div>	BRM-Table	ISO-Table	X	X	<div>Seriennummer</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>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></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X		X	X	X	SetTableData					X	X	X	FindTableIndex					X	X	X
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X		X	X	X																														
SetTableData					X	X	X																														
FindTableIndex					X	X	X																														
<div>FEDM_ISC_DATA_RxDB FEDM_ISC_DATA_RxDB_EPC_BANK FEDM_ISC_DATA_RxDB_TID_BANK FEDM_ISC_DATA_RxDB_RES_BANK</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table></div>	BRM-Table	ISO-Table	X	X	<div>Datenblöcke aus Empfangsprotokoll</div> <div>Hinweis: GetTableData und SetTableData (nur ISO-Table) für Datenblöcke verwenden !!</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	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X			X	X																														
SetTableData			X			X	X																														
FindTableIndex																																					
<div>FEDM_ISC_DATA_TxDB FEDM_ISC_DATA_TxDB_EPC_BANK FEDM_ISC_DATA_TxDB_TID_BANK FEDM_ISC_DATA_TxDB_RES_BANK</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>Datenblöcke für Sendeprotokoll</div> <div>Hinweis: GetTableData und SetTableData (nur ISO-Table) für Datenblöcke verwenden !!</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																																					
<div>FEDM_ISC_DATA_TIMER</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table></div>	BRM-Table	ISO-Table	X		<div>Timer-Wert aus Empfangsprotokoll [0x21] bzw. [0x22] Read Buffer</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>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></div>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X	X				SetTableData								FindTableIndex							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X	X																																	
SetTableData																																					
FindTableIndex																																					
<div>FEDM_ISC_DATA_RxCB</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>Konfigurations-Datenblock aus Empfangsprotokoll</div> <div>Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !!</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																																					

Konstante	Beschreibung/Verwendung																																
FEDM_ISC_DATA_TxCB	<div>Konfigurations-Datenblock für Sendeprotokoll (nur für ISO-Tabelle)</div> <div>Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !!</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 aus [0xB0] [0x2B] Get System Information</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 (nur für ISO14443A Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 (nur für ISO14443A Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 (nur für ISO14443B Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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 (nur für ISO14443B Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><td></td><td>bool</td><td>UCHAR</td><td>UCHAR[]</td><td>UINT</td><td>__int64</td><td>CString</td><td>string</td></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																																	

Konstante	Beschreibung/Verwendung																																				
<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 (nur für ISO14443B Transponder) aus Empfangsdaten [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 (nur für ISO14443B Transponder) aus Empfangsdaten [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 (nur für ISO14443B Transponder) aus Empfangsdaten [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 (nur für ISO14443B Transponder) aus Empfangsdaten [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 (nur für ISO14443B Transponder) aus Empfangsdaten [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 aus Empfangsdaten [0xB0] [0x23] Read Multiple Blocks</div> <div>Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !!</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																																					

Konstante	Beschreibung/Verwendung																																				
FEDM_ISC_DATA_BLOCK_SIZE	Blocksize aus Empfangsdaten [0xB0] [0x23] Read Multiple Blocks bzw. [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	Speichergroße aus [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-Referenz aus [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	Datenblock-Adresse aus Empfangsprotokoll [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	Anzahl Datenblöcke aus Empfangsprotokoll [0x21] bzw. [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, ob Blocksize mit [0xB0] [0x23] Read Multiple Blocks gesetzt wurde																																				
<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, ob Transponder selektiert ist. Wird mit [0xB0][0x25] Select gesetzt																																				
<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																																				

Konstante	Beschreibung/Verwendung																																				
FEDM_ISC_DATA_IS_ISO14443_4_INFO <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	Flag, ob Daten mit [0xB2] [0x2B] ISO14443-4 Transponder Info gesetzt wurden <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							
BRM-Table	ISO-Table																																				
	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData	X	X	X	X																																	
SetTableData	X	X		X																																	
FindTableIndex																																					
FEDM_ISC_DATA_EPC <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	EPC; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory. Der EPC als String mit allen Feldern in der Darstellung: "xx.xxxxxx.xxxxxx.xxxxxx" (Header.DomainManager.ObjectClass.Seriennummer) oder EPC aus [0x22] Read Buffer mit variabler Länge <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>X</td><td>X</td></tr></table>		bool	UCHAR	UCHAR[]	UINT	__int64	CString	string	GetTableData			X			X	X	SetTableData								FindTableIndex						X	X
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X			X	X																														
SetTableData																																					
FindTableIndex						X	X																														
FEDM_ISC_DATA_EPC_TYPE <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	EPC-Typ; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory. Der EPC-Typ wird aus dem Feld EPC-Header ermittelt. <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			
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X																																	
SetTableData																																					
FindTableIndex		X		X																																	
FEDM_ISC_DATA_EPC_HEADER <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	Feld EPC Header; (EPC = Electronic Product Code) aus Empfangsdaten [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			
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData				X		X	X																														
SetTableData																																					
FindTableIndex				X																																	
FEDM_ISC_DATA_EPC_DOMAIN <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	Feld EPC-DomainManager; (EPC = Electronic Product Code) aus Empfangsdaten [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
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData					X	X	X																														
SetTableData																																					
FindTableIndex					X	X	X																														
FEDM_ISC_DATA_EPC_OBJECT <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	Feld EPC-ObjectClass; (EPC = Electronic Product Code) aus Empfangsdaten [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
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData					X	X	X																														
SetTableData																																					
FindTableIndex					X	X	X																														

Konstante	Beschreibung/Verwendung																																				
FEDM_ISC_DATA_EPC_SNR <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	Feld EPC-Seriennummer; (EPC = Electronic Product Code) aus Empfangsdaten [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
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData					X	X	X																														
SetTableData																																					
FindTableIndex					X	X	X																														
FEDM_ISC_DATA_SNR_LEN <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	Länge der Seriennummer aus Empfangsprotokoll [0x22] Read Buffer oder [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>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							
BRM-Table	ISO-Table																																				
X	X																																				
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData		X		X		X	X																														
FindTableIndex																																					
FEDM_ISC_DATA_ANT_NR <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Antennennummer aus Empfangsprotokoll [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			
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_DATE <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Datumsfeld aus Empfangsprotokoll [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							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_INPUT <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Input-Byte aus Empfangsprotokoll [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							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_STATE <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Status-Byte aus Empfangsprotokoll [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							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData		X	X	X		X	X																														
SetTableData																																					
FindTableIndex																																					
FEDM_ISC_DATA_MAC_ADR <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Status-Byte aus Empfangsprotokoll [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							
BRM-Table	ISO-Table																																				
X																																					
	bool	UCHAR	UCHAR[]	UINT	__int64	CString	string																														
GetTableData			X			X	X																														
SetTableData																																					
FindTableIndex																																					

Konstante	Beschreibung/Verwendung				
<p>FEDM_ISC_DATA_ALL</p> <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 für die Methode ResetTable zum Initialisieren aller Tabellenwerte
BRM-Table	ISO-Table				
X	X				

7.5. Änderungshistorie

V4.06.01

- Unterstützung für die neuen Leser: **ID ISC.PRH200**, **ID ISC.LRU1002**, **ID myAXXESS onTop**
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

V4.05.00

- Unterstützung für den neuen Leser: **ID CPR47.xx**
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Neuer Namensraum **OBID::Fedm::Core::i_scan::ReaderCommand** enthält strukturiert alle Command-Parameter
- Klasse **FEDM_ISCReader**: Neue Methoden SetCommandPara und GetCommandPara.
- Unterstützung für **ISO 18000-3M3** Transponder.
- Unterstützung für neue ISO 15693-Transponder: **STM M24LRxxE-R** und **STM LRIS64K**
- **Neue TagHandler-Klassen** für **ISO 18000-3M3**, **STM M24LRxxE-R** und **STM LRIS64K**
- **TagHandler-Klasse** für **NXP ICode SLI-L**: neue Methode PasswordProtectAFI
- **TagHandler-Klasse** für **EPC Class 1 Gen 2**:
 1. Unterstützung der Recommissioning-Bits in der Kill-Methode.
 2. Unterstützung des Extended PC
 3. Fehlerkorrektur für 8-Byte lange EPCs.
 4. Fehlerkorrektur in WriteEPC: Rückgabe eines Fehlercodes anstelle der EPC-Länge
- Beta-Version eines neuen und vereinfachten APIs mit der Klasse **ReaderModule** im Namensraum **OBID::Fedm::Core::i_scan**. Dokumentation auf Anfrage.

V4.03.00

- Unterstützung für den neuen Leser: **ID CPR46.xx**
- Unterstützung für neue Transponder: **Innovatron (ISO 14443B')** und **ASK CTx**
- **Neue TagHandler-Klassen** für Innovatron (ISO 14443B') und ASK CTx
- **Gesicherte Datenübertragung** über Serielle Schnittstelle und USB
- Unterstützung des People Counter im Notification Mode
- Buffered-Read-Mode Tabelle unterstützt Richtungsinformation des People Counters
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Umbenennung von Namespaces:

Bisher	Neu
--------	-----

Bisher	Neu
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

Anm: xxMode steht für NotificationMode oder ScanMode

V4.02.00

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für den neuen Leser: **ID ISC.LR1002**
- **TagHandler-Klasse für ISO 14443-4 Mifare DESFire mit FlexSoft- und SAM-Crypto:** Fehlerkorrekturen in SetConfiguration mit Werten 1 und 2 in Parameter option
- **ISO 14443-3 bzw. -4 Transponder:** Optimiertes Select-Verfahren in der Methode TagSelect.
- **EPC Class 1 Gen 2:**
 1. Unterstützung für non-addressed mode.
 2. Fehlermeldung (-158) vor Ausführung von Transponder-Commands im addressed-mode, wenn die TID nicht Bestandteil der UID ist, aber UID = EPC + TID konfiguriert ist.
- **TagHandler-Klasse für EPC Class 1 Gen 2:**
 5. ISO-Errorcode wird in der Klasse gespeichert.
 6. Längenprüfung für EPC und Passwort in allen relevanten Methoden.
 7. Methode GetTidOfUid gibt TID zurück, auch wenn EPC-Länge = 0 ist.
- Klasse **FEDM_ISCReaderModule:** Umbenennung der Methode GetNonAddressedTagHandler in CreateNonAddressedTagHandler.
- Windows:
 1. Erstes Release der 64-Bit Version
 2. Anbindung an Log-Manager
- Erste Release-Version für Mac OS X, ab V10.7.3

V4.00.07

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Neue Methode in TagHandler-Klasse für EPC Class1 Gen2: Lock mit vereinfachter Parameterfolge

V4.00.02

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Prüfung auf doppelte UIDs in der Methode FEDM_ISCReaderModule::TagInventory
- TagHandler für EPC Class1 Gen2: neue Methode ReadCompleteBank
- Fehlerkorrektur in der Methode FEDM_ISCReaderModule::ReadCompleteConfiguration für ID ISC.LR2500-A und ID ISC.LRU3000
- Nur Windows: Abhängigkeit von MFC- und CRT-Bibliotheken nach MS11-025¹

V4.00.00

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für den neuen Leser: **ID ISC.LR2500-A**
- Unterstützung für UIDs bis 96 Bytes
- Unterstützung für UHF-Konfiguration UID = EPC + TID
- Die Organisation der Leserkonfiguration oberhalb CFG63 ist für den **ID ISC.LRU3000** mit der Firmwareversion V2.0.0 verändert worden und nicht mehr kompatibel zur Vorversion. Die Klassenbibliothek FEDM integriert ab dieser Version die notwendigen Anpassungen und ist damit inkompatibel für Firmwareversionen < V2.0.0. Innerhalb der Reader-Klassen findet keine Überprüfung bzgl. der Inkompatibilität statt. Diese muss Applikations-seitig erfolgen.

Die Tabelle gibt Klarheit bzgl. der Kompatibilitäten:

LRU3000-Firmware	SDK-Version	ISOStart-Version	XML-Konfigurationsdatei
< 2.00.00	<= 3.03.01	<= 8.03.02	muss mit ISOStart <= 8.03.02 erstellt sein
>= 2.00.00	>= 4.00.00	>= 9.00.00	muss mit ISOStart >= 9.00.00 erstellt sein

- Parallele Nutzung einer TCP/IP-Verbindung von mehreren Leserobjekten ist nicht mehr möglich. Der neue Fehlercode -157 wird zurückgegeben, wenn ConnectTCP mit der gleichen IP-Adresse und Port ein zweites Mal, aber von einem anderen Reader-Modul aufgerufen wird
- Disconnect der Leserklasse **FEDM_ISCReaderModule** kann einen positiven Rückgabewert haben, wenn im Fall einer TCP/IP-Verbindung der Verbindungsabbau nicht erfolgreich war. Der positive Rückgabewert entspricht dem letzten Status der Verbindung. Es wird empfohlen, die Codezeilen genau zu untersuchen, die diese Methode aufrufen.
- Neue Methoden in der Leserklasse **FEDM_ISCReaderModule**: GetTcpConnectionState, GetNonAddressedTagHandler, Convert_EPC_C1_G2_TagHandler

¹ Microsoft Sicherheitsbulletin Artikel-IDs: 2538218, 2538243 und 2542054 vom 14. Juni 2011

- Unterstützung für [0x74] Input Event im Notification-Mode für die Leser **ID CPR50** und **ID MAX50**
- Neue TagHandler-Klassen für IDS SL13A (ISO 15693) und IDS SL900A (EPC Class1 Gen2)
- TagHandler-Klasse für EPC Class1 Gen2 mit neuen Methoden: GetProtocolControl, GetEpcOfUid, GetTidOfUid
- Erweiterung in der Struktur **struct _FEDM_TASK_INIT** um Keep-Alive Parameter für den Notification-Task. Bedingt dadurch muss entweder der neue Parameter *bKeepAlive* auf false oder besser, die gesamte Struktur mit 0 initialisiert werden (z.B. mit memset). Es wird empfohlen, die Codezeilen genau zu untersuchen, die diese Struktur erzeugen und initialisieren.
- FEDM V4.00.00 setzt zwingend folgende Versionsstände abhängiger DLLs/SOs voraus:

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

- Automatische Einstellung des richtigen OBID-Protokollrahmens: Bei Verwendung der Methoden FindBaudRate() für serielle Verbindungen mit anschließendem ReadReaderInfo(GetProtocolFrameSupport()) bzw. ReadReaderInfo() für USB- und TCP-Verbindungen ist sichergestellt, dass danach der korrekte OBID-Protokollrahmen (Standard oder Advanced) eingestellt ist. Dies ist deshalb von Bedeutung, weil zukünftige Leser den Standard-Protokollrahmen nicht mehr unterstützen werden und jede Kommunikation in ein Timeout laufen wird, wenn ein Standard-Protokollrahmen verwendet wird.

V3.03.01

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für den neuen Leser: **ID ISC.MRU102**

V3.03.00

- Die APIs einiger DLLs wurden modifiziert. Deshalb müssen Applikationen neu kompiliert werden.
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für die neuen Leser: **ID ISC.LR2500-B**, **ID ISC.MR102**, **ID CPR30.xx** und **ID ISC.CPR52.xx**

V3.02.00

- Die APIs einiger DLLs wurden modifiziert. Deshalb müssen Applikationen neu kompiliert werden.
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Neue Klasse zur Unterstützung der externen Funktionseinheit **People-Counter**

V3.01.00

- Die APIs einiger DLLs wurden modifiziert. Deshalb müssen Applikationen neu kompiliert werden.
- Release der TagHandler-Klassen für vereinfachte Kommunikation mit verschiedenen Transpondertypen, insbesondere mit ISO14443 und ISO15693 konformen Transpondern
- Verschlüsselte Kommunikation für **ID CPR50**, **ID MAX50** und **ID ISC.LRU3000** optional möglich
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

V3.00.14

- Unterstützung für neuen Leser: **ID MAX50.xx**
- Unterstützung für neuen Leser: **ID ISC.LRU3000**
- 2. Beta-Release der TagHandler-Klassen für vereinfachte Kommunikation mit verschiedenen Transpondertypen, insbesondere mit ISO14443 und ISO15693 konforme Transponder
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Neue Hilfsklassen **FedmlscReport_ReaderInfo** und **FedmlscReport_ReaderDiagnostic** erzeugen aus den Informationsdaten bzw. den Zustandsdaten einen Report (s. ISOStart ab V8.01.02)

V3.00.07

- Unterstützung für neuen Leser: **ID CPR50.xx**
- Beta-Release der TagHandler-Klassen für vereinfachte Kommunikation mit verschiedenen Transpondertypen
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Die Methode `SendProtocol(0x72)` verwendet intern modifizierte Definitionen der Konstanten `FEDM_ISC_TMP_0x72_OUT_TYPE_1...FEDM_ISC_TMP_0x72_OUT_TYPE_8`: Bisher adressierten diese Konstanten ein Bit. Jetzt werden 3 Bits adressiert. Daraus resultiert, dass der OUT-TYPE Relay jetzt mit 0x04 und nicht mehr mit 0x01 angegeben werden muss. Siehe auch [6.4.3. Beispiele für die Verwendung der Methode SendProtocol](#). Diese Änderung ist für alle Lesertypen anzuwenden, die das Protokoll [0x72] Set Output unterstützen.

V3.00.02

- Neue Methode in **FEDM_ISCReaderModule**
 - ReadReaderDiagnostic
- Verbesserungen in Methoden von **FEDM_ISCReaderModule**
 - ApplyConfiguration
 - ReadCompleteConfiguration
 - WriteCompleteConfiguration
- Neue Struktur **FEDM_ISC_READER_DIAGNOSTIC**
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

V3.00.00

- Grundlegende Änderungen sind im Teil A (H10102-xd-ID-B) dokumentiert
- Unterstützung für neue Leser: **ID ISC.MRU200**, **ID ISC.PRHD102**, **ID CPR40.xx**
- Folgende ältere Leser werden nicht mehr unterstützt: ID ISC.M01 und ID ISC.LR100
- Unterstützung für UHF-Multiplexer **ID ISC.ANT.UMUX**
- Unterstützung für Transpondertyp EPC Class1 Gen2 HF
- Neue Methoden in **FEDM_ISCReaderModule**
 - ApplyConfiguration
 - ReadCompleteConfiguration
 - WriteCompleteConfiguration
 - ResetCompleteConfiguration
- Neue Methode EvalLibDependencies in Klasse **FEDM_ISCReader** zum Detektieren von Versionskonflikten mit abhängigen Bibliotheksdateien
- Struktur **FEDM_ISC_READER_INFO** wurde ausgebaut
- Zur Verbesserung der Übersichtlichkeit der Zugriffskonstanten für Leserkonfigurationen wurden diese in Namespaces strukturiert
- Neue überladene Methoden Get/SetConfigPara in der Klasse **FEDM_ISCReader** zur Modifikation der in Namespaces organisierten Konfigurationsparameter
- Das Schreiben der Leserkonfiguration ist nur noch für Konfigurationsblöcke möglich, die zuvor gelesen wurden, mit der Ausnahme, dass nach der Übernahme von Konfigurationsparametern aus einer XML-Datei das Schreiben in den Leser immer möglich ist.

V2.06.00

- Kleine Anpassungen für den Leser ID ISC.LRU2000
- Neue Funktion in der Klasse **FEDM_ISCReaderModule**: ReadReaderInfo
- Neue Struktur **FEDM_ISC_READER_INFO**

V2.05.06

- Die Linux-Version wurde mit dem Compiler GCC 3.3.3 unter SuSE Linux 9.1 erstellt
- Der Quellcode ist geeignet zur Kompilierung mit Visual Studio 2005
- Support für das Protokoll [0x6B] Centralized RF Synchronization

V2.05.01

- Neue Funktion **TriggerAsyncTask** in der Leserklasse **FEDM_ISCReader**
- Modifizierte Lizenzbestimmungen

V2.04.00

- Neue allgemeine Konstanten für den UHF-Leser LRU1000:

Bezeichner	Kommentar
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK_LGT	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp 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	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp 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	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_LGT	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp 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

- Unterstützung für den neuen HF-Leser ID ISC.LR2000
- Erweiterungen für den UHF-Leser ID ISC.LRU1000 in der Konfiguration
- Unterstützung für neue Transponder-Typen: HF-Transponder Innovision Jewel und UHF-Transponder EPC Class0/0+
- Unterstützung neuer Optionen im Advanced Buffered Read Mode:
 1. Input und Status im Trigger-Mode können übertragen werden

- 2. Erweiterung von TR-DATA in der Konfiguration und im Antwortprotokoll von [0x22] Read Buffer
- Erweiterungen für ISO 14443A Transponder:
 1. [0xB2][0x30] Mifare Value Commands
 2. [0xB0][0x25] Select unterstützt Card Information
- Neue globale Funktion **FEDM_ConvHexUCharToTwoAscii**
- Unterstützung für neues Protokoll: [0x72] Set Output:
- Neue allgemeine Konstanten:

Bezeichner	Kommentar
FEDM_ISC_TMP_B0_MODE_CINF	Flag Card Information im Mode-Byte für [0xB0][0x25] Select
FEDM_ISC_TMP_B0_MODE_WR_NE	Flag Write-Erase im Mode-Byte für [0xB0][0x24] Write Multiple Blocks
FEDM_ISC_TMP_B0_RSP_FORMAT	Format Byte im Antwortprotokoll von [0xB0][0x25] Select, wenn das CINF-Flag gesetzt ist
FEDM_ISC_TMP_B2_REQ_MF_CMD	Parameter für [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 von TR-DATA im Antwortprotokoll von [0x22] Read Bufer
FEDM_ISC_TMP_ADV_BRM_TRDATA2_...	Flags im 2. Byte von TR-DATA im Antwortprotokoll von [0x22] Read Bufer
FEDM_ISC_TMP_0x72_OUT...	Konstanten für [0x72] Set Output

- Geänderte allgemeine Konstanten:

Alter Bezeichner	Neuer Bezeichner
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

- Erweiterungen für ISO 14443A Transponder.
- Modifikation in der Zugriffskonstanten FEDM_ISC_TMP_B0_REQ_UID. Nähere Einzelheiten in [6.5.1. Besonderheiten des addressed mode](#)¹
- Neue Optionen für EPCglobal Class1 Gen2 Transponder (z.B. [0xB3] EPC Command)
- Unterstützung von Protokollen mit erweiterten Optionen (variable UID-Länge, Bank-Nummer, 2 Byte Block-Adresse, Access Passwort)

¹ Applikationen, die die FEDM-Bibliothek dynamisch laden, müssen neu kompiliert werden.

- Erweiterung der Buffered Read Mode Tabellenklasse FEDM_BRMTabItem für EPC-Transponder
- Änderung der Blockgröße des Byte-Arrays TempData für temporäre Daten von 16 auf 32.¹
- Die Tabellenkonstanten, beginnend mit FEDM_ISC_DATA_... sind neu durchnummeriert worden.¹
- Einige Zugriffskonstanten adressieren einen anderen Speicherbereich.¹
- Neue Präprozessor-Definition **_FEDM_XML_SUPPORT** zur Einbindung der XML-Serialisierungsklassen. Diese Option war in früheren Versionen nicht notwendig. Vor der Neukompilation mit der aktuellen Version muss also die Definition gesetzt werden, wenn die XML-Serialisierungsklassen genutzt werden.
- Neue Präprozessor-Definition **_FEDM_MFC_SUPPORT** zur Einbindung der MFC-Klassen CString und CArchive. Diese Option war in früheren Versionen nicht notwendig. Vor der Neukompilation mit der aktuellen Version muss also die Definition gesetzt werden, wenn die genannten MFC-Klassen genutzt werden.
- Etwas verändertes Verhalten der Funktion ResetTable.
- Neue überladene Funktion SetTableSize. Mit dieser Funktion kann die Größe einiger Datenfelder in einem Tabelleneintrag abweichend von der Voreinstellung kleiner dimensioniert werden.
- Neue allgemeine Konstanten:

Bezeichner	Kommentar
FEDM_ISC_TMP_B0_MODE_EXT_ADR	Flag im Mode Byte für extended address mode im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_MODE_UID_LF	Flag im Mode Byte für UID Längenangabe im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_REQ_BANK	Bank im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_REQ_BANK_ACCESS_FLAG	Access Flag in Bank im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_REQ_BANK_BANK_NR	Banknummer in Bank im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_REQ_DB_ADR_EXT	2 Byte Adresse im [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B1_REQ_TI_PASSWORD	Passwort für [0xB1] Cust and Proprietary Commands für Transponder von Texas Instruments
FEDM_ISC_TMP_B3_ISO_ERROR	ISO Error Code nach [0xB3] Command
FEDM_ISC_TMP_B0_ACCESS_PW_LENGTH	Länge des Passworts (in Byte) für [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B0_ACCESS_PW	Passwort für [0xB0] Sendeprotokoll
FEDM_ISC_TMP_B3_CMD	Subcommand für [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_MODE	Mode Byte für [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_MODE_EXT_ADR	Flag im Mode Byte für extended address mode im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_MODE_EPC_LF	Flag im Mode Byte für EPC Längenangabe im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_MODE_ADR	Adressierungsart im Mode Byte im [0xB3] Sendeprotokoll

Bezeichner	Kommentar
FEDM_ISC_TMP_B3_REQ_EPC_LEN	EPC Längenangabe im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_BANK	Bank im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_BANK_ACCESS_FLAG	Access Flag in Bank im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_BANK_BANK_NR	Banknummer in Bank im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_DB_ADR_EXT	2 Byte Adresse im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_DB_ADR	1 Byte Adresse im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_DBN	Anzahl Datenblöcke im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_DB_SIZE	Blockgröße im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_TR_TYPE	Transpondertyp im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_REQ_EPC	EPC im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_KILL_PW_LENGTH	Länge des Kill Passworts im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_KILL_PW	Kill Passwort im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_LOCK_DATA_LENGTH	Länge der Lock Daten im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_LOCK_DATA	Lock Daten im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH	Länge des Passworts im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_ACCESS_PW	Passwort im [0xB3] Sendeprotokoll
FEDM_ISC_TMP_B3_RSP_DB_ADR_E	Datenblockadresse bei ISO-Fehler im [0xB3] Empfangsprotokoll

- Geänderte allgemeine Konstanten:

Alter Bezeichner	Neuer Bezeichner
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

- Auslagerung der Tabellenklassen FEDM_ISOTabItem und FEDM_BRMTabItem in eigene Dateien.
- Der Buffered Read Mode für den Leser ID ISC.LR200 unterstützt die Transponder EPC und I-CODE UID.
- Neue Konstanten für Konfigurationsparameter für HF-Leser ID ISC.M02, ID ISC.MR/PR/PRH100 und ID ISC.MR101 in FEDM_ISCReaderID.h.

Bezeichner	Kommentar
FEDM_ISC_EE_SCAN_END_USER1	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_END_USER2	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_END_USER3	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER1	Parameter für Scan-Mode

Bezeichner	Kommentar
FEDM_ISC_EE_SCAN_HDR_USER2	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER3	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_HDR_USER4	Parameter für Scan-Mode
FEDM_ISC_EE_SCAN_LEN_USER	Parameter für Scan-Mode

V2.01.00

- Unterstützung für neuen Leser ID ISC.MR101 (serielle und USB-Version)
- Unterstützung für externe Funktionseinheiten
- Neue Konstanten für UHF-Leser in FEDM_ISCReaderID_LRU1000.h.
- Neue allgemeine Konstanten:

Bezeichner	Kommentar
FEDM_ISC_TMP_DIAG_DATA	Diese Konstante gilt für alle Diagnose-Modi und ersetzt die u. a. entfernten Konstanten für die Modi 0x01, 0x02, 0x03.

- Geänderte allgemeine Konstanten:

Alter Bezeichner	Neuer Bezeichner
FEDM_ISCLR_TMP_DIAG_MODE	FEDM_ISC_TMP_DIAG_MODE

- Entfernte allgemeine Konstanten:

Bezeichner	Kommentar
FEDM_ISCLR_TMP_DIAG_0x01_DATA	Durch FEDM_ISC_TMP_DIAG_DATA ersetzt
FEDM_ISCLR_TMP_DIAG_0x02_DATA	Durch FEDM_ISC_TMP_DIAG_DATA ersetzt
FEDM_ISCLR_TMP_DIAG_0x03_DATA	Durch FEDM_ISC_TMP_DIAG_DATA ersetzt
FEDM_ISC_TMP_EPC_DESTROY_LEN	Entfällt, weil die Länge der EPC/UID durch den Destroy-Mode und den Header der EPC intern berechnet wird.

- Kleinere Fehlerkorrekturen

V2.00.00

- Unterstützung für die UHF-Transponder ISO18000-6-A, EM4222, EPC Gen 2, EPC Class 1
- Unterstützung für gelesene Datenblöcke im Advanced Buffered Read Mode.
- Änderung für den Buffered Read Mode: die Blockanzahl DBN wird intern jetzt als **unsigned int** verwaltet. Deshalb kann man den Tabellenwert mittels GetTableData(idx, FEDM_BRM_TABLE, FEDM_ISC_DATA_DBN, ...) nicht mehr als **unsigned char** ermitteln. Eventuell müssen Sie Ihr Programm anpassen.
- Einführung folgender neuer Konstanten in FEDM_ISCReaderID_LRU1000.h:

Neu	Kommentar
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_A	Flag für Tag-Treiber ISO18000-6-A in der Leser-Konfiguration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_D	Flag für Tag-Treiber EM4222 in der Leser-Konfiguration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_E	Flag für Tag-Treiber EPC Gen 2 in der Leser-Konfiguration
FEDM_ISC_LRU1000_EE_RF_TAG_DRV_J	Flag für Tag-Treiber EPC Class 1 in der Leser-Konfiguration
FEDM_ISC_LRU1000_EE_PER_MODE	Persistence Modus in der Leser-Konfiguration Block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT1	Reset-Zeit für Antenne 1 in der Leser-Konfiguration Block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT2	Reset-Zeit für Antenne 2 in der Leser-Konfiguration Block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT3	Reset-Zeit für Antenne 3 in der Leser-Konfiguration Block CFG16
FEDM_ISC_LRU1000_EE_PER_RESET_TIME_ANT4	Reset-Zeit für Antenne 4 in der Leser-Konfiguration Block CFG16

V1.09.11

- Unterstützung für den HF-Transponder I-Code UID
- Entfernung aller Zugriffskonstanten, die den RAMDATA_MEM adressiert haben. Dies reduziert die Anzahl der Zugriffskonstanten erheblich und sorgt für mehr Übersicht. Der Zugriff auf Daten im RAMDATA_MEM kann statt dessen mit der Funktion FEDM_MdfyMemID und der Zugriffskonstanten für Daten im EEDATA_MEM erfolgen.
- Einführung folgender neuer Konstanten in FEDM_ISCReaderID.h:

Neu	Kommentar
FEDM_ISC_EE_RF_TAG_DRV_H	Flag für Tag-Treiber I-Code UID in der Leser-Konfiguration

V1.09.10

- Unterstützung für den UHF-Leser ID ISC.LRU1000 (Konfigurationsparameter in FEDM_ISCReaderID_LRU1000.h)
- Unterstützung für den HF-Leser ID ISC.MR200 (Konfigurationsparameter in FEDM_ISCReaderID_MR200.h)
- Unterstützung für den Advanced Protocol Frame mit zwei Längenbytes
- Erweiterungen in der BRM-Tabelle zur Unterstützung des neuen Protokolls [0x22] Read Buffer
- Aktualisierungen für den ID ISC.LR200 bzgl. neuer Konfigurationsparameter.
- Alle Konfigurationsparameter für den ID ISC.LR200 wurden in die Datei FEDM_ISCReaderID_LR200.h ausgelagert.
- Neue Protokolle: [0x18] Destroy EPC, [0x22] Read Buffer, [0x64] System Reset, [0x66] Firmware Version, [0x87] Set System Date, [0x88] Get System Date

- Einführung folgender neuer Konstanten in FEDM_ISCReader.h:

Neu	Kommentar
FEDM_ISC_DATA_ANT_NR	Antennennummer in Tabelle
FEDM_ISC_DATA_SNR_LEN	Länge der Seriennummer in der Tabelle
FEDM_ISC_DATA_DATE	Datumsfeld in der Tabelle

- Einführung folgender neuer Konstanten in FEDM_ISCReaderID.h:

Neu	Kommentar
FEDM_ISC_TMP_SOFTVER_RX_BUF	Informationsfeld Länge des Empfangspuffers in [0x65] Software Version
FEDM_ISC_TMP_SOFTVER_TX_BUF	Informationsfeld Länge des Sendepuffers in [0x65] Software Version
FEDM_ISCLR_TMP_BRM_TRDATA_ANT	Antennennummer in Protokolldaten
FEDM_ISC_TMP_EPC_DESTROY_MODE	Mode-Parameter für Protokoll [0x18] Destroy EPC
FEDM_ISC_TMP_EPC_DESTROY_LEN	Längen-Parameter für Protokoll [0x18] Destroy EPC
FEDM_ISC_TMP_EPC_DESTROY_PASSWORD	Passwort für Protokoll [0x18] Destroy EPC
FEDM_ISC_TMP_DESTROY_EPC	EPC für Protokoll [0x18] Destroy EPC Achtung: die EPC geht auf Grund ihrer Länge über die Blockgrenze hinaus. Zur richtigen Anwendung dieser Konstanten s. 6.4.3. Beispiele für die Verwendung der Methode SendProtocol
FEDM_ISC_TMP_SYSTEM_RESET_MODE	Mode Byte für [0x64] System Reset
FEDM_ISC_TMP_ADV_BRM_SETS	Übergabeparameter für [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	Übergabeparameter für [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	Übergabeparameter für [0x87] Set System Date und [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	

- Entfernte Konstanten in FEDM_ISCReaderID.h:

Neu	Kommentar
FEDM_ISCLR_EE_SYSG_SYS_MODE_BRM	Konfigurationsparameter ist durch FEDM_ISCLR_EE_SYSG_SYS_MODE_OP_MODE ersetzt
FEDM_ISCLR_EE_SYSG_SYS_MODE_SCAN	Konfigurationsparameter ist durch FEDM_ISCLR_EE_SYSG_SYS_MODE_OP_MODE ersetzt

V1.08.00

- Serialisierung der Leser-Konfiguration im XML-Format mit Hilfe der neuen Klassen FEDM_XMLBase und FEDM_XMLReaderCfgDataModul. Kenntnisse zu den Klassen sind nicht notwendig.
- Erweiterungen in der Klasse FEDM_ISOTabItem für EPC (Electronic Product Code)
- Unterstützung des neuen Lesers ID ISC.M02
- Durchgehende Unterstützung der Datentypen bool, __int64 und STL-string durch neue überladene Funktionen von GetTableData, SetTableData und FindTableIndex.
- Einführung folgender neuer Konstanten in FEDM_ISCReader.h:

Neu	Kommentar
FEDM_ISC_DATA_SAK	Select Acknowledge (nur für ISO14443A Transponder)
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	Feld EPC Header; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_DOMAIN	Feld EPC-DomainManager; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_OBJECT	Feld EPC-ObjectClass; (EPC = Electronic Product Code)
FEDM_ISC_DATA_EPC_SNR	Feld EPC-Seriennummer; (EPC = Electronic Product Code)
FEDM_ISC_TYPE_...	Lesertyp entsprechend dem Feld SW-TYPE des Protokolls [0x65] Software Version
FEDM_ISC_NAME_...	Leserbezeichnung entsprechend dem Systemhandbuch
FEDM_ISC_NAME_..._UC	Leserbezeichnung in Unicode entsprechend dem Systemhandbuch
FEDM_ISC_TR_TYPE_...	Kennung des Transpondertyps entsprechend dem Anhang im Systemhandbuch
FEDM_ISC_EPC_TYPE_...	Kennung der EPC-Typen; (EPC = Electronic Product Code)

V1.07.00

- Unterstützung für die Protokolle: [0xB1] und [0xB2]

- Einführung folgender neuer Konstanten in FEDM_ISCReaderID:

Neu	Kommentar
FEDM_CPR_EE_BLOCK2 und alle darin enthaltenen Konfigurationskonstanten für das EEPROM des Lesers	Konfigurationskonstanten für den Leser ID CPR.02
FEDM_CPR_RAM_BLOCK2 und alle darin enthaltenen Konfigurationskonstanten für das RAM des Lesers	
FEDM_ISC_TMP_B1_CMD	spezifiziert das 0xB1-Kommando
FEDM_ISC_TMP_B1_MODE	spezifiziert Modus für 0xB1-Kommando
FEDM_ISC_TMP_B1_MODE_ADR	spezifiziert den Adressierungsmodus für 0xB1-Kommando
FEDM_ISC_TMP_B1_MFR	Hersteller-Code für 0xB1-Kommando
FEDM_ISC_TMP_B1_REQ_UID	UID im Anfrageprotokoll des 0xB1-Kommandos
FEDM_ISC_TMP_B1_ISO_ERROR	empfangener ISO-Fehlercode
FEDM_ISC_TMP_B2_CMD	spezifiziert das 0xB2-Kommando
FEDM_ISC_TMP_B2_MODE	spezifiziert Modus für 0xB2-Kommando
FEDM_ISC_TMP_B2_MODE_ADR	spezifiziert den Adressierungsmodus für 0xB1-Kommando
FEDM_ISC_TMP_B2_REQ_UID	UID im Anfrageprotokoll des 0xB2-Kommandos
FEDM_ISC_TMP_B2_REQ_DB_ADR	Parameter für [0xB2] [0xB0] Authent Mifare Protokoll
FEDM_ISC_TMP_B2_REQ_KEY_ADR	
FEDM_ISC_TMP_B2_REQ_KEY_TYPE	
FEDM_ISC_TMP_B2_REQ_KEY_ADR_TAG	Parameter für [0xB2] [0xB1] Authent my-d Protokoll
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	empfangener ISO-Fehlercode

V1.06.00

- Umbenennung folgender Konstanten in FEDM_ISCReaderID:

Alt	Neu
FEDM_ISCPRH_EE_SCAN_DBN	FEDM_ISCPRH_EE_SCAN_D_LGT

- Einführung folgender neuer Konstanten in FEDM_ISCReaderID:

Neu	Kommentar
FEDM_ISC_EE_BLOCK0 .. 7 und alle darin enthaltenen Konfigurationskonstanten für das EEPROM des Lesers	gemeinsame Konfigurationskonstanten für die Leser 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 und alle darin enthaltenen Konfigurationskonstanten für das RAM des Lesers	

- Fehlerkorrekturen für Selected Mode der ISO-Host Commands
- Unterstützung für GNU C-Compiler unter Linux.

V1.05.00

- Die Puffergröße für Datenblöcke im Buffered-Read-Mode wurde von 128 auf 1024 Byte angehoben.

V1.04.00

- Unterstützung für alle Lesertypen der *i-scan*-Familie
- Die Integration der ISO-Host-Commands für alle Leser der *i-scan*-Familie ist jetzt abgeschlossen. Unterstützt werden somit Transponder nach der Norm ISO 15693 mit unterschiedlicher Blockgröße. Addressed, Non-Addressed und Selected Modi sind möglich.
- Die interne Tabelle ISCTable ist in zwei Tabellen (m_ISOTable für Datenaustausch via ISO-Host-Commands und m_BRMTable für Daten vom Long-Range-Reader ISC.LRxxx im Buffered-Read-Mode) aufgeteilt worden. Dadurch entfällt die Funktion GetTableType.
- Mit der Erweiterung der Schnittstelle in der abstrakten Basisklasse mußten alle Tabellenfunktionen um den Übergabeparameter uiTableID (Tabellentyp-Konstante) erweitert werden.
- Einige Namensänderungen von und Erweiterungen bei den Zugriffskonstanten.
- Die Unterstützung für den Multi-Job-Poll ist entfernt worden.
- Die Protokolle [0x11], [0x14], [0x15], [0x16] und [0x17] für den ID ISC.M01 werden nicht mehr unterstützt.
- Die Datencontainer SN_Mem, PubMem und ConfMem werden nicht mehr verwendet (betrifft nur den Leser ISC.M01)

V1.03.00

- Mit der Erweiterung der Schnittstelle in der abstrakten Basisklasse mußten alle Tabellenfunktionen um den Übergabeparameter uiTableID (Tabellentyp-Konstante) erweitert werden.
- Einige Namensänderungen von und Erweiterungen bei den Zugriffskonstanten.

V1.00.00

- erste Release-Version