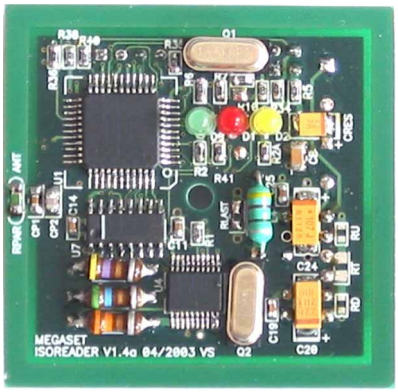
Dokumentation ISO-Reader



HW-Version: V1.4a
SW-Version: V1.4

Hinweis:
Die Angaben in unseren Informationsdateien und Datenblättern über Produkte und deren technische Daten sind sorgfältig und gewissenhaft zusammengestellt und beruhen auf dem heutigen Stand unserer Kenntnisse über die beschriebenen Produkte. Eine rechtlich verbindliche Zusicherung bestimmter Eigenschaften oder der Eignung für einen bestimmten Einsatzzweck kann nicht abgeleitet werden. Bitte nehmen Sie die Beratung unseres Vertriebs über konkrete Verwendungsmöglichkeiten in Anspruch. Es ist jedoch in jedem Fall unerlässlich, unsere Angaben und Empfehlungen vor ihrer konkreten Anwendung für den eigenen Bereich selbstverantwortlich zu prüfen. Wir können generell keine Haftung für Schäden, die sich aus der Anwendung der Produkte ergeben, übernehmen. MEGASET Systemtechnik übernimmt keinerlei Gewähr für die Aktualität, Korrektheit und Vollständigkeit der bereitgestellten Informationen. Aus Gründen der ständigen technischen Weiterentwicklung behalten wir uns technische Änderungen und Verbesserungen der Produkte jederzeit vor.

MEGASET Systemtechnik
GB der MEGATRON Elektronik AG & Co.
Hermann-Oberth-Straße 7
D-85640 Putzbrunn / München

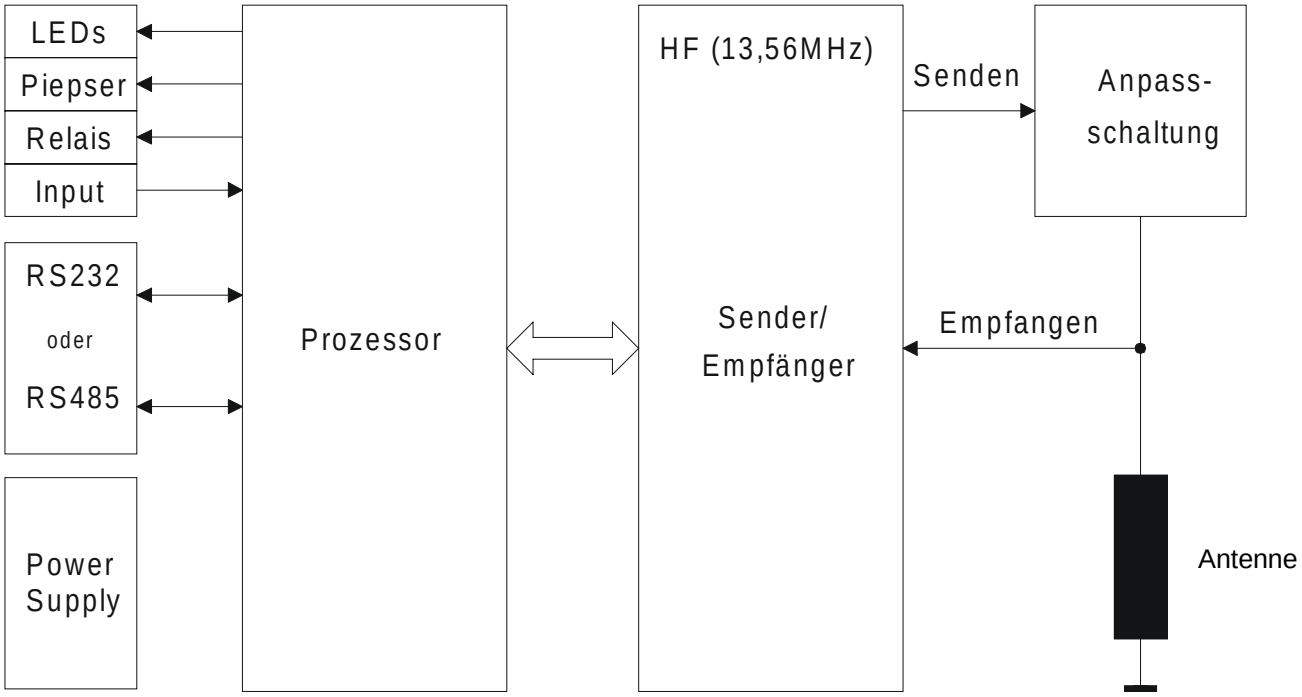
Tel.: +49 (0) 89 / 4 60 94 – 4 10
Fax: +49 (0) 89 / 4 60 94 – 4 11
Email: adic@megaset.com
Internet: www.megaset.com

Inhaltsverzeichnis:

1. Einführung:

Allgemeines:

Der ISO-Reader ist ein RFID-Controller zum Lesen und Schreiben von ISO-Transpondern nach ISO/IEC 15693 mit 13,56MHz.



Unterstützte Tags:

Tag-it HF-I	Texas Instruments
I-Code SLI	Philips

Abmessungen der Platine:

- Quadratisch 50 x 50mm

Peripherie:

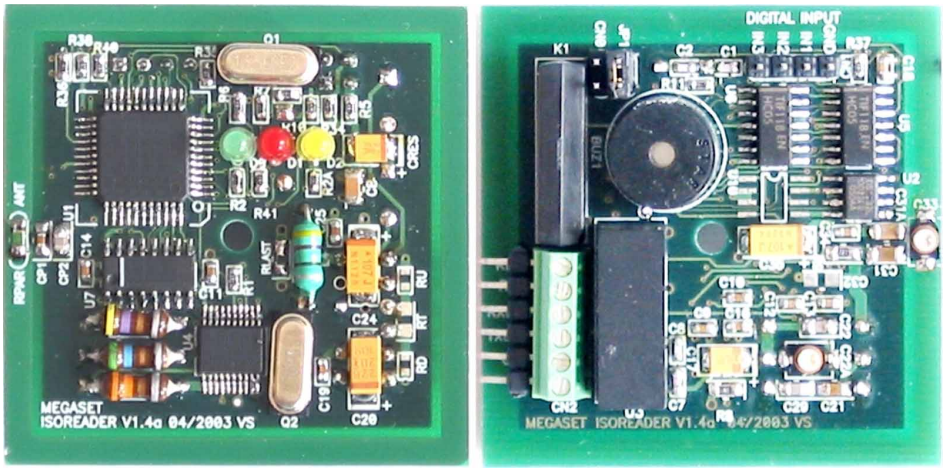
- integrierte Lese-Antenne auf der Platine
- Spannungsversorgung +5Volt oder +12V über Klemmleiste
- Gelbe LED1 zur Anzeige der Betriebsbereitschaft
- Rote LED2 zur Anzeige im Fehlerfall (nur optional bei Slave-Systemen)
- Grüne LED3 zur Anzeige der Transponder Lese-/Schreibaktionen (Send/Receive)
- Relais mit integrierter Freilaufdiode für induktive Lasten
  (1 Schließer, Kontaktstrom 1A) über Klemmleiste (nur optional bei Slave-Systemen)
- Akustischer Signalgeber (nur optional bei Slave-Systemen)
- 3 gepufferte digitale Eingänge zum Prozessor

Schnittstellen:

- 1 serielle Schnittstelle (Je nach Bestückungsvariante: RS232C oder RS485)
- Zwei Baudraten mit 9600 bps oder 19200 bps,8,N,1
  (8 Datenbit, keine Parität, 1 Stoppbit, kein Handshake)
- Bis zu 32 Halbduplex-Devices am RS-485-Bus anschließbar

Software:

- Demoprogramme für DOS und Microsoft-Windows (für Win98SE, Win2000/XP) mit der die Funktionen des ISO-Readers ausgetestet werden können.
- Programmierbeispiele mit Sourcecode für DOS und Windows



2. Steckerbelegung :

CN1 - Digitale Eingänge zum Mikroprozessor:

Pin:	Signal:	Funktion:
1	GND	Signal-Masse
2	IN1	Digitaler Eingang
3	IN2	Digitaler Eingang
4	IN3	Digitaler Eingang

CN2 - Klemmleiste:

Pin:	Signal:	Funktion:
1	VCC	Zuführung der Versorgungsspannung (+5V/+12V)
2	GND	Zuführung der Versorgungsspannung (GND)
3	TXD/RS485_A	Sendedaten für RS232; Sende-/Empfangsdaten für RS485 Kanal A
4	RXD/RS485_B	Empfangsdaten für RS232; Sende-/Empfangsdaten für RS485 Kanal B
5	REL1	Schaltkontakt des Relais
6	REL2	Schaltkontakt des Relais

3. Schnittstellen:

3.1 RS232C-Interface an CN2:

Pin:	Signal:	Richtung:	Funktion:
2	GND	-	Signal-Masse
3	TXD	OUT	Sendedaten vom Controller
4	RXD	IN	Empfangsdaten für den Controller

Signal Polarität:	Logik: "1" (HIGH)	MARK:	-3V	bis	-25V
	Logik: "0" (LOW)	SPACE:	+3V	bis	+25V

Das RS232C-Interface arbeitet ohne Handshake.

3.2 Das RS485-Interface:

Pin:	Signal:	Richtung:	Funktion:
2	GND	-	Signal-Masse
3	TXD/RXD_A	BI	Sende- und Empfangsdaten (A)
4	TXD/RXD_B	BI	Sende- und Empfangsdaten (B)

Signal-Polarität:	RS485	MARK:	Vin	>	Vin-
		SPACE:	Vin	<	Vin-

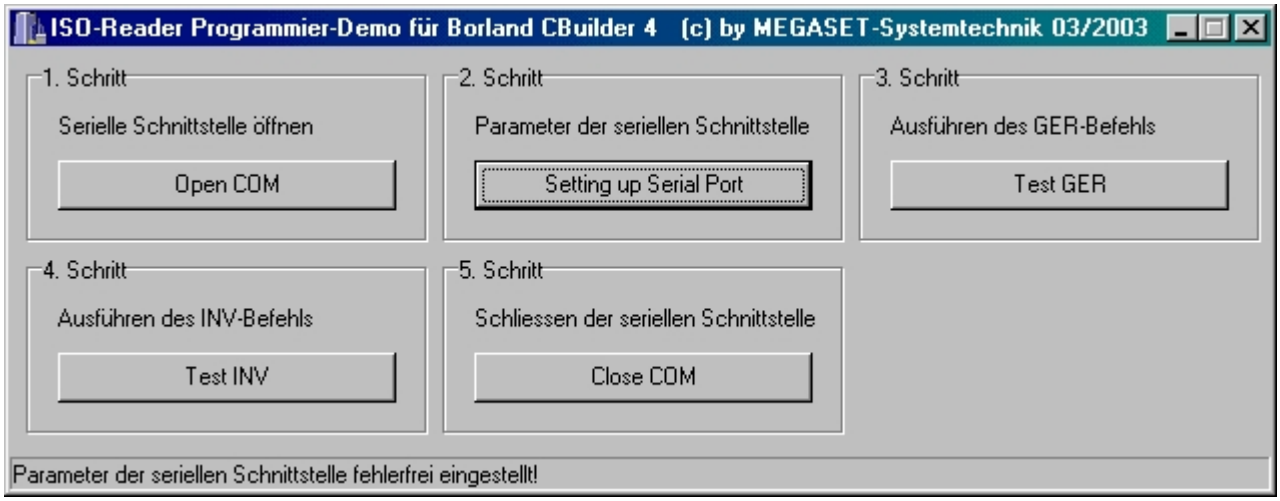
Das RS485-Interface arbeitet immer im Halb-Duplex-Verfahren. Das bedeutet, dass immer nur eine Stelle Daten senden kann während die Gegenstelle die Daten empfängt.
Der Controller arbeitet standardmäßig als Slave - er ist also immer auf Empfangen eingestellt und gibt nur Daten nach Aufforderung aus.

Es können bis zu 32 Halb-Duplex-Devices am RS-485-Bus angeschlossen werden.
Der Adressbereich lässt sich beliebig zwischen 001 bis 999 wählen.
Die Adressen "000" und "999" sind reserviert und dürfen nicht benutzt werden.

4. Software:

4.1 Übersicht:

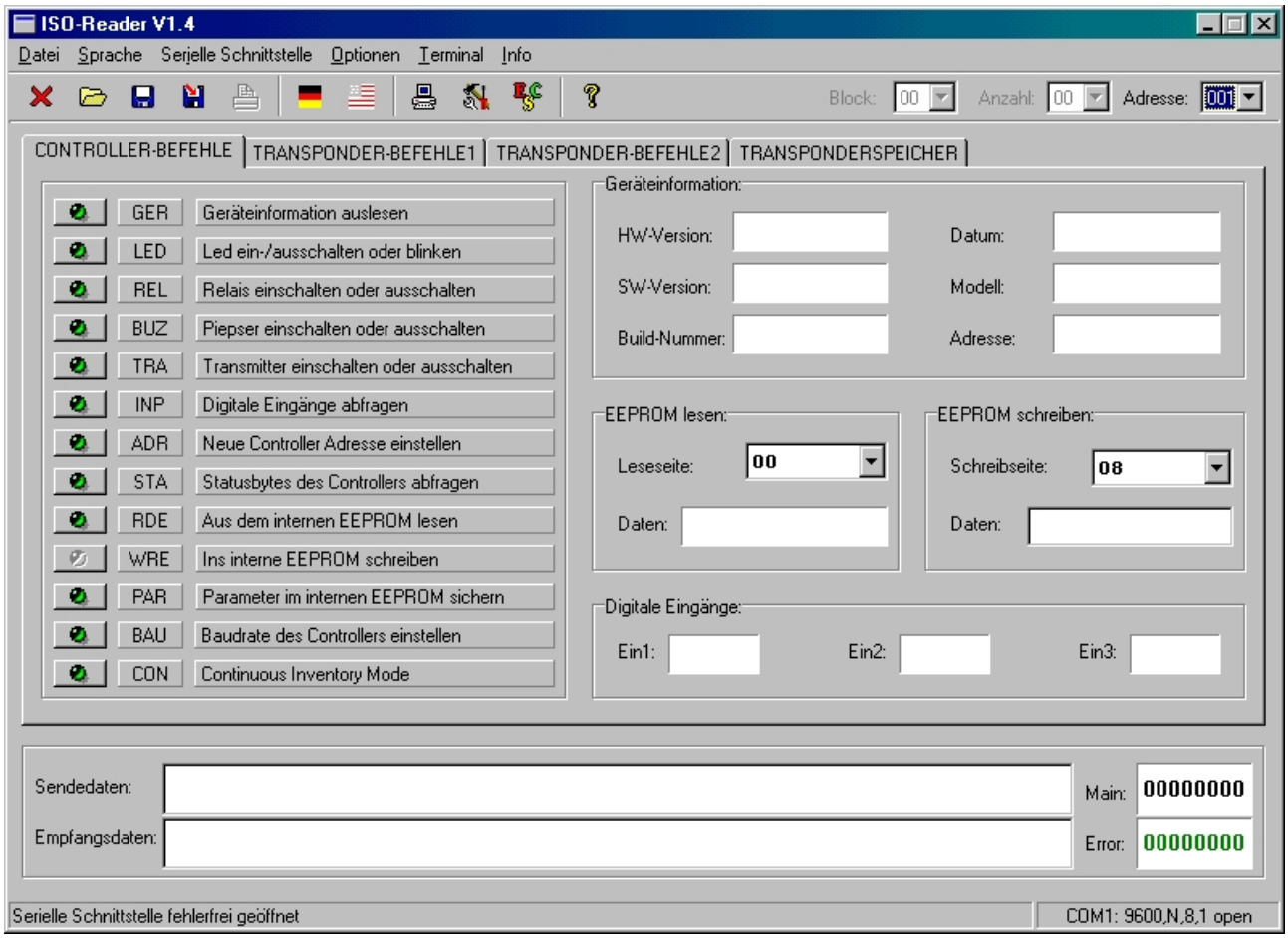
Zum ISO-Reader gibt es ein Testprogramm für die Microsoft-Windows Plattform das ein komfortables Ausprobieren aller Funktionen des ISO-Readers ermöglicht. Alternativ könnten auch alle Befehle mit einem Terminalprogramm (z.B. Hyperterminal) ausgeführt werden.
Ferner werden Programmierbeispiele (mit Quellcode) für DOS und Windows in den Programmiersprachen (Microsoft Visual Basic, Borland Delphi, Borland CBuilder und Borland C++) zur Verfügung gestellt um einen leichten Einstieg in die Programmierung des ISO-Readers zu ermöglichen.



Beachte: Die Software wurde unter Windows 98SE und Windows 2000/XP getestet.
Eine korrekte Funktionsweise unter anderen Betriebssystemversionen wie Windows 95, Windows Me oder Windows NT kann nicht garantiert werden.
Linux wird nicht unterstützt (Jedoch kann auch unter Linux mit einem beliebigen Terminalprogramm über die serielle Schnittstelle auf den ISO-Reader zugegriffen werden)

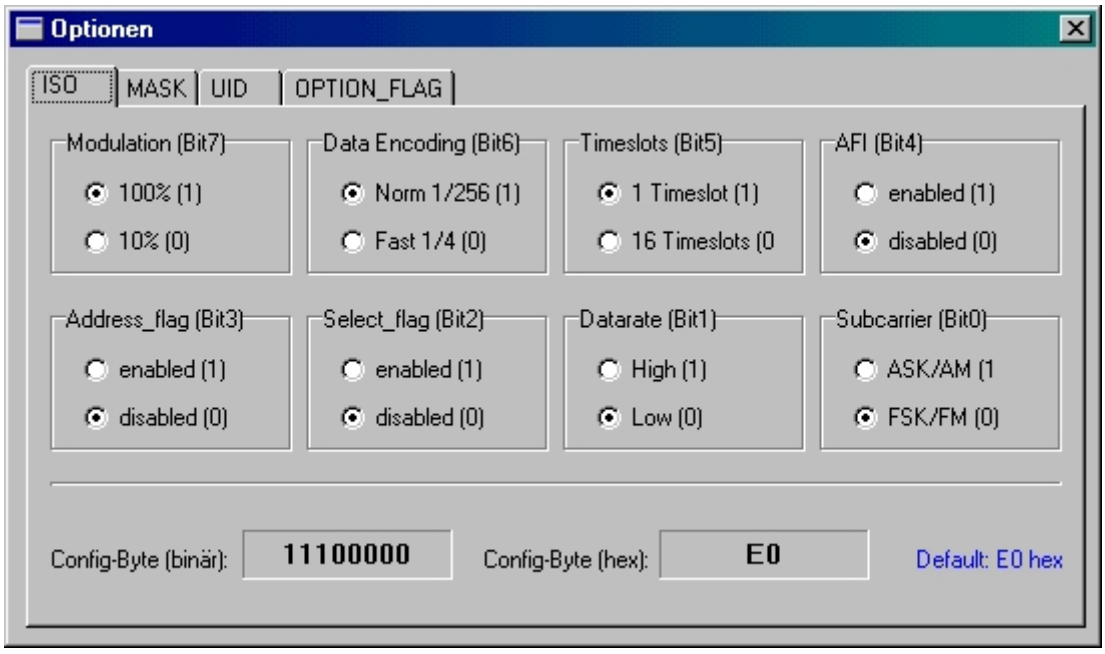
4.2 Beschreibung des Testprogramms für Windows:

Unter dem Menüpunkt *'Datei'* kann das Programm wieder beendet werden (oder man klickt auf das rote Kreuz in der Symbolleiste.
Die Sprache der Menüs, Beschriftungen und Textausgaben des Testprogramms können mit dem Befehl *'Sprache'* wahlweise in Deutsch oder Englisch dargestellt werden.
Der Transponderspeicher kann ausgelesen und abgespeichert werden. Die abgespeicherten Daten können wieder reingeladen und damit ein anderer Transponder programmiert werden.
Die Sende- und Empfangsdaten der seriellen Schnittstelle werden zur Kontrolle der Kommunikation des PC's mit dem ISO-Reader in ASCII und HEX angezeigt.

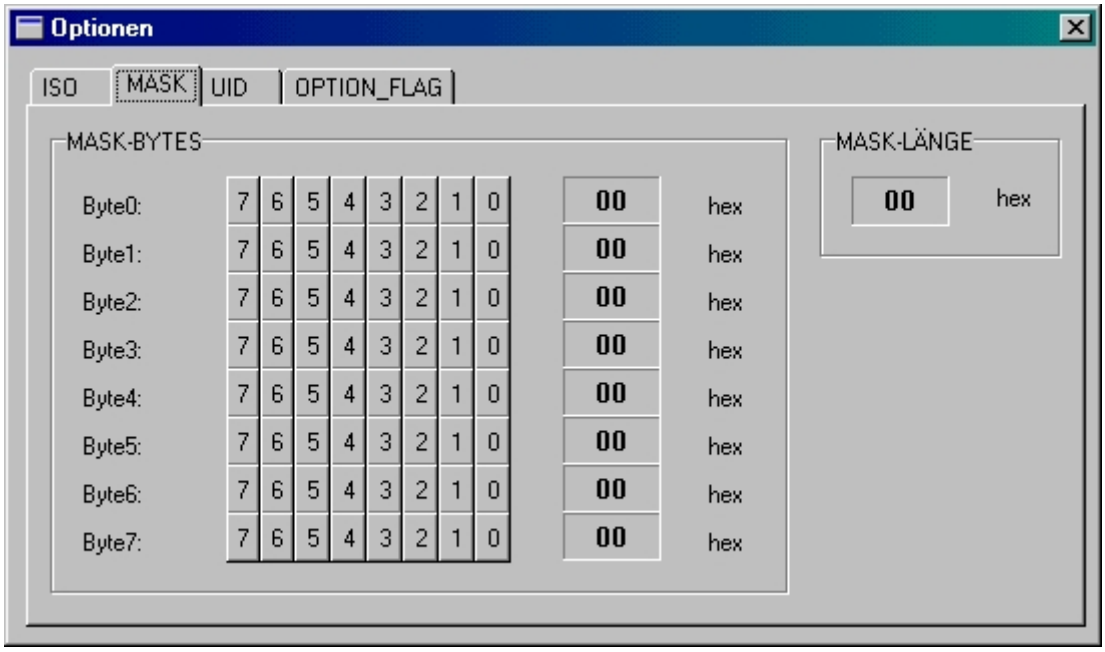


Die Parameter der seriellen Schnittstelle stellt man unter dem entsprechenden Menüpunkt ein.
Zum einwandfreien Betrieb sollte der FIFO-Puffer der seriellen Schnittstelle abgeschaltet werden.
(Rechtsklick auf Arbeitsplatz -> Eigenschaften -> Gerätemanager -> Doppelklick auf Anschlüsse COM1 und LPT -> COM1-Anschluss auswählen -> Anschlusseinstellungen -> Erweitert)

Unter *'Optionen'* werden die Übertragungsparameter für die Kommunikation mit dem Transponder festgelegt. **Diese müssen vor der Kommunikation unbedingt richtig eingestellt worden sein! Bei einer Fehleinstellung kann die korrekte Verbindung zum Transponder nicht aufgenommen werden!**



Als Kontrolle wird das eingestellt Konfigurationsbyte in binärer Form als auch in hexadezimaler Form angezeigt. *Defaulteinstellung ist: E0hex*



Unter *'Terminal'* öffnet sich ein Terminalfenster für die serielle Schnittstelle. Hier können die Befehle direkt im ASCII-Format eingegeben und ausprobiert werden, unabhängig vom eigentlichen Testprogramm. Natürlich kann auch jedes andere Terminal-Programm zum Austesten benutzt werden.

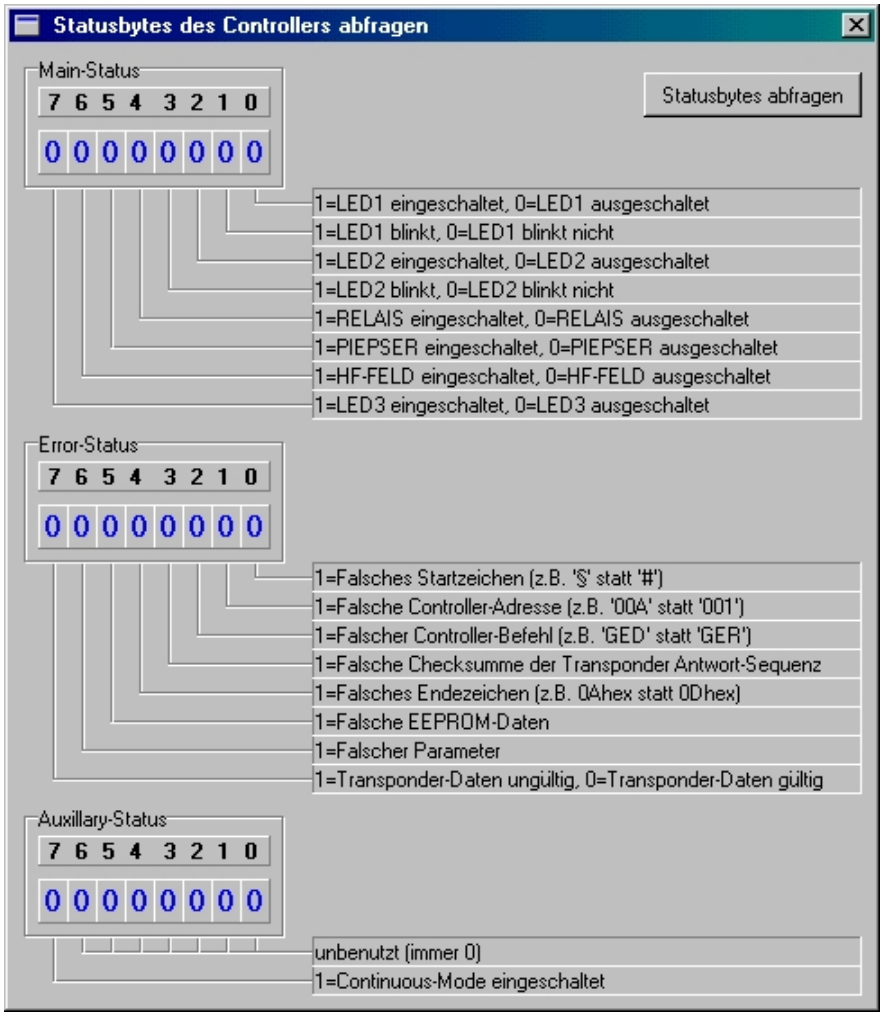
Der Eröffnungsbildschirm des Testprogramms bietet drei Karteireiter für alle implementierten Controller-Befehle und einen Karteireiter zum Lesen und Schreiben des gesamten Transponderspeichers. Auf dem ersten Karteireiter "CONTROLLER-BEFEHLE" sind alle internen Befehle des ISO-Readers aufgeführt. Diese haben mit dem Lesen oder Schreiben der Transponder nichts zu tun.

Nach Programmstart sollte der Anwender als erstes den Befehl

"GER: Geräteinformationen auslesen"

ausführen. Ist der ISO-Reader mit Spannung versorgt und die serielle Schnittstelle am PC angeschlossen und auf 9600Baud konfiguriert, dann sollten als Antwort auf diesen Befehl nun die 40Bytes an Geräteinformationen zurückgegeben werden.
Anhand dessen kann auf die korrekte Funktionsweise der Verbindung zwischen PC und ISO-Reader geschlossen werden.

Andere Befehle dienen dazu die Hardwarekomponenten des ISO-Readers (wie LEDs, Relais, Piepser und digitale Ein-/Ausgänge) anzusprechen.
Weitere Befehle bieten die Einstellung der internen Controlleradresse und das Lesen der Statusbytes. Die Statusbytes bieten im Fehlerfall Informationen über die Art des aufgetretenen Fehlers.



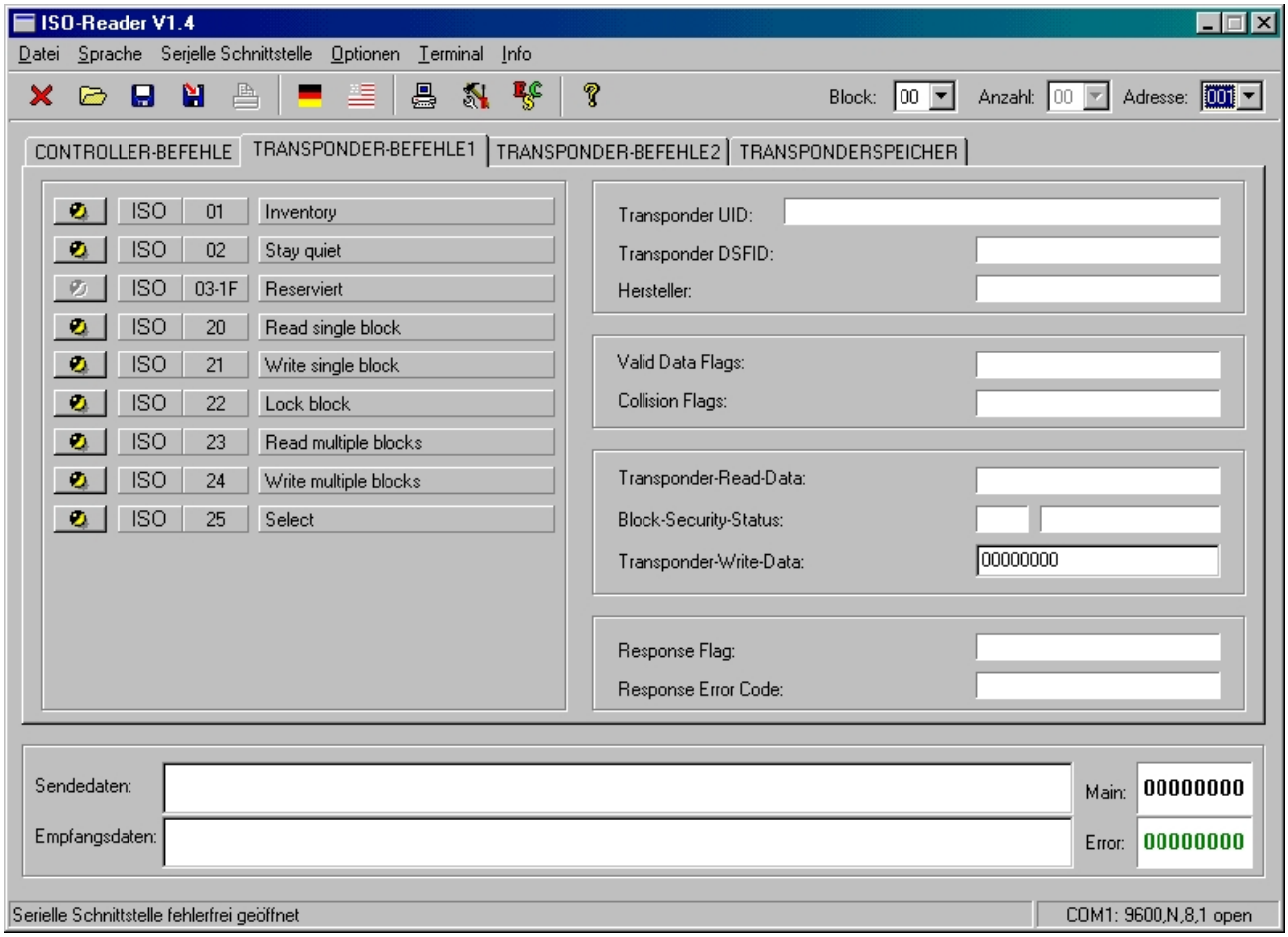
Mit den letzten drei Befehlen kann auf das interne EEPROM des Controllers zugegriffen werden. Es ist 2KByte groß**, wovon die ersten 64Bytes aber intern genutzt werden und nicht für den Anwender zur Verfügung stehen** *(Sie können lediglich gelesen, aber nicht beschrieben werden).* Die restlichen Speicherstellen des EEPROMs können wahlweise beschrieben oder gelesen werden. Das EEPROM unterstützt maximal 100000 Schreibzyklen!

In dem unteren Teil des Hauptfensters, werden die gesendeten und empfangenen Daten als ASCII und als HEX-Zeichen dargestellt, so dass der Anwender die übermittelten Zeichen schnell überprüfen kann. Dies ist vor allem bei der Entwicklung eigener Software sehr hilfreich.

Rechts unten werden dann noch die beiden Statusbytes (MAIN-STATUS und ERROR-STATUS) in binärer Form angezeigt.

In der rechten oberen Ecke kann die Adresse des Controllers eingestellt und mit dem entsprechenden Befehl (ADR) geändert werden.

Die Anzeigen für Block und Anzahl sind erst bei den Transponderbefehlen relevant und ansonsten deaktiviert.



Dieser Karteireiter zeigt die erste Hälfte der implementierten Transponderkommandos nach ISO15693.

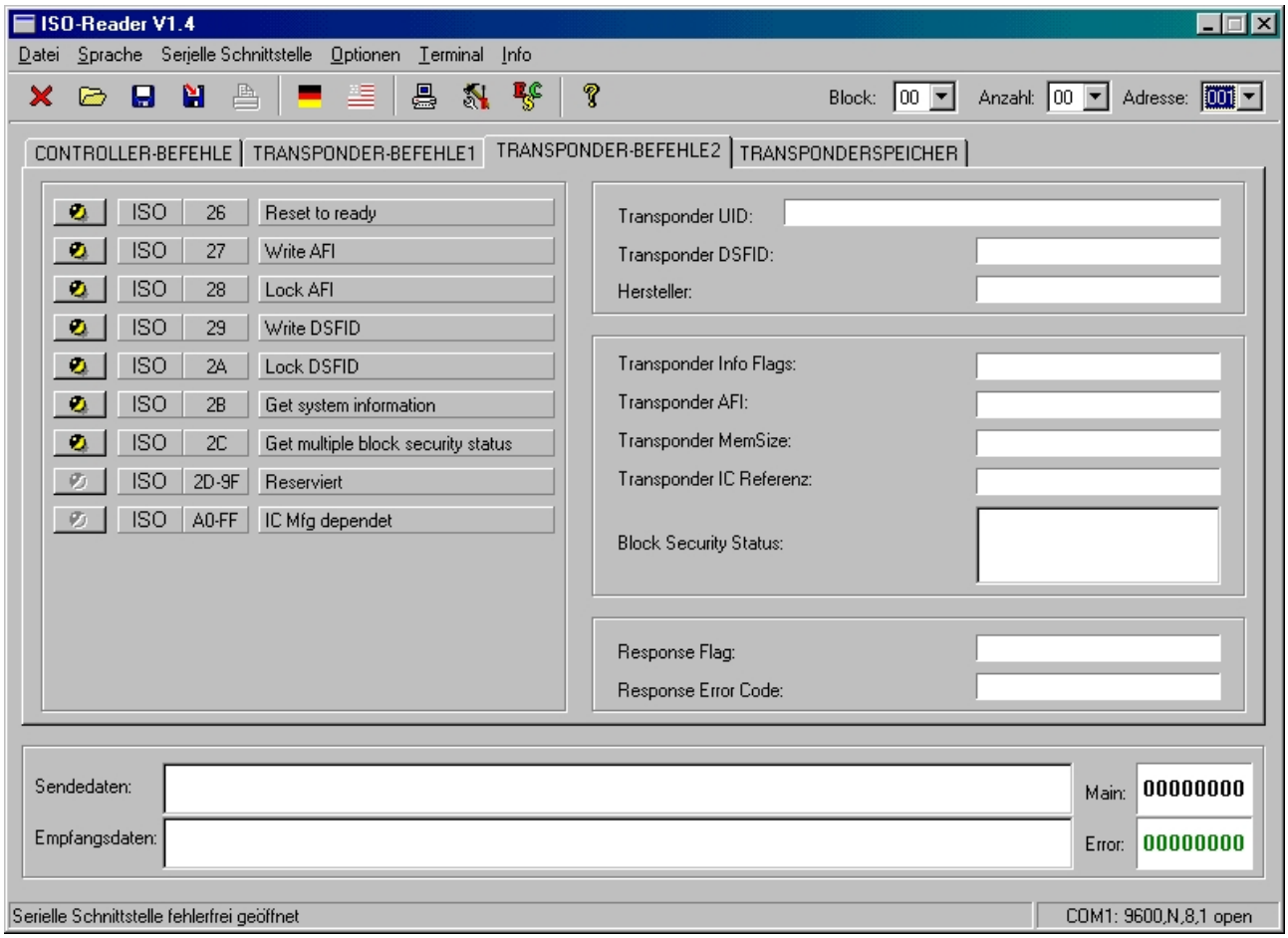
Wichtig für alle nachfolgenden Befehle ist, dass als erstes das 'Inventory' des Transponder gelesen wird, da die nachfolgenden Befehle die UID bzw. die DSFID benötigen (sofern man den Transponder adressiert ansprechen will). Ein nicht adressiertes Ansprechen ist aber auch möglich. Lediglich der "Select-Befehl" setzt die Angabe der UID voraus!
Gleichzeitig kann mit diesem Befehl die generelle Funktionsfähigkeit der Kommunikation zwischen ISO-Reader und Transponder festgestellt werden.

Befindet sich ein ISO15693 Transponder im Antennenfeld, so wird dieser nach Empfang des Inventory-Befehls seinerseits die gewünschten Informationen zurücksenden.
Hat der Inventory-Befehl die Transponder UID (Unique Identifier) fehlerfrei erkannt, dann kann man den Transponder von nun an einwandfrei identifizieren.

Ferner finden sich auf dieser Seite Befehle um einen Datenblock (jeweils 4 Bytes) aus dem Transponder zu lesen (Read Single Block) oder in ihn zu schreiben (Write Single Block). Mehrere Blöcke können durch Aufruf von "Read Single Block" bzw. "Write Single Block" in einer Programmschleife gelesen bzw. geschrieben werden.
Beachte: Die optionalen ISO-15693-Befehle "Read Multiple Blocks" und "Write Multiple Blocks" sind in der Firmware des ISO-Readers nicht implementiert!

Es können auch bestimmte Datenblöcke permanent gesperrt werden.
Vorsicht: Ein einmal gesperrter Block kann nie wieder entsperrt werden!!!

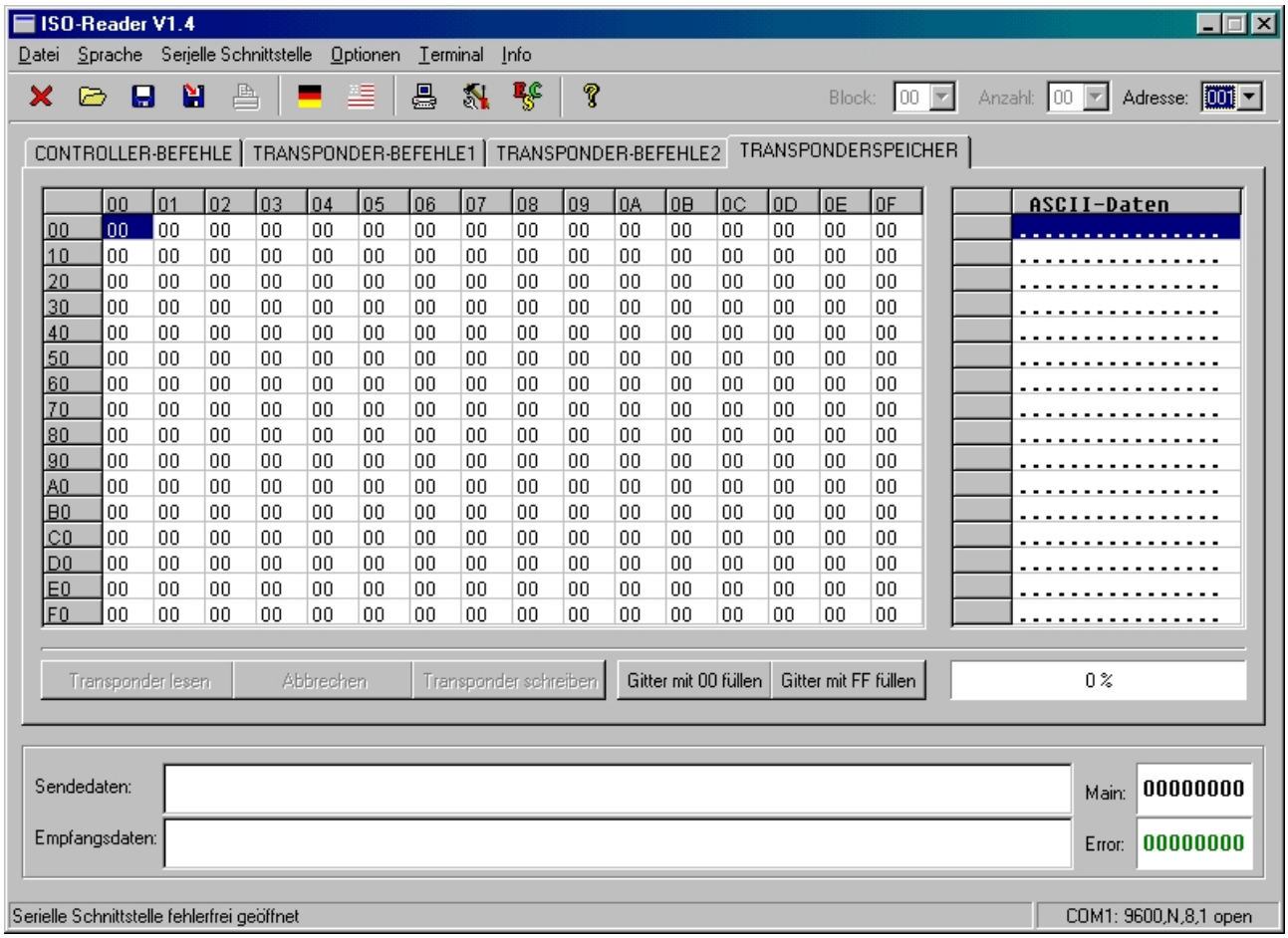
Mit dem Befehl "CONTINUOUS INVENTORY MODE" kann man den ISO-Reader in einen speziellen Modus versetzen, bei dem er von sich aus in bestimmten Zeitintervallen, das Inventory-Kommando ausführt und nachsieht, ob sich ein Transponder im Feld befindet.
Ist dies der Fall, dann wird die UID ausgelesen und sofort (ohne dass der Benutzer einen Befehl hierzu geben müsste) auf der seriellen Schnittstelle ausgegeben.
Wird ein Transponder erkannt, dann kann wahlweise dazu noch das Relais eingeschaltet oder der Signalgeber aktiviert werden.



Dieser Karteireiter zeigt die zweite Hälfte der implementierten Transponderkommandos nach ISO15693.

Hier kann man als erstes das 'Get system information'-Kommando ausführen. Anhand der zurückgelieferten Informationen kann die Größe des Transponderspeichers bestimmt werden und ob er DSFID *(Data Storage Format Identifier*) und AFI (*Application Family Identifier*) unterstützt oder nicht.

Wie auch schon auf der vorigen Seite, werden hier das 'Response-Flag' und der 'Response Error Code' angezeigt. Diese Statusbytes werden direkt vom Transponder ausgegeben und liefern Informationen darüber, ob der Transponder den Befehl fehlerfrei ausgeführt hat und ob die Daten gültig sind.



Der vierte Karteireiter schließlich zeigt den gesamten zur Verfügung stehenden Transponderspeicher an. Dieser kann hier komfortabel gelesen und beschrieben werden.

Die Daten können entweder im linken Feld direkt in HEX eingegeben werden oder im rechten Feld als ASCII-Zeichen. Ferner ist es auch möglich diese Daten als Datei zu speichern oder aber eine Datei mit diesen Daten zu laden.
Eine Druckfunktion ist nicht implementiert!

Für Testzwecke oder um den Transponderspeicher zu löschen, kann der gesamte Speicher auch auf 00hex oder FFhex gesetzt werden.

5. Befehlsübersicht:

5.1 Allgemeines:

Die Kommunikation zwischen Host (Computer) und Controller (ISO-Reader) funktioniert wie folgt:
Der Host initiiert immer die Kommunikation. Der Controller arbeitet immer als Slave und gibt nur auf Anforderung des Hosts eine Antwort (Ausnahme: Continuous-Mode).
Dadurch ergeben sich Anforderungs- und Antwortsequenzen, wobei der Host auf die Antwort des Controllers warten muss.
Alle Befehle (Pakete) sind in einen definierten Rahmen bestehend aus Startzeichen, Befehlskette und Stopzeichen gepackt.

Jedes Anforderungspaket vom Host an den Controller ist wie folgt aufgebaut:

SOF (Startzeichen)	*Adresse*	*Befehl*	*Daten*	*EOF (Endezeichen)*
#	001	GER	...	CR

Jedes Antwortpaket vom Controller an den Host ist wie folgt aufgebaut:

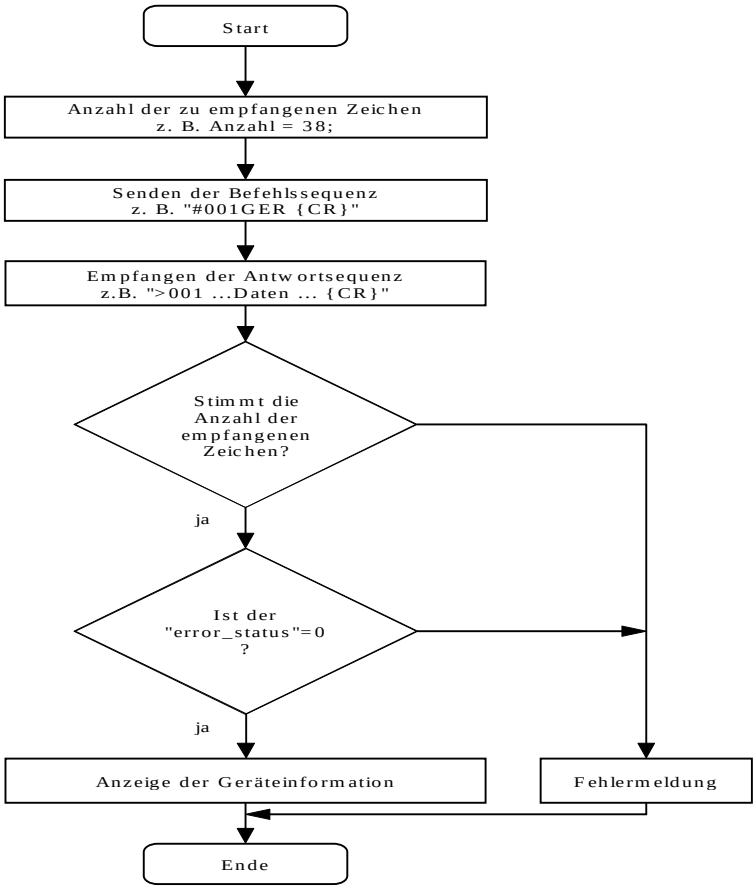
SOF (Startzeichen)	*Daten*	*Main-Status*	*Error-Status*	*EOF (Endezeichen)*
>	...	00 (befehlsabhängig)	00	CR

In jedem Antwortpaket des Controllers wird der "Error-Status" gesendet. Anhand dessen kann festgestellt werden, ob während der Bearbeitung des Kommandos ein Fehler aufgetreten war und ob die Daten vom Transponder gültig sind.

Dieser "Error-Status" sollte nach jedem Befehl vom Anwender geprüft werden. Nur wenn alle Bits des "Error-Status" gelöscht sind (Error-Status=00hex), wurde der Befehl fehlerfrei ausgeführt.

Sind eins oder mehrere Bits gesetzt, dann war ein Fehler aufgetreten und das Anwenderprogramm sollte eine entsprechende Fehlermeldung ausgeben bzw. den letzten Befehl noch einmal wiederholen.

Das nebenstehende Diagramm verdeutlicht dies anhand des "GER-Befehls".



5.2 Controller-Befehle:

In nachfolgender Tabelle sind alle Controller-Befehle und ihre Bedeutung kurz aufgeführt:

Befehl	Funktion
GER	Auslesen der Geräteinformationen des Controllers
LED	LEDs einschalten, ausschalten oder blinken lassen
REL	Relais einschalten oder ausschalten
BUZ	Buzzer einschalten oder ausschalten
TRA	Transmitter einschalten oder ausschalten
INP	Digitale Eingänge abfragen
ADR	Einstellen einer neuen Adresse des Controllers
EEP	Aus dem EEPROM des Controllers lesen (Blockweises Lesen von je 64Bytes)
RDE	Aus dem EEPROM des Controllers lesen
WRE	In das EEPROM des Controllers schreiben
PAR	Parameter im EEPROM des Controllers sichern
CON	Continuousmode einschalten oder ausschalten
BAU	Baudrate des Controllers einstellen
STA	Statusbytes des Controllers abfragen

Auslesen der Geräteinformationen des Controllers:

Befehl an den Controller:		
# AAA GER CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"GER"	3 Byte Controller-Befehl: Geräteinformationen lesen
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA "..." ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	"..."	40 Bytes Geräteinformation des Controllers als String
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 999 GER CR		23 39 39 39 47 45 52 0D (8 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 … 00h CR		3E 30 30 31 ... 00 0D (46 Hexbytes werden empfangen)

Beachte: *Adresse "999" dient zum Auslesen der Geräteinformationen, falls ein Controller mit unbekannter Geräteadresse angeschlossen werden soll. Dann kann durch Eingeben von "999" dessen aktuell eingestellte Geräteadresse und die anderen Einstellungsparameter ermittelt werden.*

Bei Controllern mit RS-485-Schnittstelle dürfen in diesem Fall keine anderen Controller gleichzeitig dranhängen, sondern immer nur der unbekannte Controller, bei dem die Geräteinformation ausgelesen werden soll.

Bedeutung der Geräteinformationen (Länge: 40 Byte):

Byte:	Länge:	Bedeutung:
1 – 5	5 Byte	Hardwareversion des Controllers - z.B. "1.00 "
6 – 10	5 Byte	Softwareversion des Controllers - z.B. "1.00
11 – 21	11 Byte	Datum (Erstellungsdatum der Controller-Software) - z.B. "08.07.2002
22 – 32	11 Byte	Name des Controller-Modells - z.B. "ISO-Reader
33 – 37	5 Byte	Build-Nummer der Firmware - z.B. "1234
38 – 40	3 Byte	Adresse des Controller-Modells - z.B. "001"

(unbenutzte Stellen sind durch Leerzeichen 20Hex aufgefüllt)

LEDs einschalten, ausschalten oder blinken lassen:

Befehl an den Controller:		
# AAA LED N P CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"LED“	3 Byte Controller-Befehl: LED schalten
	N	1 Byte LED-Nummer: 1=LED1, 2=LED2, 3=LED3
	P	1 Byte Parameter: 0=LED aus, 1=LED ein, 2=LED blinkt (nicht LED3)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA STA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	STA	1 Byte Main-Status
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 LED 1 0 CR		23 30 30 31 4C 45 54 31 30 0D (10 Hexbytes werden gesendet: LED1 aus)
# 001 LED 2 1 CR		23 30 30 31 4C 45 54 32 31 0D (10 Hexbytes werden gesendet: LED2 ein)
Empfangs-Beispiel:		
> 001 01h 00h CR		3E 30 30 31 01 00 0D (7 Hexbytes werden empfangen: LED1 ein)
> 001 03h 00h CR		3E 30 30 31 03 00 0D (7 Hexbytes werden empfangen: LED1 blinkt)

Beachte: LED3 kann nur ein- oder ausgeschaltet werden. Sie blinkt nie!

RELAIS einschalten oder ausschalten:

Befehl an den Controller:		
# AAA REL P CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"REL“	3 Byte Controller-Befehl: RELAIS schalten
	P	1 Byte Parameter: 0=RELAIS aus, 1=RELAIS ein
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA STA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	STA	1 Byte Main-Status
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 REL 0 CR		23 30 30 31 52 45 4C 30 0D (9 Hexbytes werden gesendet: RELAIS aus)
# 001 REL 1 CR		23 30 30 31 52 45 4C 31 0D (9 Hexbytes werden gesendet: RELAIS ein)
Empfangs-Beispiel:		
> 00110h 00h CR		3E 30 30 31 10 00 0D (7 Hexbytes werden empfangen: RELAIS ein)
> 001 00h 00h CR		3E 30 30 31 00 00 0D (7 Hexbytes werden empfangen: RELAIS aus)

BUZZER einschalten oder ausschalten:

Befehl an den Controller:		
# AAA BUZ P CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"BUZ“	3 Byte Controller-Befehl: BUZZER ein-/ausschalten
	P	1 Byte Parameter: 0=BUZZER aus, 1=BUZZER ein
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA STA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	STA	1 Byte Main-Status
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 BUZ 0 CR		23 30 30 31 42 55 5A 30 0D (9 Hexbytes werden gesendet: BUZZER aus)
# 001 BUZ 1 CR		23 30 30 31 42 55 5A 31 0D (9 Hexbytes werden gesendet: BUZZER ein)
Empfangs-Beispiel:		
> 001 20h 00h CR		3E 30 30 31 20 00 0D (7 Hexbytes werden empfangen: BUZZER ein)
> 001 00h 00h CR		3E 30 30 31 00 00 0D (7 Hexbytes werden empfangen: BUZZER aus)

TRANSMITTER einschalten oder ausschalten:

Befehl an den Controller:		
# AAA TRA P CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"TRA“	3 Byte Controller-Befehl: HF-Feld des Transmitters ein-/ausschalten
	P	1 Byte Parameter: 0=TRANSMITTER aus, 1=TRANSMITTER ein
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA STA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	STA	1 Byte Main-Status
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 TRA 0 CR		23 30 30 31 42 55 5A 30 0D (9 Hexbytes werden gesendet: Transmitter aus)
# 001 TRA 1 CR		23 30 30 31 42 55 5A 31 0D (9 Hexbytes werden gesendet: Transmitter ein)
Empfangs-Beispiel:		
> 001 40h 00h CR		3E 30 30 31 40 00 0D (7 Hexbytes werden empfangen: TRANSMITTER ein)
> 001 00h 00h CR		3E 30 30 31 00 00 0D (7 Hexbytes werden empfangen: TRANSMITTER aus)

Digitale Eingänge INP abfragen:

Befehl an den Controller:		
# AAA INP N CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"INP“	3 Byte Controller-Befehl: Digitale Eingänge abfragen
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA EEE ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	EEE	3 Byte Antwort vom Controller: Eingang1, Eingang2, Eingang3: 0=LOW, 1=HIGH
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 INP CR		23 30 30 31 49 4E 50 31 0D (8 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 0 0 0 00h CR		3E 30 30 31 30 30 30 00h 0D (9 Hexbytes werden empfangen: Eingänge sind LOW)
> 001 1 1 1 00h CR		3E 30 30 31 31 31 31 00h 0D (9 Hexbytes werden empfangen: Eingänge sind HIGH)

Einstellen einer neuen Adresse des Controllers:

Befehl an den Controller:		
# AAA ADR NNN CC CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ADR	3 Byte Controller-Befehl: Adresse des Controllers setzen
	NNN	3 Byte Neue Adresse des Controllers (z.B. "002")
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Antwort des Controllers		
> AAA NNN ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Alte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	NNN	3 Byte Neue Adresse des antwortenden Controllers (z.B. jetzt "002")
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ADR 002 CR		23 30 30 31 41 44 52 30 30 32 0D (11 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 002 00h CR		3E 30 30 31 30 30 32 00h 0D (9 Hexbytes werden empfangen)

Beachte: Bei der Busfähigen Version des Controllers dürfen bei diesem Befehl keine anderen Controller gleichzeitig angeschlossen sein. Immer nur der Controller, bei dem die Adresse geändert werden soll.
Nach erfolgreicher Adressänderung ist dieser Reader dann natürlich nur noch über die neu eingestellte Adresse ansprechbar!

Aus dem EEPROM des Controllers lesen:

Befehl an den Controller:		
# AAA RDE PAGE CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	RDE	3 Byte Controller-Befehl: Internes EEPROM des Controllers lesen
	PAGE	1 Byte Seitenadresse: 00hex...FFhex (Jede Seite hat 8 Bytes)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Antwort des Controllers		
> AAA DATA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	DATA	8 Byte Inhalt des internen EEPROMs
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		Die Daten von Seite 0 werden gelesen
# 001 RDE 08h CR		23 30 30 31 52 44 45 08 0D (9 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 12345678 00h CR		3E 30 30 31 31 32 33 34 35 36 37 38 00h 0D (14 Hexbytes werden empfangen)

Beachte: Das interne EEPROM des Controllers ist 2Kbyte (2048 Bytes) groß. Es ist unterteilt zu 256 Seiten mit je 8 Bytes (das sind 2048 Bytes). Die ersten 8 Seiten (64 Bytes) enthalten die Programmparameter, die dauerhaft dort abgelegt werden. Auf diesen Bereich kann vom Benutzer nur lesend zugegriffen werden.

Aus dem EEPROM des Controllers lesen (Blockweises Lesen von je 64 Bytes):

Befehl an den Controller:		
# AAA EEP BLK CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	EEP	3 Byte Controller-Befehl: Internes EEPROM des Controllers lesen
	BLK	1 Byte Blockadresse: 00hex...1Fhex (Jeder Block hat 64 Bytes)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Antwort des Controllers		
> AAA DATA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	DATA	64 Bytes Inhalt des internen EEPROMs (=1 Block)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		Die Daten von Block 0 werden gelesen
# 001 EEP 00h CR		23 30 30 31 45 45 50 00 0D (9 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 DATA 00h CR		3E 30 30 31 ... 00h 0D (70 Hexbytes werden empfangen)

Beachte: Das interne EEPROM des Controllers ist 2Kbyte (2048 Bytes) groß. Es ist unterteilt zu 256 Seiten mit je 8 Bytes (das sind 2048 Bytes). Die ersten 8 Seiten (64 Bytes) enthalten die Programmparameter, die dauerhaft dort abgelegt werden. Auf diesen Bereich kann vom Benutzer nur lesend zugegriffen werden.
Ein Block besteht aus 64 Bytes (8 Seiten zu je 8 Bytes).

In das EEPROM des Controllers schreiben:

Befehl an den Controller:		
# AAA WRE PAGE DATA CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	WRE	3 Byte Controller-Befehl: Internes EEPROM des Controllers schreiben
	PAGE	1 Byte Seitenadresse: 08hex...FFhex (Jede Seite hat 8 Bytes)
	DATA	8 Byte Daten die ins interen EEPROM geschrieben werden sollen
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Antwort des Controllers		
> AAA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		Die Daten werden an Seite 8 geschrieben
# 001 WRE 08h 12345678 CR		23 30 30 31 57 52 45 08 31 32 33 34 35 36 37 38 0D (17 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 00h CR		3E 30 30 31 00h 0D (6 Hexbytes werden empfangen)

Beachte: Das interne EEPROM des Controllers ist 2Kbyte (2048 Bytes) groß. Es ist unterteilt zu 256 Seiten mit je 8 Bytes (das sind 2048 Bytes). Die ersten 8 Seiten (64 Bytes) enthalten die Programmparameter, die dauerhaft dort abgelegt werden. Auf diesen Bereich kann vom Benutzer nur lesend zugegriffen werden. **!!! Maximal 100 000 EEPROM-Schreibzyklen**

Parameter im EEPROM des Controllers sichern:

Befehl an den Controller:		
# AAA PAR CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	PAR	3 Byte Controller-Befehl: Parameter im internen EEPROM sichern
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Antwort des Controllers		
> AAA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		Die Daten werden an Seite 8 geschrieben
# 001 PAR CR		23 30 30 31 50 41 52 0D (8 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 00h CR		3E 30 30 31 00h 0D (6 Hexbytes werden empfangen)

Beachte: Die ersten 8 Seiten (64 Bytes) des internen EEPROMs enthalten die Programmparameter, die dauerhaft dort abgelegt werden. Auf diesen Bereich kann vom Benutzer nur lesend zugegriffen werden.

CONTINUOUS-Mode ein- oder ausschalten:

Befehl an den Controller:		
# AAA CON P B CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"CON“	3 Byte Controller-Befehl: CONTINUOUS-Mode
	P	1 Byte Parameter: 0=CONTINUOUS-Mode aus, 1=CONTINUOUS-Mode ein
	B	1 Byte Parameter: 0=Buzzer bei Erkennung der UID bleibt aus, 1=Buzzer bei Erkennung der UID einschalten
	R	1 Byte Parameter: 0=Relais bei Erkennung der UID bleibt aus, 1=Relais bei Erkennung der UID einschalten
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA AUX ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	AUX	1 Byte Auxillary-Status (00hex = "Normal Mode", 80hex = "Continuous Mode")
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 CON 1 1 1 CR		23 30 30 31 43 4F 4E 31 31 31 0D (11 Hexbytes werden gesendet: CONTINUOUS-Mode ein, Buzzer ein und Relais ein)
# 001 CON 0 0 0 CR		23 30 30 31 43 4F 4E 30 30 30 0D (11 Hexbytes werden gesendet: CONTINUOUS-Mode aus, Buzzer aus und Relais aus)
Empfangs-Beispiel:		
> 001 00h CR		3E 30 30 31 80 00 0D (7 Hexbytes werden empfangen)

Beachte: Beim Ausschalten des Continuous-Mode sind die Parameter für den Buzzer und das Relais irrelevant und werden nicht ausgewertet. Sie werden nur mitgegeben, damit die Befehlssyntax eingehalten wird.

Um den ISO-Reader immer schon beim Einschalten im Continuous-Mode zu betreiben geht man wie folgt vor:

- ISO-Reader einschalten und Continuous-Mode aktivieren
- Parameter im internen EERPOM sichern und ISO-Reader ausschalten
- ISO-Reader wieder einschalten.

Zur optischen und akustischen Anzeige das der ISO-Reader sich im Continuous-Mode befindet werden die drei LEDs der Reihe nach aufblinken und der Signalgeber kurz ertönen.

Baudrate des Controllers einstellen:

Befehl an den Controller:		
# AAA BAU P CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"BAU“	3 Byte Controller-Befehl: Baudrate einstellen
	P	1 Byte Parameter: 0=9600 Baud, 1=19200 Baud
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA BPS ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	BPS	1 Byte Eingestellte Baudrate: 0=9600 Baud, 1=19200 Baud
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 BAU 0 CR		23 30 30 31 42 61 75 30 0D (9 Hexbytes werden gesendet: 9600 Baud)
# 001 BAU 1 CR		23 30 30 31 42 61 75 31 0D (9 Hexbytes werden gesendet: 19200 Baud)
Empfangs-Beispiel:		
> 001 30h 00h CR		3E 30 30 31 30 00 0D (7 Hexbytes werden empfangen: 9600 Baud)
> 001 31h 00h CR		3E 30 30 31 31 00 0D (7 Hexbytes werden empfangen: 19200 Baud)

Statusbytes des Controllers abfragen:

Befehl an den Controller:		
# AAA STA CR	#	1 Byte Startzeichen für Übertragung zum Controller (23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	"STA“	3 Byte Controller-Befehl: Status abfragen
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Antwort des Controllers:		
> AAA STA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	STA	1 Byte Antwort vom Controller: main_status
	AUX	1 Byte Auxillary-Status (00hex = "Normal Mode", 80hex = "Continuous Mode")
	ERR	1 Byte Antwort vom Controller: error_status
	CR	1 Byte Endezeichen Carriage Return (0Dhex)
Sende-Beispiel:		
# 001 STA CR		23 30 30 31 53 54 41 0D (8 Hexbytes werden gesendet)
Empfangs-Beispiel:		
> 001 STA ERR CR		3E 30 30 31 00 80 00 0D (8 Hexbytes werden empfangen)

Beachte: Das "Main-Status-Byte" gibt den aktuellen Zustand der LED's, des Relais, des Buzzers und des HF-Feldes aus.
Das "Auxillary-Status-Byte" gibt den eingestellten Abfrage-Modus des ISO-Readers aus.
Ist Bit 7 gesetzt (80hex) dann befindet sich der Controller im "Continuous-Mode". In diesem Modus wird in einem festgelegten Zeitintervall intern der Inventory-Befehl ausgeführt und nachgesehen, ob sich ein Transponder im Feld befindet. Ist einer vorhanden, dann wird seine UID ausgelesen und auf der seriellen Schnittstelle sofort ausgegeben.
Ist Bit 7 gelöscht (00hex), dann befindet sich der Controller im "Normal-Mode". Das heißt, er wartet so lange bis er den Befehl vom Host (z.B. PC) bekommt um einen Transponder im HF-Feld auszulesen.
Das "Error-Status-Byte" gibt den aktuellen Fehlerzustand des Controllers aus.
Bei fehlerfreier Ausführung des Befehls muss es 00hex ein.
(Es sollte immer nach dem Empfang einer Antwortsequenz vom Controller ausgewertet werden, um sicherzustellen, dass die Ausführung des Befehls fehlerfrei verlief!)

Bedeutung des Main-Status-Bytes:

Bit:	**Beschreibung**
0	1=LED1 eingeschaltet, 0=LED1 ausgeschaltet
1	1=LED1 blinkt, 0=LED1 blinkt nicht
2	1=LED2 eingeschaltet, 0=LED2 ausgeschaltet
3	1=LED2 blinkt, 0=LED2 blinkt nicht
4	1=RELAIS eingeschaltet, 0=RELAIS ausgeschaltet
5	1=BUZZER eingeschaltet, 0=BUZZER ausgeschaltet
6	1=TRANSMITTER (HF-Feld) eingeschaltet, 0=TRANSMITTER (HF-Feld) ausgeschaltet
7	1=LED3 eingeschaltet, 0=LED3 ausgeschaltet

Bedeutung der Auxillary-Status-Bytes:

Bit:	**Beschreibung:**
0	Unbenutzt
1	Unbenutzt
2	Unbenutzt
3	Unbenutzt
4	Unbenutzt
5	Unbenutzt
6	Unbenutzt
7	1=Controller befindet sich im Continuous-Mode, 0=Continuous-Mode ausgeschaltet

Bedeutung der Error-Status-Bytes:

Bit:	Beschreibung:
0	1=Falsches Startzeichen (z.B. "§" statt "#")
1	1=Falsche Controller-Adresse (z.B. "00A" statt "001"); erlaubt sind nur die Zahlen "0" bis "9"
2	1=Falscher Controller-Befehl (z.B. "GED" statt "GER"), da der Befehl "GED" nicht existiert
3	1=Falsche Checksumme der Transponder Antwort-Sequenz
4	1=Falsches Endezeichen (z.B. 0Ahex statt 0Dhex)
5	1=Falsche Daten im EEPROM (EEPROM-ERROR)
6	1=Falscher Parameter
7	1=Transponder-Daten sind nicht gültig, 0=Transponder-Daten sind gültig

Beachte: Bei allen Controller-Befehlen wird Bit 7 immer vom ISO-Reader auf 0 gesetzt.
Bei allen ISO-Befehlen wird Bit 7 immer gesetzt, wenn ein Fehler aufgetreten war, bzw. gelöscht, wenn die Kommunikation mit dem Transponder fehlerfrei verlief.
Bei fehlerfreier Ausführung der Befehle und einer gültigen Antwort, muss das Error-Status-Byte immer 0 sein!

Aufteilung des internen EEPROM-Speichers:

Adresse (Hex):	Beschreibung:	User-Zugriff:
0000 … 0006	Identifikationsstring: MEGASET	Read-Only
0007	Immer 00hex	Read-Only
0008	unbenutzt	Read-Only
0009	LED1	Read-Only
000A	LED2	Read-Only
000B	LED3	Read-Only
000C	RELAIS	Read-Only
000D … 000F	Controller-Adresse	Read-Only
0010	Flag für Continuous-Mode	Read-Only
0011	Flag für Buzzer im Continuous-Mode	Read-Only
0012	Flag für Relais im Continuous-Mode	Read-Only
0013	Flag für eingestellte Baudrate	Read-Only
0014	Immer 00hex	Read-Only
0015 ... 003F	Reserviert für zukünftige Erweiterungen	Read-Only
0040 ... 07FF	USER-Bereich (frei verfügbar / max. 100 000 Schreibzyklen)	Read / Write

5.3 Transponder-Befehle:

Nach ISO15693-3 sind vier Kommando-Arten definiert:

1. Mandatory:	Bereich der Kommando-Codes: 01hex ... 1Fhex. Alle ISO15693 kompatible Transponder müssen sie unterstützen. (Bemerkung: der Kommando-Code 00hex ist nicht definiert)
2. Optional:	Bereich der Kommando-Codes: 20hex ... 9Fhex. Die Unterstützung dieser Codes hängt vom eingesetzten Transponder ab. Falls ein Transponder einen dieser Codes nicht unterstützt, dann wird die Fehlermeldung "Not supported" ausgegeben.
3. Custom:	Bereich der Kommando-Codes: A0hex ... DFhex. Die Unterstützung dieser Codes hängt vom Hersteller des Transponder-IC's ab. Falls ein Transponder einen dieser Codes nicht unterstützt, dann wird die Fehlermeldung "Not supported" ausgegeben.
4. Proprietary:	Bereich der Kommando-Codes: E0hex ... FFhex. Die Unterstützung dieser Codes hängt vom Hersteller des Transponder-IC's ab. Sie sind nicht im ISO15693-Standart spezifiziert und werden von den Herstellern für verschiedene Aufgaben wie Tests, Programmierung usw. genutzt. Falls ein Transponder einen dieser Codes nicht unterstützt, dann wird die Fehlermeldung "Not supported" ausgegeben.

In nachfolgender Tabelle sind alle Kommando-Codes nach ISO15693-3 und ihre Bedeutung kurz aufgeführt:

ISO-Code	Typ	Funktion	Bemerkung
"00"	-	nicht definiert	
"01"	Mandatory	Inventory	
"02"	Mandatory	Stay Quiet	
"03 ... 1F"	Mandatory	RFU	für zukünftige Erweiterungen
"20"	Optional	Read single block	
"21"	Optional	Write single block	
"22"	Optional	Lock block	
"23"	Optional	Read multiple blocks	Befehl nicht implementiert da nicht von allen Transpondern unterstützt
"24"	Optional	Write multiple blocks	Befehl nicht implementiert da nicht von allen Transpondern unterstützt
"25"	Optional	Select	
"26"	Optional	Reset to ready	
"27"	Optional	Write AFI	
"28"	Optional	Lock AFI	
"29"	Optional	Write DSFID	
"2A"	Optional	Lock DSFID	
"2B"	Optional	Get system information	
"2C"	Optional	Get multiple block security status	
"2D … 9F"	Optional	RFU	für zukünftige Erweiterungen
"A0 … DF"	Custom	IC Mfg dependent	Hersteller abhängig (nicht unterstützt)
"E0 … FF"	Proprietary	IC Mfg dependent	Hersteller abhängig (nicht unterstützt)

ISO-Kommando: Inventory:

Befehl an den Controller:		
# AAA ISO 01 CFG AFI LEN MASK CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	01	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	AFI	1 Byte AFI (Application Family Identifier), defaultmäßig immer 00hex
	LEN	1 Byte Mask length (Anzahl der signifikanten Bits der Mask Value) 00hex: Mask Value wird nicht benutzt; es werden keine Mask Value Bytes gesendet (Default) 01hex ... 40hex: Es werden die 8 Bytes Mask value gesendet (1 Slot) 01hex ... 3Chex: Es werden die 8 Bytes Mask value gesendet (16 Slots)
	MASK	8 Bytes Mask Value (wenn Mask length=00hex ist, dann werden hier keine Zeichen eingetragen)
	CR	Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		ohne Mask length
# 001 ISO 01 CFG AFI LEN CR		23 30 30 31 49 53 4F 30 31 AE 00 30 0D (13 Hexbytes werden gesendet)
Sende-Beispiel:		mit Mask length
# 001 ISO 01 CFG AFI LEN MMMMMMMM CR		23 30 30 31 49 53 4F 30 31 AE 00 01 00 00 00 00 00 00 00 01 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA VAL COL RES RESERR DSFID UID ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	VAL	2 Byte Valid Data Flags in Hex (z.B. 00hex 00hex)
	COL	2 Byte Collision Flags in Hex (z.B. 00hex 00hex)

	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	DSFID	1 Byte DSFID (Data Storage Format IDentifier)
	UID	8 Bytes UID (Unique Identifier); Dies ist die einzigartige Seriennummer des antwortenden Transponders
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 VAL COL RES RESERR DSFID UID ERR CR		3E 30 30 31 00 00 00 00 00 00 00 E0 07 00 00 01 5C F6 0E 00 0D (21 Hexbytes werden empfangen)

Beachte: Der DSFID und die UID werden immer vom Controller zurückgesendet. War beim Inventory-Kommando ein Fehler aufgetreten (siehe Error-Status), dann sind die Daten in DSFID und UID ungültig und bestehen nur aus 00hex.

Beachte: Anticollision wird in der Firmwareversion V1.4 nicht unterstützt. Deshalb muss die Mask-Länge (LEN) immer 00hex gesetzt werden!

Jeder ISO-15693-Transponder hat eine einmalige 64bit lange Seriennummer (UID) einprogrammiert. Das Format dieser UID ist wie folgt:

64 57	56 49	48 1
'E0'	**IC Mfg code**	**IC manufacturer serial number (48Bit)**
E0	*07*	*00 00 01 70 6C EA*

Den Hersteller (IC Mfg) des Transponders kann man aus dem zweiten Byte der UID ersehen (siehe ISO/IEC 7816-6A1).

IC Mfg code	*Hersteller*
02 hex	*STMicroelectronics*
04 hex	*Philips-Semiconductor*
05 hex	*Infineon*
07 hex	*Texas Instruments*

ISO-Kommando: Stay Quiet:

Befehl an den Controller:		
# AAA ISO 02 CFG UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	02	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	UID	8 Bytes Transponder UID
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ISO 02 CFG UID CR		23 30 30 31 49 53 4F 30 32 AB E0 07 00 00 01 5C F6 0E 0D (19 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 ERR CR		3E 30 30 31 00 0D (6 Hexbytes werden empfangen)

Beachte: Der Befehl "Stayquiet" kann nur im adressierten Modus ausgeführt werden. Deshalb muss das Das Select_flag bei diesem Befehl immer auf '0' stehen und das Address_flag immer auf '1' .

ISO-Kommando: Read Single Block:

Befehl an den Controller:		
# AAA ISO 20 CFG OPT UID BLK CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	20	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	BLK	1 Byte Block Nummer (64 Blöcke Verfügbar: 00 ... 3F)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		für Block 0
# 001 ISO 20 CFG OPT UID 00 CR		23 30 30 31 49 53 4F 32 30 AB 31 E0 07 00 00 01 5C F6 0E 00 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR SEC DATA ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	SEC	1 Byte Block-Security-Status: (00hex=Block nicht schreibgeschützt, 01hex=Block schreibgeschützt)
	DATA	4 Bytes Daten des ausgelesenen Blocks
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR SEC DATA ERR CR		3E 30 30 31 00 00 00 31 32 33 34 00 0D (13 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.
Wurde das OPTION_FLAG gesetzt (31hex), dann ist bei der Antwort der "Block-Secutity-Status" gültig.
Wurde das OPTION_FLAG nicht gesetzt (30hex), dann ist der zurückgegebene "Block-Security-Status" ungültig und darf nicht beachtet werden (immer 00hex).

ISO-Kommando: Write Single Block:

Befehl an den Controller:		
# AAA ISO 21 CFG OPT UID BLK DATA CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	21	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	BLK	1 Byte Block Nummer (64 Blöcke Verfügbar: 00 ... 3F)
	DATA	4 Bytes Datenblock der in den Transponder geschrieben werden soll
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		für Block 0
# 001 ISO 20 CFG OPT UID 00 31 32 33 34 CR		23 30 30 31 49 53 4F 32 30 AB 31 E0 07 00 00 01 5C F6 0E 00 31 32 33 34 0D (25 Hexbytes werden gesendet)

Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

ISO-Kommando: Lock Block:

Befehl an den Controller:		
# AAA ISO 22 CFG OPT UID BLK CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	22	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	BLK	1 Byte Block Nummer (64 Blöcke Verfügbar: 00 ... 3F)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		für Block 0
# 001 ISO 20 CFG OPT UID 00 CR		23 30 30 31 49 53 4F 32 30 AB 31 E0 07 00 00 01 5C F6 0E 00 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

Beachte: Ein einmal gesperrter Block kann NIE wieder entsperrt werden!

ISO-Kommando: Read Multiple Blocks:

Dieser optionale Befehl ist in der Firmware des ISO-Readers nicht implementiert!

ISO-Kommando: Write Multiple Blocks:

Dieser optionale Befehl ist in der Firmware des ISO-Readers nicht implementiert!

ISO-Kommando: Select:

Befehl an den Controller:		
# AAA ISO 25 CFG UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	25	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		für Block 0
# 001 ISO 25 CFG UID CR		23 30 30 31 49 53 4F 32 35 AB E0 07 00 00 01 5C F6 0E 0D (19 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Der Befehl "Stayquiet" kann nur im adressierten Modus ausgeführt werden. Deshalb muss das Select_flag bei diesem Befehl immer auf '0' stehen und das Address_flag immer auf '1' .

ISO-Kommando: Reset to Ready:

Befehl an den Controller:		
# AAA ISO 26 CFG UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	26	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		für Block 0
# 001 ISO 26 CFG UID CR		23 30 30 31 49 53 4F 32 36 AB E0 07 00 00 01 5C F6 0E 0D (19 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

ISO-Kommando: Write AFI:

Befehl an den Controller:		
# AAA ISO 27 CFG OPT UID AFI CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	27	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	AFI	1 Byte AFI (00 ... FFhex)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ISO 27 CFG OPT UID AFI CR		23 30 30 31 49 53 4F 32 37 AB 31 E0 07 00 00 01 5C F6 0E 00 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet. Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

ISO-Kommando: Lock AFI:

Befehl an den Controller:		
# AAA ISO 28 CFG OPT UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	28	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ISO 28 CFG OPT UID CR		23 30 30 31 49 53 4F 32 38 AB 31 E0 07 00 00 01 5C F6 0E 0D (20 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet. Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

Beachte: Eine einmal gesperrte AFI kann NIE wieder entsperrt werden!

ISO-Kommando: Write DSFID:

Befehl an den Controller:		
# AAA ISO 29 CFG OPT UID DSFID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	29	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	DSFID	1 Byte DSFID (00 ... FFhex)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		mit DSFID = E2
# 001 ISO 29 CFG OPT UID DSFID CR		23 30 30 31 49 53 4F 32 39 AB 31 E0 07 00 00 01 5C F6 0E E2 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet. Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

ISO-Kommando: Lock DSFID:

Befehl an den Controller:		
# AAA ISO 2A CFG OPT UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	2A	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	OPT	1 Byte Option-Flag: (siehe Bedeutung des Option-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ISO 2A CFG OPT UID CR		23 30 30 31 49 53 4F 32 41 AB 31 E0 07 00 00 01 5C F6 0E 0D (20 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR ERR CR		3E 30 30 31 00 00 00 0D (8 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

Beachte: Eine einmal gesperrte DSFID kann NIE wieder entsperrt werden!

ISO-Kommando: Get System Information:

Befehl an den Controller:		
# AAA ISO 2B CFG UID CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	2B	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		
# 001 ISO 2B CFG UID CR		23 30 30 31 49 53 4F 32 42 AB E0 07 00 00 01 5C F6 0E 0D (19 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR INFO UID DSFID AFI SIZE IC ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	INFO	1 Byte Info-Flag (siehe Bedeutung des Info-Flags)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	DSFID	1 Byte DSFID (00 ... FFhex)
	AFI	1 Byte AFI (00 ... FFhex)
	SIZE	2 Bytes Memory-Size des Transponders (z.B. 03 für 4 Blocks, 3F für 64 Blocks)
	IC	1 Byte IC-Referenz
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		
> 001 RES RESERR INFO UID DSFID AFI SIZE IC ERR CR		3E 30 30 31 00 00 0F E0 07 00 00 01 5C F6 0E 00 00 03 3F 00 00 0D (22 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.

ISO-Kommando: Get multiple block security status:

Befehl an den Controller:		
# AAA ISO 2C CFG UID 1STBLK BLKCNT CR	#	1 Byte Startzeichen für Übertragung zum Controller (immer: 23hex)
	AAA	3 Byte Adresse des Controllers in Ascii (z.B. "001")
	ISO	3 Byte Controller-Befehl: ISO-Kommando
	2C	2 Byte ISO-Kommando: "01" ... "FF"
	CFG	1 Byte Config-Byte: (siehe Bedeutung des Config-Bytes)
	UID	8 Bytes Transponder UID (z.B. E0070000015CF60E)
	1STBLK	1 Byte Nummer des ersten zu prüfenden Blocks
	BLKCNT	1 Byte Anzahl der zu prüfenden Blöcke (jeweils max. 16 Blöcke)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Sende-Beispiel:		Security Status von 8 Blöcken lesen (Block 00 bis 07)
# 001 ISO 2C CFG UID 1STBLK BLKCNT CR		23 30 30 31 49 53 4F 32 43 AB E0 07 00 00 01 5C F6 0E 00 07 0D (21 Hexbytes werden gesendet)
Antwort des Controllers		
> AAA RES RESERR SEC ERR CR	>	1 Byte Startzeichen als Antwort vom Controller (3Ehex)
	AAA	3 Byte Adresse des antwortenden Controllers in Ascii (z.B. "001")
	RES	1 Byte Response Flag (Ist Bit 0 gesetzt, dann steht im nachfolgenden Response-Error-Code der Fehlercode drin)
	RESERR	1 Byte Response-Error-Code (War Bit 0 des Response-Error-Flags nicht gesetzt, dann steht hier 00hex drin)
	SEC	1 Byte bis 16 Bytes Block Security Status
	ERR	1 Byte Error-Status (00hex wenn fehlerfreie Befehlsausführung)
	CR	1 Byte Endezeichen Carriage Return (immer: 0Dhex)
Empfangs-Beispiel:		für Security Status von 1 Block
> 001 RES RESERR SEC ERR CR		3E 30 30 31 00 00 00 00 0D (9 Hexbytes werden empfangen)

Beachte: Wird das Address_flag gesetzt, dann wird die angegebene UID des Transponders ausgewertet.
Ist das Address_flag gelöscht, dann wird die angegebene UID nicht berücksichtigt.
Eine 00hex als Anzahl der zu prüfenden Blöcke liefert den Security Status des ersten angegebenen Blockes, eine 01hex liefert den Security Status der ersten beiden angegebenen Blöcke usw.
Maximal kann der Security Status von jeweils 16 Blöcken abgefragt werden.
Überschreitet die angegebene Anzahl der zu prüfenden Blöcke die physikalisch im Transponder tatsächlich vorhandenen, dann wird Bit7 des Error-Status-Bytes gesetzt!

5.4 Erläuterungen:

Bedeutung des Response-Error-Codes:

Error-Code:	**Beschreibung:**
"00"	nicht definiert
"01"	Das Kommando wird nicht unterstützt (z.B. Anforderungscode nicht erkannt)
"02"	Das Kommando wurde nicht erkannt (z.B. Formatfehler)
"03"	Diese Kommando-Option wird nicht unterstützt
"04" ... "0E"	nicht definiert
"0F"	Unbekannter Fehler oder dieser Fehlercode wird nicht unterstützt
"10"	Der spezifizierte Block ist nicht verfügbar (oder existiert nicht)
"11"	Der spezifizierte Block ist bereits gesperrt und kann deshalb nicht noch mal gesperrt werden
"12"	Der spezifizierte Block ist gesperrt und sein Inhalt kann nicht geändert werden
"13"	Der spezifizierte Block war nicht erfolgreich programmiert worden
"14"	Der spezifizierte Block war nicht erfolgreich gesperrt worden
"15" ... "9F"	nicht definiert
"A0" ... "DF"	Kunden spezifische Error-Codes
"E0" ... "FF"	Für zukünftige Erweiterungen reserviert (RFU)

Bedeutung des Config-Bytes:

Bit:	Beschreibung:	
0	Sub-Carrier	0=AM One Subcarrier (ASK); 1=FM Two Subcarrier (FSK)
1	Data Rate	0=Low Data Rate; 1=High Data Rate
2	Select_flag	0=Befehl soll von jedem Transponder ausgeführt werden bezüglich des Address_flags 1=Befehl soll nur von dem selektierten Transponder ausgeführt werden. Das Address_flag muss 0 gesetzt werden und die UID darf nicht im Befehl enthalten sein.
3	Address_flag	0=Die Anforderung an den Transponder ist nicht adressiert. 1=Die Anforderung an den Transponder ist adressiert. Die UID muss im Befehl enthalten sein.
4	AFI	0=AFI wird nicht verwendet; 1=AFI wird verwendet
5	Timeslots	0=16 Timeslots ; 1=1 Timeslot
6	Data Encoding	0=Fast Data Encoding (1/4); 1=Normal Data Encoding (1/256)
7	Modulation	0=10% Modulation; 1=100% Modulation

Default: 11100000bin = E0hex (AM, Low Data Rate, Kein Select, Kein Address, Kein AFI, 1 Timeslot, Normal Data Encoding, 100% Modulation)

Beachte: Es sind nicht alle zur Verfügung stehenden Kombinationen zwischen Sub-Carrier, Data Rate, Data Encoding und Modulation gleichermaßen sinnvoll und funktionsfähig. Manche Kombinationen werden so nicht zusammen funktionieren !
So sollte z.B. die 10% ASK-Modulation mit "1 aus 256"-Kodierung eingesetzt werden, oder z.B die 100% ASK-Modulation mit "1 aus 4"-Kodierung. Dies hängt vom Anwendungsfall ab bzw. von der Art der Umgebung, wo der ISO-Reader eingesetzt wird.
Gute Erfahrungen wurden unter Laborbedingungen mit der Defaulteinstellung (E0hex) gemacht.

Bedeutung des Option-Flags:

Hersteller:	Bedeutung:
Philips-Semiconductor	Das Option-Flag muss bei allen Write- und Lock-Befehlen gelöscht sein. 30hex
Texas-Instruments	Das Option-Flag muss bei allen Write- und Lock-Befehlen gesetzt sein. 31hex

Function	Hex-Code	OptionFlag (Texas Instruments)	OptionFlag (Philips)
ISO15693 Mandatory Commands			
RFU (Reserved for Future Use)	0x00	-	-
Inventory	0x01	0	0
Stay Quiet	0x02	0	0
RFU (Reserved for Future Use)	0x03-0x1F	-	-
ISO15693 Optional Commands			
Read Single Block	0x20	0 / 1	0 / 1
Write Single Block	0x21	- / 1	0 / -
Lock Block	0x22	- / 1	0 / -
Read Multiple Blocks	0x23	0 / 1	0 / 1
Write Multiple Blocks	0x24	-	-
Select Tag	0x25	0	0
Reset to Ready	0x26	0	0
Write AFI	0x27	- / 1	0 / -
Lock AFI	0x28	- / 1	0 / -
Write DSFID	0x29	- / 1	0 / -
Lock DSFID	0x2A	- / 1	0 / -
Get System Info	0x2B	0	0
Get multiple Block Security Status	0x2C	0	0
RFU (Reserved for Future Use)	0x2D-0x9F	-	-
Custom Commands			
IC Mfg dependent	0xA0 – 0xDF	-	-
IC Mfg dependent	0xE0 – 0xFF	-	-

Bedeutung des AFI (Application Family Identifier):

Die AFI gibt den Typ der Zielanwendung des Transponders an. Er kann mit den nachfolgenden beiden Kommandos geschrieben und gesperrt werden. Der AFI besteht aus einem 2 Nibbles mit je 4 Bit (1 Byte hat zwei Nibbles).
Die höherwertigen 4 Bits des Nibbles (MSB) repräsentieren eine Anwendungs- Familie und die niederwertigen 4 Bits (LSB) die entsprechende Sub-Familie. Die nachfolgende Tabelle gibt einen Überblick über den Aufbau des AFI-Bytes (nach ISO/IEC 15693-3)

Die Unterstützung des AFI vom Transponder her ist optional.
Ob der eingesetzte Transponder den AFI unterstützt ist aus den Datenblättern des jeweiligen Herstellers zu entnehmen bzw. kann mit dem ISO-Kommando "Get System Information" herausgefunden werden (siehe Bedeutung des INFO-FLAG).

AFI most significant nibble (MSB)	AFI least significant nibble (LSB)	Meaning	Example
'0'	'0'	All families and subfamilies	No applicative preselection
X	'0'	All sub-families of family X	Wide applicative preselection
X	Y	Only the Yth sub-Family of family X	---
'0'	Y	Proprietary sub-family Y only	---
'1'	'0', Y	Transport	Mass transit, Bus, Airline
'2'	'0', Y	Financial	IEP, Banking, Retail
'3'	'0', Y	Identification	Access control
'4'	'0', Y	Telecommunication	Public telephiny, GSM
'5'	'0', Y	Medical	---
'6'	'0', Y	Multimedia	Internet services
'7'	'0', Y	Gaming	---
'8'	'0', Y	Data storage	Portable files
'9'	'0', Y	Item management	---
'A'	'0', Y	Express parcels	---
'B'	'0', Y	Postal services	---
'C'	'0', Y	Airline bags	---
'D'	'0', Y	RFU (Reserved for Future Use)	---
'E'	'0', Y	RFU (Reserved for Future Use)	---
'F'	'0', Y	RFU (Reserved for Future Use)	---

Beachte X = '1' bis 'F', Y = '1' bis 'F'
:

Bedeutung des DSFID (Data Storage Format Identifier):

Der DSFID zeigt an, wie die Daten im Transponder Speicher strukturiert sind.
Er besteht aus 1 Byte und kann vom User beliebig programmiert oder gesperrt werden.

Die Unterstützung des DSFID vom Transponder her ist optional.
Ob der eingesetzte Transponder den DSFID unterstützt ist aus den Datenblättern des jeweiligen Herstellers zu entnehmen bzw. kann mit dem ISO-Kommando "Get System Information" herausgefunden werden (siehe Bedeutung des INFO-FLAG).

Bedeutung des INFO-Flags:

Bit	Flag name	Value	Description
b1	DSFID	0 1	DSFID is not supported. DSFID field is not present. DSFID is supported. DSFID field is present.
b2	AFI	0 1	AFI is not supported. AFI field is not present. AFI is supported. AFI field is present.
b3	Transponder Memory Size	0 1	Information on Transponder memory size is not supported. Memory size field is not present. Information on Transponder memory size is supported. Memory field is present.
b4	IC Reference	0 1	Information on IC reference is not supported. IC reference field is not present. Information on IC reference is supported. IC reference field is present.
b5	RFU	0	---
b6	RFU	0	---
b7	RFU	0	---
b8	RFU	0	---

Transponder memory size:

MSB		LSB
16 14	13 9	8 1
RFU	Block size in bytes	Number of blocks

Beispiel: *Block size = 1Fhex: 32 Bytes, Block size = 00hex:1 Byte*
Number of blocks = FFhex: 256 Blöcke, Number of blocks = 00hex: 1 Block

6. Technische Daten:

6.1 Umgebungstemperatur und Stromaufnahme des Controllers:

Arbeitstemperatur:	0°C bis +70°C (optional -20°C bis +70°C)
Spannungsversorgung:	+5V DC ±5% / 300mA oder +11V DC bis max. +13V DC / 300mA, typ. +12V DC
Stromaufnahme:	min: 40mA typ: >150mA max. 200mA

6.2 Abmessungen des Controllers:

Version: 5V Versorgung Abmessungen L x B x H in mm	50 x 50 x 9 - ohne LEDs auf Antennenseite - ohne Pfostensteckverbinder für Inputs auf Rückseite - ohne Klemmen, Buzzer, Relais auf Rückseite 50 x 50 x 12 - mit LEDs auf Antennenseite - ohne Pfostensteckverbinder für Inputs auf Rückseite - ohne Klemmen, Buzzer, Relais auf Rückseite 50 x 50 x 18 - Komplettbestückung
Version: 12V Versorgung Abmessungen L x B x H in mm	50 x 50 x 16 - ohne LEDs auf Antennenseite 50 x 50 x 19 - Komplettbestückung
Befestigungsbohrung	3mm zentrisch in Leiterplatte

6.3 Technische Daten des Relais:

Schaltspannung:	12 Volt DC
Schaltleistung:	15 Watt (max.)
Schaltstrom:	1 A (max.)

6.4 Arbeitsfrequenz:

Die Arbeitsfrequenz des Controllers beträgt 13,56MHz

6.5 Unterstützte Transpondertypen:

Es werden die ISO15693 kompatiblen Transponder (Tag-it HF-I und i-Code) unterstützt.

7. Anhang:

7.1 Auszüge aus dem Datenblatt des ISO15693 Transponders "Tag-it® HF-I" von Texas Instruments:

7.1.1 Spezifikation:

Nachfolgend sind die technischen Daten des Tag-it HF-I Transponders aufgelistet.

Part Number	*RI-I01-112A*	*RI-I01-112B*
Supported Standard	ISO 15693-2,-3	
Recommended Operating frequency	13.56 MHz	
Passive Resonance Frequency (at +25°C)	13.86 MHz ± 200kHz (includes frequency offset to compensate further integration into paper)	14.4 MHz ± 200kHz (includes frequency offset to compensate further integration into PVC)
Max. required activation field strength read (at +25°C)	99 dBµA/m (preliminary value)	99* dBµA/m (preliminary value)
Typ. required activation field strength write (at +25°C)	101 dBµA/m (preliminary value)	101* dBµA/m (preliminary value)
Factory programmed Read Only Number	64 bits	
Memory (user programmable)	2k bits organized in 64 x 32-bit blocks	
Typical programming cycles (at +25°C)	100,000	
Data retention time (at +55°C)	> 10 years	
Simultaneous Identification of Tags	Up to 50 tags per second (reader/antenna dependant)	
Antenna size	45 mm x 45 mm (~1.77 in x ~1.77 in)	
Foil width	48 mm ± 0.5 mm (1.89 in ± 0.02 in)	
Foil pitch	48 mm +0.1mm/-0.4mm (~1.89 in)	
Thickness	Chip: 0.355mm (~0.014 in) Antenna: 0.085mm (~0.0033 in)	
Base material	Substrate: PET (Polyethylenetherephtalate) Antenna: Aluminum	
Smallest bending radius allowed	18 mm (~0.71 in)	
Operating temperature	-25°C to +70°C	
Storage temperature (single inlay)	-40°C to +85°C (warpage may occur with increasing temperature)	
Storage temperature (on reel)	-40°C to +40°C	
Delivery	Single row tape wound on cardboard reel with 500 mm diameter Reel width: approx. 60 mm (~2.36 in); inside 50 mm (~1.97 in) Hub diameter: 76.2 mm (3 in)	
Typical quantity per reel	5,000	

Note: For highest possible read-out coverage we recommend to operate readers at a modulation depth of 20% or higher
* After PVC Lamination

7.1.2 Implementierte ISO15693 Kommandos:

Diese Tabelle stellt die Befehle dar, die vom Tag-it HF-I Transponder unterstützt werden.
Zu beachten ist, dass der Tag-it HF-I den ISO-Befehl "Write Multiple Blocks" nicht unterstützt!

		Request Mode				
Request	Request Code	Inventory	Addressed.	Non-Addressed.	Select	AFI
ISO 15693 Mandatory and Optional Commands						
Inventory	0x01	✔	-	-	-	✔
Stay Quiet	0x02	-	✔	-	-	-
Read_Single_Block	0x20	✔	✔	✔	✔	✔
Write_Single_Block	0x21	-	✔	✔	✔	-
Lock_Block	0x22	-	✔	✔	✔	-
Read_Multi_Blocks	0x23	✔	✔	✔	✔	✔
Write_Multi_Blocks	0x24	-	-	-	-	-
Select Tag	0x25	-	✔	-	-	-
Reset to Ready	0x26	-	✔	-	✔	-
Write_AFI	0x27	-	✔	✔	✔	-
Lock_AFI	0x28	-	✔	✔	✔	-
Write DSFID	0x29	-	✔	✔	✔	-
Lock DSFID	0x2A	-	✔	✔	✔	-
Get_System_info	0x2B	✔	✔	✔	✔	✔
Get_M_Blk_Sec_St	0x2C	✔	✔	✔	✔	✔
TI Custom Commands						
Write_2_Blocks	0xA2	-	✔	✔	✔	-
Lock_2_Blocks	0xA3	-	✔	✔	✔	-

✔: Implemented
- : Not applicable

7.1.3 Organisation des Transponderspeichers: