INHALTSVERZEICHNIS

WICHITIGE HINWEISE:

Diese Beschreibung enthält 'geprüfte', 'ungeprüfte' und geplante Datagramme.
Zwecks Unterscheidung werden Farben verwendet:
Alle 'blau' geschriebenen Datagramme sind durch das MX32 und einer internen PC Testsoftware geprüft und arbeiten wie angegeben.
Alle 'normal' geschriebenen Datagramme sind implementiert, aber noch nicht vollständig getestet.
Alle 'rot' geschriebenen Angaben sind geplante/vorbereitete Datagramme, welche derzeit noch nicht oder fehlerhaft implementiert sind.

ÜBERSICHT:

Das derzeit genutzte ZIMO CAN Protokoll ist mittlerweile ziemlich alt (> 10Jahre) und historisch gewachsen. Daher ist es kaum möglich dieses Protokoll an neue Anforderungen anzupassen. Aus diesem Grunde werden die Geräte der Zs Serie (2010) parallel zum derzeitigen CAN Protokoll (ZCAN10) ein neues erweitertest Protokoll verwenden.

KOMPATIBILITÄT

Das neue und das alte Protokoll können am gleichen Bus verwendet werden. Die bisherigen ZIMO Geräte ignorieren das neue Protokoll da es einen 29Bit CAN Identifier verwenden, anstatt dem bisher verwendeten 11Bit Identifier. Wenn beide Protokolle über das gleiche Kabel verwendet werden so muss die Geschwindigkeit auf 125kBaud beschränkt werden, da die bisherigen Geräte bei Framing Errors 'aussteigen'. Wenn nur Geräte der Zs Serie an einem Kabel angeschlossen werden, so kann der Anwender je nach Anlagengröße zwischen 250 und 500kBaud wählen.

BITRATE UND LEITUNGSLÄNGEN

Das CAN Netzwerk kann prinzipiell Bitraten bis zu 1Mbit/s übertragen. Alle CAN-Knoten müssen die Nachricht gleichzeitig verarbeiten können. Die maximale Kabellänge ist daher abhängig von der Bitrate. Die Tabelle zeigt empfohlene Bitraten und die entsprechende maximale Kabellänge.

Bitrate	Kabellänge
10 kbits/s	6,7 km
20 kbits/s	3,3 km
50 kbits/s	1,3 km
125 kbits/s	530 m
250 kbits/s	270 m
500 kbits/s	130 m
1 Mbits/s	40 m

Die hier angegebenen Längen sind die üblicherweise machbaren Entfernung bei Verwendung eines 'normalen' Kabels, und Stichleitungen von weniger als 10% der gesamt Länge des Busses. Für eine Modellbahn Anlage sind also 125kBaud bis 500kBaud möglich. ZIMO verwendet z.B.: 125kBaud, ESU/Märklin 250kBaud. Mit hochwertigen Kabeln (CAT-5) sind ca. 25% größere Entfernung bzw. längere Stichleitungen machbar.

FEHLERERKENNUNG IM CAN NETZWERK

Das CAN-Protokoll kann Fehler selbst erkennen und signalisieren. Um Fehler zu erkennen, sind im CAN-Protokoll drei Mechanismen auf der Nachrichernebene implementiert:

1. Cyclic Redundancy Check (CRC)

Der CRC sichert die Information des Rahmens , indem sendeseitig redundante Prüfbits hinzugefügt werden. Empfangsseitig werden diese Prüfbits aus den empfangenen Bits neu berechnet und mit den empfangenen Prüfbits verglichen. Bei Nichtübereinstimmung liegt ein CRC-Fehler vor.

2. Frame-Check

Dieser Mechanismus überprüft die Struktur des übertragenen Rahmens Die durch Frame-Check erkannten Fehler werden als Formatfehler bezeichnet.

3. ACK-Fehler

Von allen Empfängern werden die empfangenen Rahmen durch positives Acknowledgement quittiert. Wird am Sender kein Acknowledgement erkannt (ACK-Fehler), so deutet dies auf einen möglicherweise nur von den Empfängern erkannten Übertragungsfehler, auf eine Verfälschung des ACK-Feldes oder auf nicht vorhandene Empfänger hin.

Außerdem sind im CAN-Protokoll zwei Mechanismen zur Fehlererkennung auf der Bitebene implementiert.

4. Monitoring

Jeder Knoten der sendet, beobachtet gleichzeitig den Busspegel. Er erkennt dabei Differenzen zwischen gesendetem und empfangenen Bit. Dadurch können alle globalen Fehler und lokal am Sender auftretenden Bitfehler sicher erkannt werden.

5. Bit-stuffing

Auf der Bitebene wird die Codierung der Einzelbits überprüft. Das CAN-Protokoll nutzt die NRZ-Codierung (Non-Return-to Zero), die eine maximale Effizienz bei der Bitcodierung gewährleistet. Dabei werden die Synchronisationsflanken nach der Methode des Bit-stuffings erzeugt, indem vom Sender nach fünf aufeinanderfolgenden gleichwertigen Bits ein Stuff-Bit mit komplementärem Wert in den Bitstrom eingefügt wird, welches die Empfänger automatisch wieder entfernen.

Werden ein oder mehrere Fehler mit Hilfe der oben beschriebenen Mechanismen von mindestens einem Knoten entdeckt, so wird die laufende Übertragung durch Senden eines "Error flag" abgebrochen. Dadurch wird die Annahme der übertragenen Nachricht durch andere Stationen verhindert und somit die netzweite Datenkonsistenz sichergestellt. Nach Abbruch der Übertragung einer fehlerhaften Botschaft beginnt der Sender automatisch, seine Nachricht erneut zu senden (Automatic Repeat Request).

Tritt ein Fehler mehrmals aufeinanderfolgend auf, führt dies zur automatischen Abschaltung des Knotens.

PRINZIP DES DATENAUSTAUSCHES IM CAN NETZWERK

Bei der Datenübertragung in einem CAN Bus werden keine Knoten adressiert, sondern der Inhalt einer Nachricht (z.B. Lok Geschwindigkeit oder Weichenstellung) wird durch einen eindeutigen Identifier gekennzeichnet. Neben der Inhaltskennzeichnung legt der Identifier auch die Priorität der Nachricht fest.
Mit der dann folgenden Akzeptanzprüfung stellen alle Stationen nach korrektem Empfang der Nachricht anhand des Identifiers fest, ob die empfangenen Daten für sie relevant sind oder nicht. Durch die inhaltsbezogene Adressierung wird eine hohe Flexibilität erreicht: Es lassen sich sehr einfach Stationen zum bestehenden CAN-Netz hinzufügen.
Außerdem ergibt sich die Möglichkeit des Multicasting: Eine Nachricht kann von mehreren Teilnehmern gleichzeitig empfangen und ausgewertet werden. Schaltbefehle, die von mehreren Steuergeräten als Information benötigt werden, können über das CAN-Netz so verteilt werden, dass nicht jedes Steuergerät einen eigenen Schaltbefehl benötigt.

KOLLISIONSPRÜFUNG

Jeder Teilnehmer darf Daten ohne besondere Aufforderung irgend eines Masters verschicken. Wie bei Ethernet kann es dazu kommen, dass mehrere Teilnehmer gleichzeitig senden. Die Nachricht mit dem niedrigsten Identifier setzt sich am Bus durch.
Der Identifier mit der niedrigsten Binärzahl hat somit die höchste Priorität.
Den Vorgang zur Kollisionsprüfung über den Identifier nennt man „bitweise Arbitrierung“. Entsprechend dem "Wired-and-Mechanismus", bei dem der dominante Zustand (logisch 0) den rezessiven Zustand (logisch 1) überschreibt, verlieren all diejenigen Knoten den Wettstreit um die Buszuteilung, die rezessiv senden, aber auf dem Bus dominant beobachten. Alle "Verlierer" werden automatisch zu Empfängern der Nachricht mit der höchsten Priorität und versuchen erst dann wieder zu senden, wenn der Bus frei wird.
Der CANbus ist somit ein Bussystem mit bedarfsabhängiger Buszuteilung.
Auch gleichzeitige Buszugriffe mehrerer Knoten müssen immer zu einer eindeutigen Busvergabe führen. Durch das Verfahren der bitweisen Arbitrierung über die Identifier der zur Übertragung anstehenden Botschaften wird jede Kollision nach einer berechenbaren Zeit eindeutig aufgelöst: Im CAN Standard Format sind es maximal 13 Bitzeiten, im erweiterten Format sind es maximal 33 Bitzeiten.

SCHICHTEN DER CAN-SOFTWARE UND CAN-HARDWARE

Die einzelnen Aufgaben des CANBus sind in sogenannten „Schichten“ (Layer) definiert.

- 1.Bitübertragungsschicht (Physical Layer). Diese Schicht beschreibt die physikalischen Eigenschaften, wie z.B. Stecker, Kabel, Signalpegel und wie ein Bit auf der Leitung dargestellt wird.
- 2.Übertragungsschicht (Transfer Layer)
- Die Übertragungsschicht hat die Aufgabe, das spezifizierte Busprotokoll abzuarbeiten. Dazu gehören die Erzeugung eines Übertragungsrahmens (,,Pakets"), die Anforderung des Busses mit der nötigen Erkennung des Buszustands (frei, belegt) und evtl. die Durch- bzw. Weiterführung des Zugriffs (dezentrale Buszuteilung). Dazu kommen die Aufgaben der Fehlererkennung und Fehleranzeige.
- 3.Objektschicht (Object Layer)
- Diese Schicht hat als Hauptaufgaben die Botschaftenverwaltung und Zustandsermittlung. Sie entscheidet, welche Botschaften momentan zu übertragen sind. Auf der Empfängerseite nimmt sie eine Botschaftenfilterung anhand der Kennung im Identifikationsfeld vor, d.h. eine Entscheidung, welche Botschaften vom Knoten akzeptiert werden müssen und welche nicht.
- 4.Anwendungsschicht (CAN Application Layer CAL)
- In dieser Schicht werden die zu übertragenden Daten als Botschaften bereit gestellt und mit einer Kennung versehen, die eine inhaltsbezogene Adressierung ermöglichen. Durch die Wahl der Kennung wird jede Nachricht mit einer festgelegten Priorität versehen.

ZCAN20 LAYER-7 DEFINITION

Wie oben beschrieben kümmert sich die CAN Hardware um die unteren, Hardware nahen Schichten der Kommunikation.
Damit aber eine Modell Bahn tatsächlich gesteuert werden kann, muss auch ein Layer-7 (Application/Anwendung) Layer definiert werden.
Die Layer-7 Definition enthält die Definition der 29Bit Id's und der 8 Datenbytes. Damit so ein CAN Protokoll einwandfrei funktioniert, muss sichergestellt werden, dass die 29Bit ID's des CAN Busses systemweit eindeutig ist. Zusätzlich ist zu bedenken, das praktisch alle CAN Controller die CAN Id maskieren und/oder Filtern können. Durch einen geschickten Aufbau der CAN ID's ist es also möglich alle CAN Nachrichten so zu filtern, das das jeweilige Gerät nur jene Nachrichten verarbeiten muss, welche tatsächlich relevant sind. Ein Weichendekoder bekommt so beispielsweise 'Lok Befehle' gleich gar nicht mit. Umgekehrt braucht sich ein Booster nicht um 'Schaltbefehle' kümmern.

> *Hinweis:*
> *Die aktuell verwendeten Command Codes sind noch nicht auf Ihre 'Maskierungs/Filterungs' Fähigkeiten geprüft. Dies erfolgt erst in einem zweiten Schritt.*

DER BEGRIFF OBJECT:

Im Folgenden wird häufig der Begriff 'Object' verwendet. Unter diesem Begriff kann man sich jegliches 'adressierbares' Ding auf einer Modellbahnanlage vorstellen. Konkret ist es also erst mal unerheblich ob es sich um ein Fahrzeug, eine Weiche oder einem USB Stick handelt. Wichtig ist nur, dass so ein Object eine eindeutige Adresse (=UID) im aktiven System hat und einige wenige 'Basis' Eigenschaften (Properties).
Naturgemäß verfügen konkrete Objekte neben Ihren Core Properties auch über ‚eigene' Properties, welche eben nur für dieses eine Objekt Gültigkeit haben.
Aus historischen Gründen ist es notwendig zwischen 'Legacy Objects' und ZCAN20 Objects zu unterscheiden. Legacy Objects (DCC Lok /Weichen Dekoder, ...) haben keine eigene UID und Ihre Properties sind meist über unterschiedliche CV's verteilt, bzw. teilweise nicht vorhanden. ZCAN20 (aber auch ESU/Märklin) Objects haben eine eindeutige UID und auch die notwenigen Basis Properties. Damit auch herkömmliche Dekoder und Geräte genutzt werden können muss es ein 'Verwaltungssystem' geben, welches durch Berechnung, gespeicherte Konfigurationen, etc. dem restlichen System gegenüber ZCAN20 Objekte simuliert. Dadurch verhalten sich auch 'alte/herkömmliche' Dekoder und andere Geräte nach außen hin wie ZCAN20 Objects.

NETWORK ID (NID) VERGABE:

Die Network ID wird bei Anmeldung eines 'Objects' von der Zentrale vergeben. Damit dies möglich ist, benötigt jedes Gerät primär eine **eindeutige** UID. Diese kann entweder aus der 'normalen' Geräte Nummer berechnet werden, oder bei echten CAN Geräten schon in diesem festgelegt sein. Nach dem Anmeldeprozess verwenden alle Module, Dekoder, Rückmelder, etc. Ihre NID, welche für eine Session ebenfalls **eindeutig** ist. Somit können bis zu 65536 Module innerhalb eines Systems unterschieden werden, dies sollte auch für große Anlagen reichen. Der Vorteil in der Nutzung der NID besteht darin das so nur 16Bit benötigt werden. Diese können im Befehlssatz Orthogonal sowohl als Absende wie auch als Ziel ID genutzt werden. Dadurch kann in jedem Datagramm sowohl der **Sender wie auch der Empfänger eindeutig identifiziert** werden.

AUSNAHME!!:

Normalerweise dürfen Geräte erst Befehle am CAN Bus senden, nachdem sie sich ordnungsgemäß angemeldet haben (Über eine gültige NID verfügen). Von dieser Regel ist der NOTAUS Befehl ausgenommen. Diesen darf jedes Gerät mit der temporären NID 0xFFFF senden, vollkommen egal ob angemeldet oder nicht.

A) 'LEGACY' DEVICES

Das sind all jene 'Geräte', welche durch 'Fremdprotokolle' (MM1/2, DCC, ...) angesprochen werden. Diese haben üblicherweise eine max. 4 stellige Nummer aber keine eigene UID.
Zu der jeweiligen 'simplen' Nummer wird gem. folgender Tabelle ein Offset addiert, dadurch ergibt sich eine (hoffentlich) eindeutige UID. Da hierbei das obere Word immer 0x0000 ist, kann das untere Word normalerweise auch gleich als NetworkId (NID) verwendet werden. Es muss aber in allen Geräten die Möglichkeit vorgesehen werden, das diese 'berechnete' Zuordnung beim Anmeldeprozess geändert werden kann. Ein Track-Prozessor (Gleis Signal Generator) muss beispielsweise für alle Loks eine 'Translate' Tabelle vorsehen.

TRANSLATE TABELLE FÜR LEGACY DEVICES:

UID Word1	UID Word2 Min.	UID Word2 Max	Verfügbare Adressen	
0x0000	0x0000	0x27FF	10240	DCC Loks
0x0000	0x2800	0x28FF	256	MM1/MM2 Loks
0x0000	0x2900	0x2EFF	3072	Frei [1]
0x0000	0x2F00	0x2FFF	256	Multitraktionen
0x0000	0x3000	0x31FF	512	DCC ‚Basic' Zubehördekoder
0x0000	0x3200	0x39FF	2058	DCC ‚eXtended' Zubehördekoder
0x0000	0x3A00	0x3DFF	1024	MM1/MM2 Dekoder
0x0000	0x4000	0x43FF	1024	S88 Rückmelder
0x0000	0x4400	0x45FF	1024	X-Net Dekoder
0x0000	0x4600	0x47FF	1024	X-Net FeedBack
0x0000	0x4800	0x4FFF	2048	Frei [2]
ZIMO Geräte Generation 1				
0x0000	0x5000	0x503F	64	MX1
0x0000	0x5040	0x507F	64	MX8 Module
0x0000	0x5080	0x50BF	64	MX9 Module
0x0000	0x50C0	0x50CF	16	CSA Module
0x0000	0x50D0	0x50FF	48	ABA's
0x0000	0x5100	0x51FF	256	MX10 eXtension
	0x6000	0x7FFF		RESERVIERT
mfx Adressen				
0x0000	0x8000	0xBFFF	16384	Mfx Loks
ZIMO CAN 2.xx Geräte (Auch von nicht ZIMO Herstellern)				
	0xC000	0xC0FF	256	Zentralen / Booster
	0xC100	0xC1FF	256	Spezial Geräte (IF, ...)
	0xC300	0xC3FF	256	Fahrpulte
	0xC400	0xC4FF	256	MX32 Funkmodule
	0xCF00	0xCFFF	256	Fahrstraßen
	0xD000	0xDFFF	4096	Module
	0xE000	0xEFFF	4096	Objekte
	0xF000	0xFFFF	4096	Files

B) ZCAN20 OBJECTS

Echte ZCAN20 Objects verfügen über eine UID, welche ZIMO weit eindeutig ist. Folgendes UID Nummernschema wird dabei benutzt:

0x10nnnnnn:	Basis Geräte
0x30nnnnnn:	Fahrpulte
0x40nnnnnn:	Spezial Module (Funk, RailCom, ...)
0x50nnnnnn:	Nicht Sound Dekoder
0x60nnnnnn:	Sound Dekoder
0x70nnnnnn:	Frei
0x80nnnnnn:	Stationäre Dekoder/Encoder
0x90nnnnnn:	Frei
0xA0nnnnnn:	PC Software

Die jeweils 2. Stelle kann zur weiteren Unterteilung verwendet werden, z.B.:

0x10nnnnnn:	Einfaches Basisgerät
0x12nnnnnn:	Basis Gerät mit vollwertiger Doppelendstufe
0x14nnnnnn:	Basis Gerät mit vierfach Endstufe

PC-Software Kennungen:

0xA0nnnnnn:	ZIMO
0xA4nnnnnn:	ESTGWJ
0xA5nnnnnn:	STP
0xA6nnnnnn:	TrainController

GENERELLER AUFBAU TELEGRAMME

ID Command Group	Counter	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7	Data Byte 8
Befehls Gruppe		[Ziel-Id]		[Weitere Daten ja nach Befehl]					

Die Befehlsgruppen sind so aufgebaut, das die jeweiligen CAN Geräte diese als 'Filter' Kriterium verwenden können und somit nicht alle Nachrichten am CAN Bus auswerten müssen.
Die Verwendung der Datenbytes ist vom jeweiligen Kommando abhängig.
Soweit Sinnvoll werden Sie in folgender Reihenfolge benutzt:
1. Ziel-Id
Wird verwendet, wenn ein bestimmtes Gerät angesprochen werden soll (z.B.: Eine Weiche, ein Rückmelder oder eine Lok).
2. Restliche Datenbytes
Diese werden ja nach Befehl unterschiedlich benutzt, die genau Verwendung ist bei den einzelnen Kommandos angeführt.

GRUNDSÄTZLICHER ID AUFBAU:

Hinweis: Es sind alle 29 ID Bits in Folge dargestellt, die 'CAN' internen Flags sind nicht dargestellt.

Bit 28	Bit 27 ... 24	Bit 23 ... 18	Bit 17 .. 16	Bit 15 ... 0
1	4	6	2	16
Flag ('1')	Group	Command	Mode	Network Id
Flag ('1')	Group	Counter		File Id

BESCHREIBUNG DER BIT FELDER:

ID Feld Aufteilung bei Anfragen, Befehlen, Events und Bestätigungen	
Flag	Immer '1', dient zur Unterscheidung anderer Protokolle
Group	4 Bit für die jeweilige Kommando Gruppe. Gibt die jeweilige Command Group an (Sys, FeedBack, Loco, ...)
Cmd	Dieses 6 Bit Feld enthält das jeweilige Command
Mode	0b00: Req (Abfragen) 0b01: Cmd (Steuer Befehle, Wert Setzen, ...) 0b10: Evt (Events = Ungefragte Informationen) 0b11: Ack (Bestätigung)
NetworkId	Identifikationsnummer des 'Absenders'. Primär notwendig um Kollisionen am Bus zu vermeiden. Grundsätzlich wird die Short bei der Geräte Anmeldung vom Bus Master vergeben.
ID Feld Aufteilung bei File Transfer	
Flag	Immer '1', dient zur Unterscheidung anderer Protokolle
Group	0x0F (File Transfer)
Counter	8 Bit Telegramm Zähler
File Id	File Id

COMMAND GROUP'S (BEFEHLS GRUPPEN):

Alle Befehle sind in folgenden Gruppen zusammengefasst:

Gruppe	Code	Verwendung/Inhalt
System	0x00	System kritische Aufgaben (Ein/Aus, Nothalt, ...)
Zubehör	0x01	Befehle zum steuern des Zubehörs. Damit sind sowohl Encoder/Rückmelder wie auch Dekoder gemeint.
Fahrzeuge	0x02	Befehle zum steuern der Fahrzeuge (Mobile Dekoder)
Frei	0x03	Derzeit noch unbenutzt
GBS	0x04	Telegramme für Gleisbildstellpulte
Config	0x05	Konfiguration von Geräten, ZIMO Command Language
TrackCfg	0x06	Konfiguration von 'Gleis' Zubehör
Data	0x07	Object Daten Transfer
Info	0x08	Status Meldungen, meist ungefragte Meldungen
Frei	0x09	Darf von Fremdsystemen je nach Bedarf verwendet werden.
Network	0x0A	Network Management, Modul Anmeldung, ...
File Control	0x0E	Steuer Befehle für File Transfer wie z.B.: File Open, Close, ..
File Transfer	0x0F	File 'Inhalt' (Daten) Telegramme

PC INTERFACE:

Die Verbindung zum PC kann über zwei Interfaces hergestellt werden. In beiden Fällen wird das CAN Protokoll transparent weitergeleitet.

PC USB INTERFACE

Zwischen dem Zs USB Interface (egal ob MX10 oder ZsCom) und dem PC werden die CAN Datagramme mit folgender Methode übertragen.

AUFBAU / INITIALISIERUNG DER VERBINDUNG:

Wenn der PC eine Verbindung aufbauen will so muss er den Aufbau durch senden der Zeichenfolge ‚Z11Z' (=0x5A, 0x31, 0x31, 0x5A) oder ‚Z22Z' (=0x5A, 0x32, 0x32, 0x5A) initialisieren. Erst nachdem das MX10 eine derartige Zeichenfolge ‚verstanden' hat, antwortet es mit einem ‚Ping' Telegramm.
Sollte das Ping für mehr als 500mS ausbleiben, so muss der Aufbaustring wiederholt werden.
Wenn auch nach dem dritten Versuch kein Ping kommt, so ist von einem Fehler auszugehen.

AUFBAU DER DATENTELEGRAMME FÜR DAS ZIMO 2.X FORMAT:

Für das neue CAN Protokoll werden als Telegramm Delimiter die Zeichen ‚Z2' / ‚2Z' verwendet. In diesem Falle wird der CAN Id Feldweise übertragen (Group, Direction, Command und NId). Diese Aufteilung ist insbesondere wesentlich für Übertragungen am Ethernet, da dort ja mehrere Geräte simultan kommunizieren können und auch die Datenframes erheblich umfangreicher sein können.

Header	Size	Group	Cmd+Mode	NId	Data 0 ... 8	CRC16	Tail
16 Bit	8 Bit	8Bit	8Bit	16 Bit	8x8Bit	16Bit	16 Bit
0x5A32							0x325A

Grundsätzlich werden die CAN Datagramme 1:1 in den oben definierten Frame gesendet.
Da aber eine USB (VCom) bzw. Ethernet Verbindung keine fixe Limitierung auf 8 Datenbytes hat, können auch Umfangreichere Datagramme gesendet bzw. Empfangen werden.

ETHERNET/UDP INTERFACE

Das Ethernet Interface nutzt grundsätzlich die gleiche Methode zur Daten Übertragung. Da aber Ethernet Frames deutlich größer sind als CAN / USB Frames können in einem Ethernet Frame mehrere CAN Datagramme zusammengefasst werden.

HINWEISE ZUR PROTOKOLL IMPLEMENTIERUNG

1. Grundsätzlich werden alle Befehle durch ‚ACK's' bestätigt, diese Bestätigung kann jedoch je nach Befehl einige Zeit dauern.
2. Eine PC Software MUSS auch ‚ACK's' für Befehle, welche Sie nicht selber gesendet hat verarbeiten. Das MX10 sendet ‚Befehlsbestätigungen' immer und generell an alle Schnittstellen (physischen CAN Bus, X-Pressnet, PC Interface).
3. Eine PC Software MUSS mit abweichenden Daten in einem ‚ACK' umgehen können. Es obliegt der PC die abweichenden Daten entsprechend zu verarbeiten und Darzustellen.
4. Eine PC Software MUSS davon ausgehen, das Sie nicht die einzige Befehlsquelle im System ist, das also z.B.: Fahrpulte, eventuell andere Programme oder System Interne Abläufe Befehle generieren.
5. Die ‚ACK's' repräsentieren IMMER den MX10 internen Zustand. Dieser kann aus diversen Gründen vom gewünschten ‚SOLL' Zustand einer PC Software abweichen.
6. Das MX10 ist auf maximalen Datendurchsatz ausgelegt. Per USB Interface können Befehle mit bis zu 1MBaud vom PC gesendet werden. Das entspricht etwa 5000 bis 15.000 Befehlen / Sekunde (abhängig von den übertragenen Daten Bytes).
7. Alle Befehle (CMD's) werden vom MX10 sofort im jeweils betroffenen Objekt (Fahrzeug, Weiche, MX8, MX9, …) abgebildet.
8. Jedes Objekt verwendet hat eine eigene, asynchron laufende State Engine, welche den gewünschten ‚SOLL' Zustand umsetzt.
9. Die Objekt State Engines verarbeiten die Befehle mit der jeweils möglichen Geschwindigkeit. DCC Befehle benötigen ca. 20mS bis sie einen Fahrzeugdekoder zum ersten Mal erreichen. Befehle an MX8 oder MX9 Module im Mittel nur ca. 2,5mS.
10. Daraus ergibt sich, das MX8/MX9 Befehle z.B.: Schienen Befehle (Fahrzeuge, Weichen) ‚überholen' können. Der PC MUSS also davon ausgehen, dass er ein ACK für ‚später' gesendete Befehle bekommen kann.
11. Insbesondere bei Fahrzeug Befehlen ist auf folgendes zu achten:
    Wenn an ein und das selbe Fahrzeug Geschwindigkeitsänderungen und Funktionsbefehle gesendet werden, so haben die Geschwindigkeitsbefehle VORRANG vor den Funktionsbefehlen, werden daher zuerst verarbeitet und auch bestätigt.
12. Sollten per Interface fortlaufend Geschwindigkeitsbefehle gesendet werden, schneller als diese ans Gleis gesendet werden können, so wird immer die ‚aktuellste' Geschwindigkeit gesendet. Zwischenstufen werden ignoriert und somit übersprungen.
13. Eine PC Software MUSS ebenfalls davon ausgehen, das Funktionsbefehle erheblich verzögert an das Fahrzeug gesendet werden, wenn das Fahrzeug gleichzeitig mit Geschwindigkeitsbefehlen ‚zugemüllt' wird. Im Worst Case (Weil Fahrbefehle vorliegen) wird nur jeder 32ste Befehl zum senden der Funktionen verwendet (somit kann dies 640mS dauern), für ein komplettes Update aller Funktionen (28) ergibt das ca. 1.5Sekunden!!!
14. Wie oben angeführt wird normalerweise jeder Befehl durch ‚sein' ACK bestätugt, also im wesentlichen das jeweilige Commando 1:1 als ACK mit den jeweiligen ‚IST' Daten zurückgesendet. In einigen Fällen ist dies jedoch nicht möglich. Im DCC Format werden Funktionen ja in Gruppen an das Fahrzeug übertragen. Daher erfolgt das ACK auch mit einem ‚Sammel' ACK, in welchem der Zustand aller 32 Funktionen enthalten ist.

OPTIMALE IMPLEMENTIERUNG FÜR PC SOFTWARE:

Damit eine PC Software den möglichen Interface Durchsatz nutzen kann, sollte diese folgende Implementierung umsetzten:

1. Alle von Ihr gesendeten Befehle sollten Sie in einer Liste ablegen (Ringbuffer)
2. Befehle, welche NUR Daten verändern (z.B.: Geschwindigkeit) sollen in dieser Liste nicht NEU eingetragen werden, sondern den schon vorhandenen Befehl überschreiben.
3. Alle ‚ACK's' vom MX10 sollten aus der Liste gelöscht werden, da ja ‚erledigt'
4. Gleichzeitig sollte geprüft werden, ob die ‚bestätigten' Daten (z.B.: Geschwindigkeit) von dem eigenen ‚SOLL' Wert abweichen.
5. Bei Abweichung MUSS die Software sinnvoll reagieren. Z.B.: Eigenen ‚SOLL' Wert erneut senden, Benutzer Info, bis hin zu einer automatischen Anpassung von Stellwerken, Geschwindigkeiten anderer Fahrzeuge, …
6. ACK's, welche nicht in der Liste vorhanden sind, also nicht von der PC Software gesendet wurden, müssen sinnvoll verarbeitet werden. Insbesondere sind hier ‚ALLES AUS', ‚ALLES STOP' zu beachten. Aber natürlich auch Fahrbefehle und Schaltbefehle an Fahrzeuge.
7. Je nach Befehlsart kann eine PC Software davon ausgehen, dass die ACK's 500mS bis 2000mS benötigen. Vor allem Schienen Befehle können erhebliche Verzögerungen aufweisen. Sollte die PC Software innerhalb dieser Worst-Case Zeitspanne KEIN ACK bekommen, so muss diese von einem Fehler ausgehen.

BEFEHLSSATZ:

SYSTEM CONTROL GROUP (0X00)

Die Command Group 0x00 fasst alle System ‚High-Priority' Befehle zusammen und muss von allen Boostern und Fahrpulten implementiert werden.

SYSTEM POWER (0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x00	0b00		3	SystemNId		Port					
0x00	0x00	0b01		4	SystemNId		Port	Mode				
0x00	0x00	0b11		4	SystemNId		Port	Mode				

Mit Cmd=0x00/M=0b00 kann der Power Status des jeweiligen Gerätes abgefragt werden.
ACHTUNG:
Eine Abfrage unmittelbar nach einem Power Command kann zu inkonsistenten Antworten führen!! Nach einem Power Mode Command wechselt das MX10 in den jeweils gewünschten Mode, dieser wird aber erst NACHDEM die internen Regelschleifen den Wechsel ausgeführt und durch Messungen verifiziert haben auch gemeldet. Dieser Vorgang kann je nach gewünschtem Wechsel mehrere 100ms dauern.

Mit Cmd=0x00/M=0b01 kann der Port Power Status des Gerätes gesetzt werden, nach ‚Ausführung' der Status Änderung wird der aktuelle Status per Cmd=0x00/M=0b11 ‚quittiert'.
Der jeweils gültige Status ist auch in der regelmäßigen (ca. 500ms) Power Meldung enthalten.
Die gültigen Port Nummern sind in der Tabelle Port Zuordnungstabelle zu finden.
Die gültigen Power Stati sind in der Tabelle Power Stati zu finden.

Anwendungen (egal ob per PC Interface oder an einem der internen Bussysteme) sollten nach einem Power Mode wechsel IMMER auf das jeweilige ACK des MX10 warten, womit sich im Grunde eine ‚Abfrage' erübrigt.

SYSTEM POWER SOURCE (0X04)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x04	0b01		3	SystemNId		Mode					
0x00	0x04	0b11		3	SystemNId		Mode					

Durch dieses Kommando meldet die Zentrale, die Art Ihrer Stromversorgung.
Mode = 0x01: Hauptversorgung abgeschaltet
Sobald dieses Kommando von einem Gerät (MX32) empfangen wird, soll es ‚herunterfahren'.

Mode = 0x02: Zentrale läuft mit USB Versorgung
Dieses Kommando ist eigentlich nur für PC Software von Bedeutung. Grundsätzlich werden ‚Leistungsbefehle' (FAHR, WIE) mit diesem ‚ACK' beantwortet, wenn die Zentrale per USB versorgt wird.

SYSTEM ERROR (0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x10	0b10		4	SystemNId		Code	Port				

Mit diesem Event meldet das System Fehlerzustände.
Hinweis: Die Fehler Codes und deren Bedeutung sind derzeit noch nicht definiert.

TSE ERROR (0X11)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x11	0b10		3	SystemNId		Code	Port				

Fehlermeldungen der Track Signal Engine.
Hinweis: Die Fehler Codes und deren Bedeutung sind derzeit noch nicht definiert.

ACCESSORY COMMAND GROUP (0X01)

Die Command Group 0x01 fasst die Zubehör Befehle zusammen. Als Zubehör gilt dabei jegliches stationäres Gerät angefangen bei simplen Weichendekodern oder S88 Rückmelder bis hin zu komplexen Modulen mit RailCom/mfx Empfängern.

ACCESSORY STATE (0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x00	0b00		2	ZubehörNId							
0x01	0x00	0b1x		8	ZubehörNId		State/Error		Data 1		Data2	

Diese Abfrage sollte von jedem Steuersystem (Fahrpult/Computer) immer als erstes an die Zentrale gesendet werden.
Wenn M = 0b00, DLC = 2, dann wird der Status des Zubehörs mit 'NId' angefragt.
Wenn M = 0b11, DLC = 8, dann sendet die Zentrale die Status Antwort für das jeweilige Zubehör.
Wenn ‚State/Error' = 0x0000, dann befindet sich das Modul in einem ‚normalen' Betriebszustand. In Data1 wird die CtrlNId von jenem Gerät gesendet, welches das jeweilige Zubehör zuletzt gesteuert hat. In Data2 werden die Anzahl mS gesendet, welche seit dem letzten Steuerbefehl vergangen sind.

Alle ‚States/Errors' ungleich 0x0000 sind Fehlercodes.

Hinweis:
Sollte ein Gerät für mehr als 65 Sec. (65000ms) keinen Steuerbefehl senden, so wird es zu einem nicht aktiven Steuergerät. In dem Falle sendet die Zentrale nur mehr die CtrlNId und als CtrlTick 0xFFFF.
Die Status Flags sind im Anhang aufgelistet.

ACCESSORY MODE (0X01)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x01	0b00		2	ZubehörNId							
0x01	0x01	0b1x		4	ZubehörNId		Mode					

Diese Datagramm dient der Abfrage und des einstellen der Zubehör Betriebsart.

HINWEIS, DCC BASIC DEKODER:

DCC Basic Dekoder haben eine Zubehör NId im Bereich 0x3000 … 0x31FF (Adresse 1 … 512).
Die Standard DCC Zubehör Dekoder kennen 2 Betriebsarten:
Mode ‚0‘: Default Mode (bzw. Betriebsart unbekannt)
Mode ‚1‘: Paar Betrieb (Typischerweise Weichen)
Mode ‚2‘: Einzel Betrieb (Jeder Ausgang kann getrennt geschalten werden).

Wenn das MX10 Eingeschalten wird, befinden sich alle Dekoder im ‚Default‘ Mode (‚0‘), typischerweise arbeiten DCC Dekoder dann im Paar (Weichen) Modus.
Wenn eine bestimmte Betriebsart gewünscht ist, so muss diese zuvor durch diesen Befehl für den jeweiligen Dekoder festgelegt werden. Die Festlegung gilt dann für die gesamte Session.

Hinweis:
In Zukunft wird das MX10 sich eine Festlegung der Betriebsart auch über Power On/Off merken.

HINWEIS, CSA MODULE:

Die CSA Module werden in den NId Bereich 0x50C0 bis 0x50CF gemappt.
Die CSA Module benötigen ein Accessory Mode Command, damit Sie in der richtigen/gewünschten Betriebsart operieren. Derzeit sind folgende Betriebsarten definiert:
Type = ‚1‘: 32x Eingang
Type = ‚2‘: 32x Ausgang
Type = ‚3‘: 64x PTP Ansteuerung
Type = ‚4‘: 32x PLV Ansteuerung
Type = ‚5‘: 32x Eingang, invertiert
Der Mode MUSS als erstes Kommando an die CSA Module gesendet werden.

ACCESSORY GPIO (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x02	0b00		4	ZubehörNId		Type					
0x01	0x02	0b01		8	ZubehörNId		Type		GPIO States			
0x01	0x02	0b10		8	ZubehörNId		Type		GPIO States			
0x01	0x02	0b11		8	ZubehörNId		Type		GPIO States			

Diese Datagramme dienen der effizienten Abfrage und Meldung von Simplen Ein-/Ausgängen. Es werden je Gruppe bis zu 32 Ein-Ausgangs Zustände übertragen.
Durch M=0b00 kann vom Gerät ‚ZubehörNId' der GPIO Bereich ‚Type' abgefragt werden.
Externe Änderungen werde als Event (M=0b10) gemeldet, dezidierte Abfragen werden als Ack (M=0b11) beantwortet.

HINWEIS, DCC BASIC DEKODER:

Die DCC Basic Dekoder sind in den NId Bereich 0x3000 bis 0x31FF gemappt.
Diese Datagramme werden NUR für DCC Dekoder im Einzel Betrieb (Mode=2) unterstützt. Diese Betriebsart MUSS VOR Verwendung entsprechend gesetzt werden.
Der GPIO Befehl für DCC Dekoder ist insbesondere nützlich, wenn Signale oder andere Lichteffekte geschalten werden sollen (Bitmuster für alle Ausgänge mit nur einem Befehl).
ACHTUNG:
Da es keinen DCC ‚Bitmuster' Befehl gibt, wird das übergeben Bitmuster in Einzelbefehle zerlegt und diese dann als getrennte DCC Befehle an den Dekoder gesendet.
Da jede ‚IST' Änderung entsprechende ‚Events' feuert, bekommt der PC die Rückmeldungen ebenfalls Bit für Bit gemeldet. Die PC Software muss also bei Verwendung dieser Datagramme selber für die geeignete / sinnvolle Umsetzung der Zustandsmeldungen (ACK, EVT) sorgen.

HINWEIS, MX8:

Die MX8 Module werden in den NId Bereich 0x5040 bis 0x507F gemappt.
Type='0': MX 8 in Betriebsart unbekannt ➔ FEHLER
Type='1': MX 8 in Paar/Paar Mode
Type='2': MX 8 in Paar/Einzel/ Einzel Mode
Type='3': MX 8 in Einzel/Einzel/Einzel/Einzel Mode

HINWEIS, MX9:

Die MX9 Module werden in den NId Bereich 0x5080 bis 0x50BF gemappt.

Type	Verwendung	Value
0x0000	Gleisabschnitt 1 … 16, Besetztmeldungen Da ein MX9 über 16 Abschnitte verfügt, sind nur die ersten 16 Bits benutzt.	‚0' = Frei, ‚1' = Besetzt
0x0002	MX9 Signal Ausgänge, sofern das jeweilige MX9 mit ALA Platinen bestückt ist.	‚0' = AUS, ‚1' = EIN

Sofern die Daten des MX9 zum Zeitpunkt der Abfrage ‚unklar' sind, wird die Abfrage mit einer Accessory Error Meldung (Grp=0x01, Cmd=0x00) beantwortet.

HINWEIS, CSA:

Die CSA Module werden in den NId Bereich 0x50C0 bis 0x50CF gemappt.

Type entspricht der Betriebsart des CSA Modules, unter GPIO werden die 32 Ports des CSA Modules übermittelt.

Type = ‚1‘: 32x Eingang
Type = ‚2‘: 32x Ausgang
Type = ‚3‘: 64x PTP Ansteuerung
Type = ‚4‘: 32x PLV Ansteuerung
Type = ‚5‘: 32x Eingang, invertiert

HINWEIS, STEIN:

Die StEin Module werden in den NId Bereich 0xD000 bis 0xDFFF gemappt.

Type	Verwendung	Value
0x0000	Gleisabschnitt 1 … 16, Besetztmeldungen	‚0‘ = Frei, ‚1‘ = Besetzt
0x0100	Hochleistungs-Eingänge	‚0‘ = AUS, ‚1‘ = EIN
0x0110	Hochleistungs-Ausgänge	‚0‘ = AUS, ‚1‘ = EIN
0x0200	Digital Eingänge	‚0‘ = AUS, ‚1‘ = EIN
0x0210	Digital Ausgänge	‚0‘ = AUS, ‚1‘ = EIN

ACCESSORY PORT (0X04)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x04	0b00		3	ZubehörNId		Port					
0x01	0x04	0b01		4	ZubehörNId		Port	Value				
0x01	0x04	0b1x		4	ZubehörNId		Port	Value				
					ZubehörNId		Port	Type	Value			

Wenn M = 0b00, DLC = 3, dann wird der Zustand des Ein/Ausganges (Port) vom Zubehör NId abgefragt.
Durch M = 0b01, DLC = 4 wird der Ausgang des Zubehörs (NId) auf den angegeben Wert gestellt.
Die Abfrage wird durch 0b1x beantwortet, ebenso Änderungen welche durch andere Einflüsse verursacht werden (Z.B.: Manuelles verstellen, Zeitablauf, ...)
Jedes Zubehör (Egal ob 'Schienen' gebunden oder am Bus System) darf bis zu 256 Ein/Ausgänge haben. Jeder Ausgang darf bis zu 256 'Stellungen' haben. Wie viele dieser Möglichkeiten genutzt sind, ist vom jeweiligen Modul abhängig.

HINWEIS, DCC BASIC DEKODER:

DCC Basic Dekoder haben eine Zubehör NId im Bereich 0x3000 … 0x31FF (Adresse 1 … 512).
Die Ports werden von 0 … 7 gezählt, bei Weichendekodern sind nur die geraden Port Nummern gültig (0 = Weiche 1, 2 = Weiche 2, 4 = Weiche 3, 6 = Weiche 4).
Ein Value von ‚0' bedeutet, dass der jeweilige Ausgang (Port) abgeschaltet sein soll, ein Value von ‚1', das dieser eingeschalten sein soll.
ACHTUNG:
Die tatsächliche Funktion bei DCC Dekodern ist extrem vom jeweiligen Dekoder und dessen Konfiguration abhängig. Im Grunde bewirkt dieses Befehl nur, dass die Zentrale einen DCC Befehl gemäß NMRA Norm ‚Basic Accessory Decoder Packet Format' ans Gleis sendet.
Folgende Bitzuordnung wird dabei verwendet:
NMRA Befehl: 10AAAAAA 0 1AAACDDD
A = 9 Bit Adresse des Dekoders
D = Port Nummer
C = Port Zustand

HINWEIS, DCC EXTENDED DEKODER:

DCC Extended Dekoder haben eine Zubehör NId im Bereich 0x3200 … 0x39FF (Adresse 1 … 2048). Die Port Nummer ist bei diesen Dekodern immer ‚0', mit Value wird der jeweilige Aspect angegeben.
ACHTUNG:
Die tatsächliche Funktion bei DCC Dekodern ist extrem vom jeweiligen Dekoder und dessen Konfiguration abhängig. Im Grunde bewirkt dieses Befehl nur, dass die Zentrale einen DCC Befehl gemäß NMRA Norm ‚Extended Accessory Decoder Packet Format' ans Gleis sendet.
Folgende Bitzuordnung wird dabei verwendet:
NMRA Befehl: 10AAAAAA 0 0AAA0AA1 0 000XXXXX
A = 11 Bit Adresse des Dekoders
X = Port Aspect (Value)

HINWEIS, MX8:

Die bekannten MX8 Module werden entsprechend Ihrer Adressen in den Zubehör NId Bereich gemappet.
Die MX8 Module belegen dabei den Bereich 0x5040 bis 0x507F (Max. 64).
Die MX8 Module haben 32 Ausgänge, welche je nach MX8 Konfiguration getrennt oder paarweise angesteuert werden können.
Im Paar Betrieb können Weichen entweder durch Ansteuern der ‚geraden' Port Nummer geschalten werden 0, 2, 4, … Wert 0/1 oder durch ‚1' setzten des jeweiligen Zielausganges.

HINWEIS, MX9:

Die MX9 Module werden in den NId Bereich 0x5080 bis 0x50BF gemappt.
Die Port Nummer wird für die verschiedenen MX9 Funktionen wie folgt genutzt:

Port	Verwendung
0 … 15	Gleisabschnitt 1 … 16, Besetztmeldungen
32 … 63	ALA Ausgänge
128 … 143	HLU Geschwindigkeit. Wobei die HLU Geschwindigkeit immer für Hauptabschnitt und Folgeabschnitt gemeinsam gilt.

Sofern die Daten des MX9 zum Zeitpunkt der Abfrage ‚unklar' sind, wird die Abfrage mit einer Accessory Error Meldung (Grp=0x01, Cmd==x00) beantwortet.

HINWEIS, STEIN:

Die StEin Module werden in den NId Bereich 0xD000 bis 0xDFFF gemappt.

Die Port Nummer wird für die verschiedenen StEin Ein/Ausgänge wie folgt genutzt:

Port	Verwendung	Value
0 … 15	Gleisabschnitt 1 … 16, Besetztmeldungen	‚0' = Frei, ‚1' = Besetzt
16 … 31	Hochleistungs Ausgänge (Schalten)	‚0' = AUS, ‚1' = EIN
32 … 47	Hochleistungs Eingänge (Abfrage)	‚0' = AUS, ‚5' = 5V, ‚20'=20V, 250 = Three State, 255 = ERROR
48 … 63	Digital Ein/Ausgänge	‚0' = AUS, ‚1' = EIN
64 …	Zusatzmodule	

ACCESSORY DATA (0X05)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x05	0b00		4	ZubehörNId		Port	Type				
0x01	0x05	0b01		6 .. 8	ZubehörNId		Port	Type	Data	Data	Data	Data
0x01	0x05	0b11		6 .. 8	ZubehörNId		Port	Type	Data	Data	Data	Data

Mit diesen Datagrammen können Objekt Daten abgefragt und gesetzt werden.
In einigen Fällen sind die Objekt Daten ‚Read Only', z.B.: Zugnummern können immer nur abgefragt werden, bzw. werden bei Änderung als ‚Event' gemeldet.

HINWEIS, MX9:

Die MX9 Module werden in den NId Bereich 0x5080 bis 0x50BF gemappt. Die Port Nummer wird fortlaufend von ‚0' bis ‚15' gezählt. Der ‚alte' Hauptabschnitt 1 hat daher die Port Nummer ‚0' und ‚1', usw.

Type	Verwendung	Data (DB5 … DB8)
0x00	MX9 HLU Geschwindigkeit	
0x10	MX9 Fahrzeug ‚Sammelmeldung' ACHTUNG: Nur für PC Interface	
0x11	MX9 Fahrzeug 1	Fahrzeugadresse 1 in DB5, Db6
0x12		
0x13		
0x14	MX9 Fahrzeug 4	Fahrzeugadresse 2 in DB5, Db6

Hinweis zu den MX9 Zugnummern:
Über das ‚Interface Option' Kommando (Grp=0x0A, Cmd=0x0A) kann man die Zugnummern Meldungen beeinflussen.
Solange das ‚MX9 Zugnummer 0' Bit nicht gesetzt ist, meldet das MX10 nur echte Zugnummern Meldungen vom MX9 an den PC weiter. Mit dieser Methode kann ein Zug, welcher mehrerer Abschnitte lang ist auch in allen Abschnitten gemeldet werden.
Setzt man das ‚MX9 Zugnummer 0' Interface Option Flag, dann werden ALLE MX9 Meldungen an den PC weitergeleitet. Dadurch kann es aber auch zu falschen ‚0' Meldungen kommen!!
Das MX9 meldet auch bei der geringsten Erkennungsunsicherheit SOFORT ‚Zugnummer 0', ebenfalls wenn auf einem Abschnitt ‚Lange' Zugnummern vorhanden sind!!
Bei aktivieren ‚MX9 Zugnummer 0' Bit MUSS die PC Software sich selber um alle damit verbunden Ärgernisse kümmern.

HINWEIS, STEIN:

Die StEin Module werden in den NId Bereich 0xD000 bis 0xDFFF gemappt.
Die Port Nummer wird fortlaufend von ‚0' bis ‚7' gezählt.

Type	Verwendung	Data (DB5 … DB8)
0x00	MX9 HLU Geschwindigkeit	
0x10		
0x11	StEin Fahrzeug Liste	Db5: Anzahl Fahrzeuge
0x20	StEin Fahrzeug 1	Fahrzeugadresse in DB5, Db6
0x3F	StEin Fahrzeug 32	Fahrzeugadresse in DB5, Db6

OBJECT COMMANDS (0X1N), EINFÜHRUNG MIT STEIN MODUL

Vorbereitung für die Objekt Steuerung, Einführung mit dem StEin Modul.

Mit den StEin Modulen wird die neue ZIMO Objekt Steuerlogik eingeführt und erstmalig in einem realen Modul umgesetzt.
Die objektfähigen Module (MX10, StEin) verwenden im CAN Identifier ihre physische Adresse (UId), diese ist im gesamten System eindeutig und wird vollautomatisch vergeben. Hierzu ist kein Eingriff vom Anwender her notwendig.
Die jeweiligen Objektnummern hingegen werden vom Anwender nach seiner Logik per Konfiguration in die jeweiligen Stein Module geladen. Somit erfolgt eine vollkomme Trennung zwischen der Steuerhardware und den jeweils angesteuerten Objekten. Dadurch kann jederzeit ein StEin Modul gegen ein anderes getauscht werden, ohne das deswegen die Objektzuordnung verloren geht.

OBJECT STATE (0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x10	0b00		4	ObjNId		Item					
0x01	0x10	0b1x		6	ObjNId		Item		State			

Vorbereitung für die Objekt Steuerung, Einführung mit dem StEin Modul.

Mit diesen Datagrammen kann der Objekt Status abgefragt bzw. beantwortet werden.

OBJECT COMMAND (0X12)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x12	0b00		4	ObjNId		Prop.					
0x01	0x12	0b1x		8	ObjNId		Prop.	Object Data				

Vorbereitung für die Objekt Steuerung, Einführung mit dem StEin Modul.

Mit diesen Datagrammen werden die ‚Zubehör Objekte' (StEin) gesteuert.
Das jeweilige Objekt ist an ein StEin Modul gebunden, da es ansonsten zu Kollisionen am CAN Bus kommen kann.

STEIN, OBJECT NID VERGABE:

Die Objekte der StEin Module werden durch die StEin Konfiguration festgelegt. Dies kann entweder durch entsprechende Befehle per CAN Bus oder per Script über ein File erfolgen.

ObjNId	Verwendung
0xE000	Broadcast Adresse für Gleisabschnitte
0x0001 … 0xE7FF	Gleisabschnitts Objekte
0xE800	Broadcast Adresse für Ein/Ausgänge
0xE801 … 0xEFFF	Ein/Ausgangs Objekte

STEIN, GLEISABSCHNITT

Gleisabschnitte haben (zumindest) folgende Eigenschaften (Property's)

Property	Verwendung	Data (DB5 … DB8)
0x01	Besetztmeldung	DB5: ‚0' = frei, ‚1' = besetzt DB6: frei DB7, 8: Abschnittsstrom in mA
0x02	HLU Limit	DB5: HLU Limit 0xFF: AUS 0x00: Halt 0x04, 0x08, 0x10, 0x14, …
0x10	RailCom Nachricht	DB5: MeldungsId DB6: SubId DB7, 8: Meldungs-Inhalt
0xF0	Fehler: Überstrom	DB5: 0xFF DB6: frei DB7, 8: Abschnittsstrom in mA

STEIN, EIN-/AUSGANG

Ein/Ausgangs Steuerung

Property	Verwendung	Data (DB5 … DB8)
0x01	Abfrage	Antwort: DB5: Schaltzustand
0x10	Schalten	DB5: gewünschter Schaltzustand
0xF0	Fehler: Überstrom	DB5: 0xFF DB6: frei DB7, 8: Ausgangsstrom in mA

FAHRZEUG CONTROL GROUP (0X02)

Die Command Group 0x02 fasst alle Fahrzeug Steuer Befehle zusammen und muss von allen Boostern und Fahrpulten implementiert werden. Diese Gruppe enthält jedoch KEINE Programmierbefehle (diese befinden sich in der ‚Config' Group)

FAHRZEUG STATE (0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x00	0b00		2	FahrzeugNId							
0x02	0x00	0b1x		8	FahrzeugNId		StateFlags		CtrlNId		CtrlTick	

Diese Abfrage sollte von jedem Steuersystem (Fahrpult/Computer) immer als erstes an die Zentrale gesendet werden.
Wenn M = 0b00, DLC = 2, dann wird der Status des Fahrzeuges mit 'NId' angefragt.
Wenn M = 0b11, DLC = 8, dann sendet die Zentrale die Status Antwort für das jeweilige Fahrzeug.
In der Antwort gibt die CtrlNId an welches Gerät das jeweilige Fahrzeug zuletzt gesteuert hat, der Wert CtrlTick enthält dabei die Anzahl der vergangenen ms seit dem letzten Steuerbefehl des CtrlNId Gerätes.
Hinweis:
Sollte ein Gerät für mehr als 65 Sec. (65536ms) keinen Steuerbefehl senden, so wird es zu einem nicht aktiven Steuergerät. In dem Falle sendet die Zentrale nur mehr die CtrlNId und als CtrlTick 0xFFFF.
Die Status Flags sind im Anhang aufgelistet.

FAHRZEUG MODE (0X01)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x01	0b00		2	FahrzeugNId							
0x02	0x01	0b01		4	FahrzeugNId		M1	M2				
0x02	0x01	0b11		4	FahrzeugNId		M1	M2				

Mit diesem Befehl kann der Fahrzeug Mode abgefragt bzw. geändert werden.

MODE 1 FLAGS

Bit	Info Command
0 .. 3	Speed Steps: 0: 14FS, 1: 27FS, 2: 28FS, 3: 128FS, 4: 1024FS
4 .. 7	Schienen Format: 0: unbekannt, 1: DCC, 2: MM2, 4:mfx

MODE 2 FLAGS

Bit	Info Command
0 .. 6	Max. Funktionen: Keine (0) bis 32Fx
7	Puls Fx

FAHRZEUG SPEED (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x02	0b00		2	FahrzeugNId							
0x02	0x02	0b01		6	FahrzeugNId		Speed		Div	0		
0x02	0x02	0b11		6	FahrzeugNId		Speed		Div	0		

Wenn M = 0b00, DLC = 2, dann wird die Geschwindigkeit der Lok mit 'NId' angefragt.
Wenn M = 0b01, DLC = 6, dann wird die Geschwindigkeit der Lok mit 'NId' wird auf den übergeben Wert gesetzt.
Wenn M = 0b11, DLC = 6, dann Antwortet die Lok mit 'NId' auf die Abfrage nach Ihrer Geschwindigkeit.
Geschwindigkeiten werden immer mit 10Bit gesendet, in den obersten 6Bit werden zusätzliche Flags gesendet (siehe Fahrzeug Status Flags). Die jeweilige Umrechnung ins Schienenformat erfolgt im TSE.
Der ‚Div' (Byte 5) Wert gibt einen Rangier Divisor an (Z.B.: /2 oder /4) für eine feinere Rangier Auflösung.

FAHRZEUG BASIS FUNKTION SCHALTEN (0X03)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x03	0b00		2	FahrzeugNId							
0x02	0x03	0b01		6	FahrzeugNId		Status für Fx0 bis Fx31					
0x02	0x03	0b11		6	FahrzeugNId		Status für Fx0 bis Fx31					

Wenn M = 0b00, DLC = 2, dann wird der Funktionsstatus der Lok mit 'NId' abgefragt
Wenn M = 0b01, DLC = 3..6, dann werden die Funktionen auf den jeweils angegebenen Status geschalten.
Wenn M = 0b11, DLC = 6, dann antwortet die Lok auf eine Status Abfrage.
Egal ob Befehl oder Antwort, enthalten die 32 Bits in den Datenbytes 3 bis 6 den jeweiligen Status der Lok Funktionen 0 bis 31. Wobei das höchste Bit im DB3 die Funktion 0 enthält und das niedrigste Bit in DB6 die Funktion 31 darstellt.

FAHRZEUG FUNKTION SCHALTEN (0X04)

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x04	0b00		4	FahrzeugNId		FxNr					
0x02	0x04	0b01		6	FahrzeugNId		FxNr		FxVal			
0x02	0x04	0b11		6	FahrzeugNId		FxNr		FxVal			

Wenn M = 0b00, DLC = 4, dann wird die Funktion der Lok mit 'NId' und der Funktion 'Nr.' abgefragt.
Wenn M = 0b01, DLC = 6, dann wird die Lokfunktion 'FxNr' der Lok 'NId' auf den angegebenen Wert gesetzt.
Wenn M = 0b11, DLC = 6, dann antwortet die Lok auf eine Funktionswert Abfrage.
Wobei FxVal = 0x00 immer 'Aus' bedeutet, FxVal ungleich 0x00 sind vom jeweiligen Lok Dekoder abhängig. Für 'normale' DCC und MM Lok Dekoder bedeutet dies einfach Funktion 'Ein'.
Die ‚FxNr' ist für diesen Befehl in mehrere Bereiche aufgeteilt:

Von FxNr	Bis FxNr	Beschreibung	Wertebereich
0	31	Die bekannten ‚normalen' Funktionen, die maximal Fx Nummer ist dabei vom jeweiligen Format abhängig.	Ein/Aus
256	512	256 Analog Funktionen	Ein/Aus
32768	65536	Binäry States	0 … 255

ZUG SUCHE (0X08)

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x04	0b00		2	FahrzeugNId							
0x02	0x04	0b01		4	FahrzeugNId		ZugNId					
0x02	0x04	0b11		4	FahrzeugNId		ZugNId					

Mit diesen Datagrammen kann abgefragt werden ob ein Fahrzeug zu einem Zug (Traktion) gehört.

ZUG ZUORDNUNG (0X08)

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x04	0b01		4	ZugNId		FahrzeugNId		Action			
0x02	0x04	0b11		4	ZugNId		FahrzeugNId		Action			

Diese Datagramme dienen dazu Zug Objekte zu ändern.

FAHRZEUG AKTIV (0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x10	0b00		2	Ziel-NId							
0x02	0x10	0xb01		4	Ziel-NId		Mode					

Jedes Steuergerät (Fahrpult/Computer) sollte diesen Befehl für ‚aktive' Fahrzeug etwa alle 500 … 1000ms aussenden. Dieses Kommando veranlasst das MX10 das das Fahrzeug zumindest in Priorität 1 verbleibt.

Wenn ein Steuergerät ein Fahrzeug aktiv steuern will, so muss es zuerst abfragen, ob das Fahrzeug nicht schon von einem anderen Gerät gesteuert wird. Wenn die Abfrage NICHT innerhalb von 500mS beantwortet wird, so wird das Fahrzeug von keinem anderen Gerät gesteuert und kann aktiviert werden.
Wenn die Abfrage beantwortet wird (Mode = 1), so ist das Fahrzeug auf einem anderen Gerät aktiv. Dies ist an sich eine reine Absicherung, da jedes Steuergerät sowieso periodisch für die von Ihm gesteuerten Fahrzeuge eine ‚Aktiv' Meldungen senden muss.
Wenn ein Steuergerät ein ‚aktives' Fahrzeug übernehmen will, so muss es das ‚Übernahme' Kommando senden (Mode = 0x10).

ACHTUNG!!! UNTERSCHEIDUNG STELLWERK/FAHRPULT

Dieser Befehl ist die WESENTLICHSTE Unterscheidung zwischen einer Stellwerks und einer Fahrpult Anwendung (egal ob am PC oder Tab, …).
Eine Fahrpult Anwendung MUSS die Übernahme/Übergabe Prozedur implementieren, da sonst andere Fahrpulte kommentarlos gegensteuern können.
Eine Stellwerk Software muss Fahrzeuge nicht zwangsweise ‚aktiv' melden, sofern sie mit manuellen Eingriffen umgehen kann.
Unter Anwendung der Übergabe/Übernahme Technik, kann immer nur jenes Fahrpult ein Fahrzeug steuern, für welches es den ‚aktiv Focus' hat.
Ohne der Übergabe/Übernahme Logik, inkl. der aktiv Meldung, kann jederzeit ein anderes Steuergerät Fahrstufen und/oder Funktionen ändern. Es ist dann Aufgabe der jeweiligen Software mit solchen Änderungen umzugehen. Abweichungen zwischen eigenen ‚SOLL' Zustand und gemeldeten ‚IST' Zustand müssen entsprechend abgebildet werden bzw. im weiteren Ablauf der Software berücksichtigt werden.

FAHRZEUG IM RÜCKHOLSPEICHER (0X11)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x11	0b11		8	Ziel-NId1		Ziel-NId2		Ziel-NId3		Ziel-NId4	

Durch dieses Kommando können Geräte dem MX10 mitteilen, welche Fahrzeuge sie im Rückholspeicher haben. Alle angeführten Fahrzeuge werden vom MX10 in die Prioritätsstufe ‚1' gesetzt und verbleiben dort bis alle Fahrzeug Befehle (Geschwindigkeit, Funktionen, …) zumindest 3x ausgesendet wurden. Danach werden sie im normalen Prioritätsschema weiter ausgesendet.

FAHRZEUG BIDI INFO (0X18)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x18	0b10		6/8	FahrzeugNId		Type		Info		[TrkCmd]	

Mit diesem Event meldet das MX10 diverse BiDi Rückmeldungen.

Type	Verwendung	DB5, DB6	DB7, DB8
0x0000	RailCom Statistik	Anzahl DCC Befehle an dieses Fahrzeug	Anzahl gültiger RailCom Nachrichten
0x0100	km/h Meldung	km/h Betrag	DCC Befehl (Fahren, Schalten, …)

In weiterer Folge sind auch Meldungen über Betriebsmittelverbrauch geplant.

FREE GROUP (0X03)

Diese Gruppe ist für Fremdanwendungen frei.

RAILWAY CONTROL SYSTEM (0X04)

Die Datagramme dieser Gruppe dienen der Kommunikation mit Stellwerken (Railway Control System, in weiterer Folge mit RCS abgekürzt).
Sie dienen in erster Linie einem verbesserten Zusammenspiel zwischen dem ZIMO System (MX10, MX32) und einer PC Stellwerks Software.
Diese Befehle sind erst nach einer primären Identifikation der Stellwerks Software frei geschalten.
Diese Identifikation läuft über die Interface Options (Grp=0x0A, Cmd=0x0A, Type=0x0000, 0x0001).

Ebenso ist es sinnvoll (aber nicht zwingend erforderlich), wenn ein PC Software Ihren Namen dem System mitteilt (Data Group 0x07, CMD=0x10, NId=0xC199 ... 0xC1FF).

RCS STATUS 0X00

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x00	0b01		4	Stellwerk		Status					
0x04	0x00	0b11		4	Stellwerk		Status					

Durch diese Datagramme kann eine Stellwerks-Software Ihren Status ans System weitergeben.
Jede Stellwerks-Software darf bis zu 16 ‚Stellwerke' haben. Aktuell sind nur zwei Stati definiert:
‚0' ➔ Stellwerk AUS, Offline
‚1' ➔ Stellwerk aktiv.

RCS POSITION (0X01)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x02	0b00		2	FahrzeugNId							
0x04	0x02	0b01		6	FahrzeugNId		ObjectNId		Port	Speed		
0x04	0x02	0b11		6	FahrzeugNId		ObjectNId		Port	Speed		

Mit diesen Datagrammen kann die Fahrzeug Position abgefragt bzw. gesetzt werden.
Üblicherweise wird ein Fahrpult (MX32), welches ja keine Ahnung vom Stellwerk hat, die Position von der Stellwerks Software abfragen. Eine Stellwerks Software kann die jeweilige Fahrzeugposition auch ungefragt an das System (MX10) senden. In jedem Falle sollte eine Stellwerks Software jede Änderung der Fahrzeugposition an das System melden.
Die Geschwindigkeit ist in Schritten zu je 5kmH anzugeben.

RCS LOCK (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x02	0b00		4	ObjectNId		Port	Type				
0x04	0x02	0b01		6 .. 8	ObjectNId		Port	Type	Data			
0x04	0x02	0b11		6 .. 8	ObjectNId		Port	Type	Data			

Durch diese Datagramme können Objekte ‚gesperrt' werden. Dies dient in erster Linie um bessere Betriebsabläufe zwischen der manuellen Steuerung und PC Programmen zu ermöglichen.
Type = 0x01: Bedeutet Sperre setzen
Type = 0x02: Hebt die Sperre wieder auf
Type = 0x00: ‚Sperre' unbekannt.
Data gibt an welche Fahrstraße oder durch welche WSpT diese Sperre gesetzt wird.

RCS SPEED LIMIT 0X04

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x04	0b00		2 [4]	FahrzeugNId		[Stellwerk]					
0x04	0x04	0b11		5	FahrzeugNId		Stellwerk		Speed			

Mit diesen Datengrammen werden die Stellwerks / Fahrstraßen Geschwindigkeits-Vorgaben behandelt.

Wenn ein Fahrpult (MX32) ein Fahrzeug ‚aktiviert' (Die Steuerung übernimmt) so muss es die erlaubte Höchstgeschwindigkeit vom Stellwerk abfragen. Eine Stellwerks Software hat bei Änderung der Höchstgeschwindigkeit dies an das System per Command zu übermitteln.

Die Fahrpult Firmware hat diese Informationen in sinnvoller Form anzuzeigen.
Die Geschwindigkeit ist in Schritten zu je 5kmH anzugeben.

RCS LOOK AHEAD 0X05

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x05	0b00		2 [4]	FahrzeugNId		[Stellwerk]					
0x04	0x05	0b10		8	FahrzeugNId		Stellwerk		Signal		Distanz	
0x04	0x05	0b11		8	FahrzeugNId		Stellwerk		Signal		Distanz	

Über die Abfrage kann ein Fahrpult das Stellwerk fragen, auf welches Signal das Fahrzeug zufährt.
Die Stellwerks Software kann diese Information entweder ‚ungefragt' als Event melden, oder per ACK eine entsprechende Abfrage beantworten.

RCS SHUNTING 0X08

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x04	0x08	0b00		2 [4]	FahrzeugNId		[Stellwerk]					
0x04	0x08	0b10		6	FahrzeugNId		Stellwerk					
0x04	0x08	0b11		6	FahrzeugNId		Stellwerk					

Diese Datagramme sind für einen ‚vorbildgerechten' Rangierbetrieb vorgesehen.
Der genaue Ablauf ist jedoch noch in Definitions-Phase.

CONFIG GROUP (0X05)

In der Config Group sind alle Telegramme zusammengefasst die der Einstellung von System Modulen dienen.

CONFIG MODUL POWER (0X01)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x01											

CONFIG MODUL VALUE (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x02	0b00			NID		Object		Item			
0x05	0x02	0b01			NID		Object		Item		Val	
0x05	0x02	0b11			Object		Item		Val			

Mit diesen Datagrammen können Parameter des Gerätes 'NId' abgefragt (M=0b00) und geändert (M=0b01) werden. Die Rückantwort bzw. Bestätigung erfolgt durch M=0b11.

Bei Abfrage bzw. Änderung ist unter 'NId' das jeweils angesprochen Gerät anzugeben. Die Parameter 'Num' , 'Idx' und 'Val' sind vom jeweiligen Gerät abhängig und sind der Beschreibung des Gerätes zu entnehmen.

TRACK CFG GROUP (0X06)

In dieser Gruppe sind alle Schienen Signal Funktionen zusammengefasst.
Insbesondere sind dies die verschiedenen Methoden um Dekoder zu Programmieren, Firmware Updates und Sound Ladefunktionen.

DEFINITION FÜR ‚CFG NUM':

Für die CfgNum wird ein 24Bit Wert verwendet, damit können theoretisch 16777215 unterschiedliche Konfigurationswerte adressiert werden.
Die Interpretation dieser Adresse ist vom jeweiligen Schienen Format abhängig.
Bei DCC werden die ersten 1024 Adressen 1:1 zu den CV Nummern gem. NMRA verwendet.
Bei DCC Config Adressen > 1024 werden diese als indexierte CV interpretiert. Dazu werden die CV's 31, 32 auf den Wert der übergeben Adresse Modulo 256 gesetzt.

TSE MODE (0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x00	0b00		2	NId							
0x06	0x00	0b01		4	NId		Port	Mode				
0x06	0x00	0b1x		4	NId		Port	Mode				

Durch M=0b00 kann der aktuelle Status der Track Signal Engine abgefragt werden.
Port gibt an welcher MX10 Port in den gewünschten Mode geschalten werden soll.
ACHTUNG:
Nicht jeder Port kann alle Betriebsmodi ausführen.
Durch Mode wird der jeweilige Betriebsmode festgelegt.

TSE INFO (0X01)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x01	0b01		7	FahrzeugNId		Cfg Num			State	Count	
0x06	0x01	0b11		7	FahrzeugNId		Cfg Num			State	Code	

Durch diese Datagramme meldet die Zentrale (MX10) die diversen Programmierzustände.
Während dem Lesen bzw. Schreiben enthält ‚State' den aktuellen Status und Count einen Zähler für den jeweiligen Status.
Wenn die jeweilige CV gelesen bzw. geschrieben ist, wird dies entweder durch das jeweilige ACK Datagramm gemeldet, bzw. bei Fehler durch ein ‚Info ACK' mit Angabe des Fehlercodes beendet.

State	Code	Beschreibung
0x00	0x00	Nicht verwendet
0x10	0x00	Programmier Mode ‚Init'
0x10	0x01	Service Mode sendet ‚Idle', wartet auf Programmierbefehl
0x10	0x02	Service Mode sendet ‚Reset': Programmierung ‚startet'
0x2n		Fortschrittsmeldung für Gleis ‚n'. Count gibt den jeweiligen Fortschritt an. Im Service Mode ist das der Bit Abfrage Counter
0x3n		Fortschrittsmeldung für Gleis ‚n', wenn im ‚Byte Verify' Mode.
0x8n		‚Busy': Programmiermode aktiv. Zentrale arbeitet Programmierbefehle ab und kann derzeit keine weiteren mehr annehmen.
0x9n		Stromverbrauch auf Gleis ‚n' beträgt ‚Val' mA, wahrscheinlich kein Dekoder angeschlossen
0xFn		Fehler Meldungen

TSE CLEAR (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x02	0b01		2	NId		FahrzeugNId					

Dieser Befehl (M=0b01) werden die im MX10 gespeicherten CV Werte des Fahrzeuges (FahrzeugNId) gelöscht.

TSE READ (0X04)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x04	0b01		7	NId		FahrzeugNId		Cfg Num			
0x06	0x04	0b11		8	NId		FahrzeugNId		Cfg Num			Val

Das Kommando (0b01) veranlasst, das die Zentrale (NId) einen ‚Config Read' Befehl an den Schienen Decoder (FahrzeugNId) sendet.
Solange der Lesebefehl ‚aktiv' ist, wird durch ‚TSE Info's der Fortschritt gemeldet.
Sobald die Zentrale den gewünschten Config Wert hat wird dies durch ein ‚TSE Read Ack' Telegramm gemeldet.

TSE WRITE (0X05)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x05	0b01		8	NId		FahrzeugNId		Cfg Num			Val
0x06	0x05	0b11		8	NId		FahrzeugNId		Cfg Num			Val

Das Kommando (0b01) veranlasst, das die Zentrale (NId) einen ‚Config Write' Befehl an den Schienen Decoder (FahrzeugNId) sendet.
Solange der Schreibbefehl ‚aktiv' ist, wird durch ‚TSE Info's der Fortschritt gemeldet.
Sobald die Zentrale den gewünschten Config Wert geschrieben wird dies durch ein ‚TSE Write Ack' Telegramm gemeldet.

ACHTUNG:
Nach Read/Write Befehlen sollte das jeweilige Steuergerät (MX32, PC, …) eine Pause von ca. 200mS einhalten.
Dies dient dazu, dass andere Schienenbefehle abgearbeitet werden können.

TSE UPD FIRMWARE (0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x10	0b01		8	NId		FahrzeugNId		Dekoder		Firmware	

Durch diesen Befehl (M=0b01) wird in den ‚Dekoder' die ‚Firmware' eingespielt. Der Wert Dekoder gibt dabei den Dekoder Typ an (MX64…). Durch Firmware kann die jeweilige Version der Firmware angegeben werden. Wenn dieser Wert ‚0' ist, so wird die jeweils aktuellste Version an den Dekoder gesendet.

TSE UPD SOUND, INIT1 (0X11)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x18	0b01		8	NId		FahrzeugNId		Sound Project			

Durch diesen Befehl (M=0b01) wird das ‚Sound Project' zum Laden in das Fahrzeug (Fahrzeug-NId) vorbereitet.

TSE UPD SOUND, INIT2 (0X12)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x19	0b01		8	NId		FahrzeugNId		Dekoder		Mode	

Hiermit wird der Sound Lade Prozess für den Dekoder im Mode gestartet.

DATA GROUP (0X07)

GROUP COUNT (0X00)

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x00	0b00		4	Ziel-NId		Group					
0x07	0x01	0b11		4	Group		Count					

Durch M = 0b00 kann ein Gerät abfragen ob das MX10 eine bestimmte Geräte Gruppe kennt. Das MX10 antwortet (M = 0b11) mit Gruppe und der ihm bekannten Anzahl an Geräten in der jeweiligen Gruppe.

GROUP CODES:

Group	
0x0000	Fahrzeuge
0x3000	Zubehör, DCC ‚simpel'
0x3200	DCC ‚eXtended' Zubehördekoder
0x5040	MX8 Module
0x5080	MX9 Module

ITEM LIST (0X02)

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x02	0b00		4	Ziel-NId		Index					
0x07	0x02	0b11		4	Index		NId					

Durch M = 0b00 kann ein Gerät die Objekt Liste im MX10 abfragen.
Das MX10 antwortet (M = 0b11) mit dem ‚nächsten' Objekt Index und der NId des Objektes. Dadurch kann ein Gerät eine Liste der dem MX10 bekannten Objekte aufbauen.

ITEM FLAG (0X04)

Die ‚Item Flag' Datagramme dienen der Abfrage von Flags. Die werden in erster Linie verwendet um zu übermitteln welche Daten für das Item gespeichert sind. Diese Flags können nur abgefragt und nicht gesetzt werden, da sich die jeweiligen Flags nach den tatsächlich vorhandenen Daten richten.

Grp	Cmd	D	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x02	0b00		4	SrcNId		NId					
0x07	0x02	0b11		6	SrcNId		NId		Flags			

FAHRZEUG FLAGS

Bit	Beschreibung
0	Name vorhanden
1	Gruppe vorhanden
2	Fahrzeug Bild vorhanden
3	Fahrzeug Tacho vorhanden
4	Fahrzeug Geschwindigkeit (V-Max.) vorhanden
5	Fahrzeug Geschwindigkeitstabelle vorhanden
6	
7	
8	Fahrzeug Funktions-Modi vorhanden
9	
10	
11	
12	Internal Use
13	Internal Use
14	Internal Use
15	Internal Use

DATA VALUE (0X05)

Mit den ‚Data Value' Datagrammen können Objekt (Data) Werte abgefragt bzw. gesetzt werden.

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x05	0b00		7	SrcNId		NId		Type			
0x07	0x05	0b01		8	NId		Type		Value			
0x07	0x05	0b11		8	NId		Type		Value			

DATA GROUP (0X06)

Mit den ‚Data Group' Datagrammen kann für jedes Fahrzeug, Zubehör, Gerät im System eine Gruppe abgefragt bzw. festgelegt werden.

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x06	0b00		7	SrcNId		NId					
0x07	0x06	0b01		8	NId		Grp					
0x07	0x06	0b11		8	NId		Grp					

Diese Gruppen dienen z.B.: bei Fahrzeugen der Such und Filter Funktion.

DATA NAME (0X10)

Mit den ‚Data Name' Datagrammen kann für jedes Fahrzeug, Zubehör, Gerät im System ein Name abgefragt bzw. festgelegt werden. Da Namen typischerweise aus mehreren Zeichen bestehen, gibt es dafür zwei Datagramm Sets:

VERSION FÜR DIE CAN INTERNE VERWENDUNG:

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x10	0b00		7	SrcNId		NId		Port	Size	Pos	
0x07	0x10	0b01		8	NId		Port	Pos	Z1	Z2	Z3	Z4
0x07	0x10	0b11		8	NId		Port	Pos	Z1	Z2	Z3	Z4

Durch M = 0b00 kann ein Gerät vom Gerät ‚SrcNId' den Namen vom Gerät ‚NId' abfragen. Mit ‚Size' gibt das anfragende Gerät wie viel Zeichen es maximal Platz hat. Mit ‚Pos' gibt es an ab welcher Position es die nächste Zeichengruppe haben will.
Durch M = 0b01 kann ein Gerät einen Namen für das Gerät ‚NId' festlegen.
Eine Abfrage wird durch M=0b11 beantwortet.

VERSION FÜR DIE PC – SCHNITTSTELLE (USB UND ETHERNET):

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB …
0x07	0x10	0b00		8	SrcNId		NId		Port			
0x07	0x10	0b01		7 …	NId		Port				Z1 …. Z[x]	
0x07	0x10	0b11		7 …	NId		Port				Z1 …. Z[x]	

Da die PC Schnittstelle bis zu 255 Zeichen in einem Datagramm senden und empfangen kann, kann ein PC den Namen in einem Datagramm abfragen. Die Size darf bis zu 24 Zeichen sein.
Sofern das PC Interface die ‚Remote Control' Technik zur Verfügung stellt, muss es über diese Datagramme auch die Fahrstraßen Namen zur Verfügung stellen.

ITEM IMAGE CONFIG (0X12)

NICHT FREIGEGEBEN, WIRD ÜBERARBEITET

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x12	0b00		4	Ziel-NId		Type					
0x07	0x12	0b01		6	Ziel-NId		Type		ImageId			
0x07	0x12	0b11		6	Ziel-NId		Type		ImageId			
0x07	0x12	0b01		8	Ziel-NId		Type		ImageCRC32			
0x07	0x12	0b11		8	Ziel-NId		Type		ImageCRC32			

Durch M = 0b00 kann ein Gerät die der ‚Ziel-NId' zugeordneten Images abrufen.
Durch M = 0b01 wird der ‚Ziel-NId' ein Image vom jeweiligen Typ zugeordnet.
Mit M = 0b11 beantwortet das Gerät eine ImageId Abfrage.
Das MSB (Bit 15) vom Type gibt an ob die Image Zuordnung über die ImageId oder über die Image CRC32 erfolgen soll.

IMAGE TYPES

Type	Beschreibung
0x1010	Fahrzeug Bild ‚klein'
0x1011	Fahrzeug Bild ‚groß'
0x1012	Fahrzeug Bild ‚2'
0x1020	Fahrzeug Instrument: Tacho
0x1028	Fahrzeug Instrument: Bremsbalken
0x1030	Fahrzeug Instrument:
0x2000 … 0x20FF	Fahrzeug Funktionsicons (F000 … F255) Hinweis: Je nach Schienenformat sind nur 1, 4, 8, 28, 32, … Icons erlaubt/möglich

ITEM FX MODE (0X14)

NICHT FREIGEGEBEN, WIRD ÜBERARBEITET

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x12	0b00		3	Src-NId		Ziel-NId		FGrp			
0x07	0x12	0b01		4 .. 7	Ziel-NId		FGrp	Function Mode				
0x07	0x12	0b11		4 .. 7	Ziel-NId		FGrp	Function Mode				

Diese Datagramme dienen der Abfrage und dem setzen von Funktionseigenschaften.
In der Abfrage wird durch ‚SrcNId' angegeben, welches Gerät antworten soll (Üblicherweise die Zentrale).
Die ‚Ziel-NId' gibt das Gerät an, dessen Funktionsmode abgefragt bzw. geändert werden soll.
In jeder Funktionsgruppe können die Modes für bis zu 16 Funktionen übertragen werden.

FGrp	
0	Funktionsmode für F00, … F15
1	Funktionsmode für F16, … F31

Funktionsmode Bitdefinition:

Bit	
0b00	‚Normale' Ein/Aus Funktion
0b01	Moment Funktion
0b10	Sonder Funktionen, Definition per Fx Config Command
0b11	Undefiniert!!!, Don't Use

ITEM FX CONFIG (0X15)

NICHT FREIGEGEBEN, WIRD ÜBERARBEITET

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x12	0b00		4	Ziel-NId		FxNum					
0x07	0x12	0b01		8	Ziel-NId		FxNum		Type		Data	
0x07	0x12	0b11		8	Ziel-NId		FxNum		Type		Data	

Durch M = 0b00 kann ein Gerät die der ‚Ziel-NId' zugeordneten Funktionseigenschaften abrufen. Mit M = 0b01 kann ein Gerät die Funktionseigenschaften der Funktion ‚FxNum' für die ‚Ziel-NId' ändern. Mit 0b11 beantwortet die Zentrale die Abfrage nach den Funktionseigenschaften.

INFO GROUP (0X08)

In der Info Group sind diverse Informations- Abfragen und Meldungen zusammengefasst.

MODUL POWER INFO (0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x00	0b00		3	Ziel-NId		Port					
0x08	0x00	0b10		8	Port	0	Status		Track U		Track I	
0x08	0x00	0b11		8	Port	0	Status		Track U		Track I	

Port:

Bit	
0 … 3	Port Nummer: 0=Schiene 1, 1=Schiene 2, 2=Booster
4	Frei
5	Schiene 2: Unterspannung
6	Schiene 1: Unterspannung
7	Netzteil Überstrom/Leistung

Status:

Bit	
0	ZACK
1	RailCom
2	mfx
3	Frei
4	Sammelstopp
5	Anmeldung ‚anstehend‘
6 ... 7	Frei
8 … 11	Betriebsmode: 0 = Normal, 1 = Serv. Prog, 3 = Update
12 ... 15	Strom Status: 0 = Normal, 1 = Konstant Strom, 2 = Überstrom, … ,15 = AUS

NETWORK GROUP (0X0A)

In der Network Group sind all jene Telegramme zusammengefasst, welche sich mit dem Network Management befassen.

PING (0X00)

Grp	Cmd	M	NId	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x00	0b10		8	Master-UID				Type		Session	

Die primäre Zentrale versendet diesen Befehl etwa alle 500ms, zumindest jedoch jede Sekunde.
Master-UID: UID der Zentrale
Type: Art der Zentrale, siehe Tabelle
Session: Session Nummer

Anhand dieses Befehls sollen die angeschlossenen Module erkennen, ob sie immer noch mit der Ihnen bekannten Zentrale verbunden sind. Dabei muss auch die Session Nummer geprüft werden. Diese Session Nummer wird von der Zentrale bei jeder UID Änderung inkrementiert. Dies erfolgt z.B.: Wenn die Zentrale ein neues Objekt in Ihre Objektliste aufnimmt, oder wenn ein vorhandenes aus dieser Liste gelöscht wird.
Erkennt ein Modul, das es mit einer ‚unbekannten' Zentrale verbunden ist, so muss es einen Anmeldevorgang initiieren.

LOGIN [1] (0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x02	0b00	UID-Low	0								
0x0A	0x02	0b11		2	UID-Low							

Durch dieses Telegramm erfolgt der erste Schritt der Modul Anmeldung.
Das jeweilige Modul sendet dabei in der ID die unteren 16Bit seiner UID.
Die Zentrale beantwortet das Telegramm mit M = 0b11 und der 'erkannten' UID-Low.
Sollten mehrere Geräte mit unterschiedlicher UID-Low einen Anmeldeversuch machen, so gewinnt durch die CAN Arbitrierung immer die niedrigste.

LOGIN [2] (0X03)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x03	0b00	UID-High	0								
0x0A	0x03	0b11		6	'Angemeldete' UID				NId			

Durch dieses Datagramm geben alle durch Login [2] ausgewählten Geräte ihr UID High bekannt.

LOGIN [3] (0X04)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8

LOGIN ERROR (0X05)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x05	0b10		1	1							

Mit diesem Telegramm meldet die Zentrale einen Anmeldefehler.
Die genauere Ursache steckt im Db1 (Muss aber noch definiert werden).

LOGOFF / PORT CLOSE (0X07)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x07	0b01		2	NId							

Durch dieses Telegramm kann sich ein Gerät von einer Zentrale abmelden.
Sofern dies eine PC Software sendet, wird dadurch auch automatisch das jeweilige Kommunikationsport (USB oder Ethernet) geschlossen. Das jeweilige Gerät muss zur Wiederaufnahme der Verbindung wieder die jeweiligen Initialschritte abbarbeiten.
Als (Ziel-) NId ist dabei die NId jenes Gerätes anzugeben, von welchem sich der Schnittstellen Benutzer (PC Software) abmelden will.

MODUL INFO (0X08)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x08	0b00		4	NID		Type					
0x0A	0x08	0b01		8	NID		Type		Info			
0x0A	0x08	0b11		6	Type		Info					

Über die Modul Info Datagramme können diverse Informationen abgefragt werden.
ACHTUNG:
Die meisten Informationen sind ‚Read Only' Informationen, können also NICHT per Command geändert werden. In der nachfolgenden Tabelle ist angegeben, welche Informationen R/W sind und das jeweilige Format.

INFO TYPES

Type	Verwendung	R/W	Format
1	Hardware Version	RO	
2	Software Version	RO	
3	Software Build Date	RO	
4	Software Build Time	RO	
5	Realtime Clock Date	RW	
6	Realtime Clock Time	RW	
7 …	Frei		

INTERFACE OPTION (0X0A)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x0A	0b00		4	NID		Type					
0x0A	0x0A	0b01		8	NID		Type		Value			
0x0A	0x0A	0b11		8	NID		Type		Value			

Durch diese Datagramme kann eine PC Software diverse Kommunikationsoptionen abfragen bzw. einstellen.
Derzeit ist nur Type=0x0000 verwendet/definiert.

INFO VALUES FÜR TYPE 0X0000

Type	Value Bits	Verwendung
0x0000	0	‚Lange' Datagramme Ein/Aus (0/1)
	1	Sperr Logik Ein/Aus
	2	Fahrstraßen (PC Software stellt Fahrstraßen Informationen zur Verfügung)
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	MX9 ‚0' Zugnummern senden
	17	

INFO VALUES FÜR TYPE 0X0001

Type	Value	Verwendung
0x0001	0x0000	ZIMO Intern
	0x0010	Kennung für ESTWGJ
	0x0020	Kennung für STP
	0x0030	Kennung für TrainController
	0x0031	Kennung für TrainProgrammer

RF CONNECT MAKE/KILL (0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x10	0b01		8	Task	Chn.	Cu-NId		Cu-UId			
0x0A	0x10	0b11		8	Task	Chn.	Cu-NId		Cu-UId			

Durch das Command wird eine Funkverbindung zu der durch ‚CU-NId' angegebene Zentrale aufgebaut bzw. wieder abgebaut. Das Funkmodul bestätigt den Empfang der Nachricht durch ‚ACK'.

RF CONTROL TASKS

Task	
0x1x	Verbindung Aufbauen ‚x' gibt das Land an: ‚1' : Amerika ‚2': Europa ‚F' : Alle restlichen Länder
0xFx	Verbindung abbauen
Chn. (Channel)	Funkkanal

RF STATE (0X11)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x11	0b00		4	Task		Wert					
0x0A	0x11	0b1x		4	Task		Wert					

Status Abfrage und Meldungen des Funkmodules.
Während des Auf/Ab Baues meldet das Funkmodul periodisch den jeweiligen Schritt/Zustand des Auf und Abbaues der Funkverbindung (zumindest. alle 500mS).
Während des aktiven Betriebes meldet es die jeweilige Feldstärke.
Wenn die periodischen Meldungen ausbleiben, sendet das Fahrpult eine Abfrage, wird diese auch nicht beantwortet, so geht das Fahrpult von einem Defekt des Funkmodules aus.

RF STATE TASKS

Task	Beschreibung	Wert
0x0001	Zwischen Meldungen während dem Verbindungsaufbau	0 … 255
0x0002	Feldstärke	0 … 255
0x0003	Akku Spannung in mV	0 (Entladen) 3600 … 4200mV

FILE CONTROL (0X0E)

Die File Control Group (0x0E) arbeitet eng mit der File Transfer Group (0x0F) zusammen. Im wesentlichen werden die Transfers durch die Befehle dieser Gruppe gesteuert.

FILE CONTROL: STATE (0X0E, 0X00)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x00	0b01		2	File-Id							
0x0E	0x00	0b01		3	File-Id		State					
0x0E	0x00	0b1x		3	File-Id		State					

Durch 0b01 kann der jeweilige Status der Transfers File-Id abgefragt werden.

FILE CONTROL: OPEN (0X0E, 0X02)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x02	0b01		8	Ziel-Id		File-Id		Size			
0x0E	0x02	0b11		8	File-Id		Status		Size			

Mit M=0b01 öffnet das 'Sende' Gerät ein File am 'Ziel' Gerät mit der angegeben File-Id.
Das Ziel Gerät bestätigt das öffnen durch M=0b11 und der möglichen Size.
Wenn Status != 0 so liegt ein Fehler vor.

FILE CONTROL: CLOSE (0X0E, 0X03)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x02	0b01		8	Ziel-Id		File-Id		Size			
0x0E	0x02	0b11		8	File-Id		Status		Size			

FILE CONTROL: INFO (0X0E, 0X03)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x02	0b01		8	Ziel-Id		File-Id		Info		Data	
0x0E	0x02	0b11		8	File-Id		Info		Data			

FILE CONTROL: BLOCK START (0X0E, 0X10)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x02	0b01		8	Ziel-Id		File-Id		BlockNum			
0x0E	0x02	0b11		8	File-Id		BlockNum					

FILE CONTROL: BLOCK CRC (0X0E, 0X11)

Grp	Cmd	M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x02	0b01		8	Ziel-Id		File-Id		BlockNum			
0x0E	0x02	0b11		8	File-Id		BlockNum		Block CRC			

FILE TRANSFER GROUP (0X0F)

In der File Transfer Group gibt es keine Commands, da das Command Field als Counter verwendet wird.

FILE TRANSFER: WRITE (0X0F)

Grp	Cmd+M	Id	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0F	Counter		8	File-Id		6 Daten Bytes					

Durch dieses CAN Telegramm werden jeweils 6 Datenbytes übertragen.
Jeder 'Transfer' Block darf bis zu 256x6 = 1536 Bytes haben. Nach jedem Block muss eine Checksumme gesendet werden.

FUNKTIONELLE EIGENSCHAFTEN

ABLAUF CAN GERÄTE ANMELDUNG

Damit am CAN Bus dynamisch Geräte angemeldet werden können, ist folgender Ablauf vorgesehen:

Schritt	
1	Jedes Gerät erkennt am 'ItsMe' Broadcast, ob es immer noch an 'seiner' bekannten Zentrale hängt.
2	'Neues' Gerät sendet 'Anmeldung', in den 16Bit der Bus-Id überträgt es die unteren 16 Bit seiner UId. Dabei ist DLC = 0, somit kann es zu keinem Datencrash kommen. Cmd=0x02
3	Die Zentrale antwortet mit: Anmeldung erkannt und den 'gesendeten' unteren 16Bit der UId (Cmd = 0x02)
4	Nun sendet das Gerät seine oberen 16 UId Bits, ebenfalls mit DLC=0, Cmd = 0x03
5	Die Zentrale antwortet mit: Anmeldung erkannt und den 'gesendeten' oberen 16Bit der UID (Cmd = 0x03) In DB 1, 2, 3, 4 wird die erkannte UID gesendet in DB 5, 6 wird die zugewiesene NID übertragen.
6	Gerät antwortet mit zugewiesener Bus-Id, Status, Version
7	Zentrale 'aktiviert/erlaubt' allgemeinen Bus Zugriff

ABLAUF FAHRZEUG ‚AKTIVIEREN'

Mit dem im folgenden beschriebenen Ablauf aktiviert ein Fahrpult bzw. sonstiges Steuergerät (z.B.: Computer) ein Fahrzeug um dieses steuern.

Schritt	
1	Abfrage des Fahrzeug Status
2	Antwort abwarten, max. 500ms. Kommt in dieser Zeit keine Antwort, so kann das MX10 das gewünschte Fahrzeug nicht aktivieren. Ein Steuern des Fahrzeuges ist somit unmöglich.
3	Abfrage des Fahrzeug Modes.
4	Antwort abwarten, max. 500ms. Normalerweise kommt die Antwort in weniger als 10ms. Sollte die Antwort nicht innerhalb von 500ms so liegt ein Fehler vor.
5a	Ab hier kann das Fahrzeug im vollem Umfang gesteuert werden. Sämtliche Fahr, Schalt und POM Befehle können genutzt werden.
5b	Ebenso können ab hier alle ‚Daten' des Fahrzeuges verändert werden. Damit sind in erster Linie die GUI Daten gemeint (Name, Bild, Fx Icons, …)

ABLAUF MX8, MX9

Das MX10 verwaltet für diese Module eigene Objekte. Grundsätzlich werden Abfragen von diesem Objekt Speicher beantwortet bzw. Befehle in diesen Objektspeicher eingetragen.
Jedes dieser Objekte bildet gleichzeitig eine Autonom laufende Task Engine. Diese sendet bei Daten / Zustands Änderungen (durch Befehle) die passenden Befehle an die Module (MX8/MX9). Umgekehrt werden alle Informationen von diesen Modulen ebenfalls im jeweiligen Objektspeicher eingetragen und danach an den PC weitergeleitet.
Diese Logik hat sowohl Vorteile wie auch Nachteile:
Vorteile:

- Im laufenden Betrieb kann die volle Bandbreite des PC Interfaces genutzt werden.
- Die fortlaufende Überwachung der Module wird vom MX10 übernommen.
- Einheitliche Kommando Logik, egal ob es sich um ein MX8, MX9 oder später StEin handelt.

Nachteile:

- Unmittelbar nach dem Hochfahren des MX10 sind alle Daten ‚Invalid'

ABLAUF STEIN

Der im folgenden beschriebene Ablauf soll die Startsequenz des StEin Modules darstellen.

Schritt	

OBJECT CONTROL / STEIN:

Mit dem ZIMO StEin Modul wird eine neue Objekt orientierte Steuerlogik eingeführt.

WAS IST EIN OBJEKT

Objekte stellen die diversen steuerbaren Elemente einer Modellbahn Anlage dar.
Dies können Weichen, Signale, Bahnübergänge, aber auch simple Straßenbeleuchtungen, oder andere einfache Dinge sein.

Jedes Objekt hat dabei eine Systemweit eindeutige Kennung, welcher eine ‚Kurznummer' und ein Name zugeordnet werden kann. Die Systemweit eindeutige Kennung wird automatisch vergeben.

Die ‚Kurznummern' dürfen dabei im Bereich 1 … 65535 liegen. Die Kurznummer ‚0' ist für Broadcasts reserviert (Z.B.: Status/Existenz Abfrage an ALLE).

Die Objekte werden nach Objektarten gruppiert, momentan sind folgende Objektarten vorgesehen:

- Gleisabschnitte
- Weichen
- Signale
- Lampen/Beleuchtungen
- …..

Diese Liste IST unvollständig und wird je nach Bedarf erweitert.
Die oben genannten Kurznummern müssen je Objektart eindeutig sein, die gleiche Kurznummer darf jedoch in unterschiedlichen Objektarten mehrfach genutzt werden.
Beispiel:
Eine Weiche mit Nummer ‚1' und ein Signal mit Nummer ‚1' ist ZULÄSSIG.
Zwei Gleisabschnitte mit Nummer ‚5' hingegen NICHT.

Objekte können auch ‚Slave' Objekte enthalten, so kann z:b: ein Gleisabschnitt die jeweiligen Signale als ‚Slave' Objekte ansteuern. Oder eine Weiche kann das jeweilige Signal entsprechen stellen (Gerade = Volle Fahrt, Abzweig = Langsamfahrt). Natürlich könnte ein Signal so auch direkten Einfluss auf die HLU Information nehmen. Bei ‚grün' kann der Abschnitt so automatisch auf ‚Fahrt' gestellt werden, bei ‚rot' auf Halt.

TABELLEN:

PORT ZUORDNUNGSTABELLE

Gerät	Port	Beschreibung
MX10	0	Schiene 1
MX10	1	Schiene 2
MX10	3	Booster Ausgang
MX10	254	Netzteil
MX10	255	Alle Schienen AUS

SYSTEM MODE

Gerät	Status	Beschreibung
MX10	0b0000,0000	Normalbetrieb
MX10	0b0000,0001	Sammelstopp
MX10	0b0000,0010	Service Prog.
MX10	0b0000,0100	Dekoder Update
MX10	0b0001,0000	Konstant Strom
MX10	0b0010,0000	Abgeschalten
MX10	0b0100,0000	Unterspannung
MX10	0b1000,0000	Überstrom

FEEDBACK DATA TYPES:

Type	
0x00	Keine Daten, bzw. allgemeine Abfrage
0x01	Adresse, In erster Linie für MX9 AZN Adressen Erkennung. Sollte aber auch von RailCom Detektoren bei eindeutig erkannter Adresse verwendet werden.
0x10	BiDi Channel 1, Datagramm 0x00
0x11	BiDi Channel 1, Datagramm 0x01
0x20	BiDi Channel 2, Datagramm 0x00
0x21	BiDi Channel 2, Datagramm 0x01
0x22	BiDi Channel 2, Datagramm 0x02
0x23	BiDi Channel 2, Datagramm 0x03
0x24	BiDi Channel 2, Datagramm 0x04
0x25	BiDi Channel 2, Datagramm 0x05
0x26	BiDi Channel 2, Datagramm 0x06
0x27	BiDi Channel 2, Datagramm 0x07
0x28	BiDi Channel 2, Datagramm 0x08
0x29	BiDi Channel 2, Datagramm 0x09
0x2A	BiDi Channel 2, Datagramm 0x0A
0x2B	BiDi Channel 2, Datagramm 0x0B
0x2C	BiDi Channel 2, Datagramm 0x0C
0x2D	BiDi Channel 2, Datagramm 0x0D
0x2E	BiDi Channel 2, Datagramm 0x0E
0x2F	BiDi Channel 2, Datagramm 0x0F

ANHANG:

SYSTEM STATUS FLAGS

Status Zuordnungstabelle:

Gerät	Status	Beschreibung
MX10	0x00	‚Normalbetrieb'
MX10	0x80 (10000000)	Unterspannung
MX10	0x40 (01000000)	Überstrom
MX10	0x20 (00100000)	Strom ‚AUS'
MX10	0x08 (00010000)	Sammelstop
MX10	0x04 (00001000)	Prog./Update Mode
MX10	0x02 (00000010)	RailCom ‚vorhanden'
MX10	0x01 (00000001)	ZACK ‚vorhanden'

FAHRZEUG STATUS FLAGS

In diversen Fahrzeug Befehlen werden auch die Status Flags für das jeweilige Fahrzeug gesendet. Diese Flags sind in zwei Gruppen eingeteilt.
Die oberen 6Bit sind auch in den Geschwindigkeitsbefehlen und Meldungen enthalten, die unteren 10Bit werden je nach Befehl entweder für die Geschwindigkeit oder für ergänzende Informationen genutzt.

Bit	Speed Command	Info Command
0		Fahrzeug unbekannt, Zentrale hat keine Daten für dieses Fahrzeug
1		Name ‘vorhanden’
2		Image ‘vorhanden’
3		
4		
5		
6		
7		
8		RailCom
9		ZACK
10	Ist Richtung	Ist Richtung
11	Soll Richtung	Soll Richtung
12		
13	Rangier Mode	Rangier Mode
14	Manual Mode	Manual Mode
15	Fahrzeug befindet sich derzeit in Notstopp	Fahrzeug befindet sich derzeit in Notstopp

SPEZIELLE NID‘S

NId Min.	NId Max	Anzahl	
0x7000	0x7000	1	Namen für Gruppen
0x7100	0x71FF	256	Hersteller Namen

FIXED FILE HANDLES (ID'S)

Für einige System Daten sind 'fixe' File Handles vorgesehen.

File-Id	Verwendung
0	
1	System Update
2	
3	
4	Sprachen
5	System Icons
6	Fahrzeug Instrumente
7	Fahrzeug Bilder
8	
9	Fahrzeug Datenbank
10	Zubehör Datenbank
11	
12	
13	
14	
15	

EINGETRAGENE MARKENZEICHEN

mfx	Märklin
ZIMO	ZIMO
HLU	ZIMO
DCC	NMRA

KONFIGURATIONSPARAMETER ALLGEMEIN

Die folgenden Parameter sollten von jedem Modul implementiert werden.

HARDWARE/FIRMWARE VERSION

Object	Item	Beschreibung	Min.	Max.
0x0101	1	Hardware Version Low	RO	RO
	2	Hardware Version High	RO	RO
0x0102	1	Firmware Version Low	RO	RO
	2	Firmware Version High	RO	RO
	3	Firmware Version Year	RO	RO
	4	Firmware Version Month	RO	RO
	5	Firmware Version Day	RO	RO
	6	Firmware Version Hour	RO	RO
	7	Firmware Version Min	RO	RO
	8	Firmware Version Sec.	RO	RO
0x0103	1	Real Time Clock Year		
	2	Real Time Clock Month		
	3	Real Time Clock Day		
	4	Real Time Clock Hour		
	5	Real Time Clock Min		
	6	Real Time Clock Sec		

KONFIGURATIONSPARAMETER FÜR MX10

Die folgende Liste enthält die Konfigurationsparameter für die MX10 Zentrale.
Die angegeben Werte dienen als Parameter für die Config Modul Value Datagramme.

SPANNUNG UND STROM EINSTELLUNGEN

Object	Item	Beschreibung	Min.	Max.
0x0001	1	Netzteil Spannung (RO), Einheit: mV	Aktueller Messwert	
	2	Netzteil Strom, Einheit: mA		
	3	Netzteil Leistung, Einheit: W		
0x0020	1	Schiene 1: Aktuelle Spannung (RO), Einheit: mV	0mV	32767mV
	2	Schiene 1: Aktueller Strom (RO), Einheit: mA	0mA	32767mA
	4	Schiene 1: Power Mode		
	5	Schiene 1: 'Soll' Spannung	8000mV	32767mV
	6	Schiene 1: 'Soll' Strom	1000mA	16383mA
	7	Schiene 1: Delay 1 (Überstrom Abschaltzeit)	0ms	5000ms
	0x10	Schiene 1: Flanken Strom	2000mA	8000mA
	0x11	Schiene 1: Flanken Zeit	0ms	5000ms
	0x20	Schiene 1: Überstromzeit 10%	0Sec.	32767Sec. (ca. 10,5 Std.)
	0x21	Schiene 1: Überstromzeit 25%	0Sec.	32767Sec.
	0x22	Schiene 1: Überstromzeit 50%	0Sec.	32767Sec.
0x0028	1	Schiene 2: Aktuelle Spannung (RO), Einheit: mV	0mV	32767mA
	2	Schiene 2: Aktueller Strom (RO), Einheit: mA	0mV	32767mA
	4	Schiene 2: Power Mode		
	5	Schiene 2: 'Soll' Spannung	8000mV	32767mV
	6	Schiene 2: 'Soll' Strom	1000mA	16383mA
	7	Schiene 2: Delay 1 (Überstrom Abschaltzeit)	0ms	5000ms
	0x10	Schiene 2: Flanken Strom	2000mA	8000mA
	0x11	Schiene 2: Flanken Zeit	0ms	5000ms
	0x20	Schiene 2: Überstromzeit 10%	0Sec.	32767Sec.
	0x21	Schiene 2: Überstromzeit 25%	0Sec.	32767Sec.
	0x22	Schiene 2: Überstromzeit 50%	0Sec.	32767Sec.

TRACK SIGNAL ENGINE SETTINGS

Object	Item	Beschreibung	Min.	Max.
0x0030	0x01	Max. DCC Funktionen	Bit 0 ... 7: Anzahl Bit 8: Pulskette	
	0x02	Max. MM2 Funktionen	4	16
	0x03	ZIMO Setting (MN, HLU, ZACK)		

TRACK SIGNAL TIMING

Alle Signal Timings werden in µS angegeben bzw. übertragen.
Somit sind Zeiten zwischen 0µS und 65535µS=65.5ms möglich, langsamere Signale sind derzeit nicht bekannt.
Die interne Timer Auflösung beträgt derzeit 100nS, durch diese 10-fach höhere Auflösung bleibt der Jitter immer deutlich unter diesen 100nS.

Object	Item	Beschreibung	Min.	Max.
0x0034	0x01	DCC '0' Bit, Teil A (µS)		
	0x02	DCC '0' Bit, Teil B (µS)		
	0x03	DCC '1' Bit, Teil A (µS)		
	0x04	DCC '1' Bit, Teil B (µS)		
	0x05	DCC Analog (µS)		
	0x06	DCC BiDi Lücke (µS)		
	0x07	ZIMO ‘0’, Teil A (µS)		
	0x08	ZIMO ‘0’, Teil B (µS)		
	0x10	MM2 Fahrzeug '0' Bit, Teil A (µS)		
	0x11	MM2 Fahrzeug '0' Bit, Teil B (µS)		
	0x12	MM2 Fahrzeug '1' Bit, Teil A (µS)		
	0x13	MM2 Fahrzeug '1' Bit, Teil B (µS)		
	0x14	MM2 Zubehör '0' Bit, Teil A (µS)		
	0x15	MM2 Zubehör '0' Bit, Teil B (µS)		
	0x16	MM2 Zubehör '1' Bit, Teil A (µS)		
	0x17	MM2 Zubehör '1' Bit, Teil B (µS)		
	0x18	MM2, Pause 1 (µS)		
	0x19	MM2, Pause 2 (µS)		
	0x1A	MM2, Pause 3 (µS)		
	0x20	Mfx (µS)		
	0x80	DecUp ‚0‘ Bit, Teil A (µS)		
	0x81	DecUp ‚0‘ Bit, Teil B (µS)		
	0x82	DecUp ‚1‘ Bit, Teil A (µS)		
	0x83	DecUp ‚1‘ Bit, Teil B (µS)		

TRACK SIGNAL ENGINE, SERVICE MODE PROGRAMMING

Object	Item	Beschreibung	Min.	Max.
0x0038	0x01	DCC Service Mode: Power Off Dauer VOR Programming (ms)		
	0x02	DCC Service Mode: Power Off Dauer NACH Programming (ms)		
		DCC Service Mode: Preambel Bits		
		DCC Service Mode: Idle Packets		
		DCC Service Mode: Reset Packets		
		DCC Service Mode: Read Methode		

TRACK SIGNAL ENGINE (SCHIENE 1)

Object	Item	Beschreibung	Min.	Max.
0x0034	1	Schiene 1: Signal Mode (DCC, MM2, ...)	Bit 0: DCC Fahrzeuge Bit 1: MM2 Fahrzeuge Bit 3: mfx Bit 4 .. 7: frei Bit 8: DCC Zubehör Bit 9: MM2 Zubehör Bit 10 .. 15: frei	
	2	Schiene 1: Update Mode	Bit 12: Svc Prog Bit 13: Svc Update Bit 14: Main Upd Bit 15: Snd Update	
	3	Schiene 1: Feedback	Bit 0..3: RailCom Bit 4..5: ZACK Bit 6..7: mfx	

TRACK SIGNAL ENGINE (SCHIENE 2)

Object	Item	Beschreibung	Min.	Max.
0x0038	1	Schiene 2: Signal Mode (DCC, MM2, ...)	Bit 0: DCC Fahrzeuge Bit 1: MM2 Fahrzeuge Bit 3: mfx Fahrzeuge Bit 4 .. 7: frei Bit 8: DCC Zubehör Bit 9: MM2 Zubehör Bit 10 .. 15: frei	
	2	Schiene 1: Feedback	Bit 0..3: RailCom Bit 4..5: ZACK Bit 6..7: mfx	
	3	Schiene 1: Update	Bit 0: DecUp Methode Bit 1: Fast Svc Update Bit 2: Sound Update 1 Bit 3: Sound Update 2	

NETWORK CONFIG

Object	Item	Beschreibung	Min.	Max.
0x0220		Lokal Network		
	1	CAN Bus 1	1 = ZIMO 125kBaud 2 = ZIMO 256kBaud 3=ZIMO 512kBaud 4=Märklin CAN	
	2	CAN Bus 2		
	3	X-PressNet 1	Bit 0, 1 = 0: Aus Bit 0, 1 = 1: Slave Bit 0, 1 = 2: Master Bit 2, 3 = 0: 62.5kBaud Bit 2, 3 = 1: 125kBaud Bit 2, 3 = 2: 250kBaud Bit 4 .. 8 = Max. Adresse	
	4	X-PressNet 2		
	5	Loco Net		
	0x10	X-PressNet 1 Devices 0 ... 15	Je ein Bit für ein Device	
	0x11	X-PressNet 1 Devices 16 ... 31		
	0x12	X-PressNet 2 Devices 0 ... 15		
	0x13	X-PressNet 2 Devices 16 ... 31		
0x0228		PC Interface		
	1	USB Device Interface Mode	0 = Autodetect 1 =	
	0x10	IP Adresse Byte 1, 2		
	0x11	IP Adresse Byte 3, 4		
	0x12	UDP Port		
	0x13	DHCP Mode		

GPIO CONTROL

Object	Item	Beschreibung	Min.	Max.
0x0200	1	GPIO 1 Mode		
....		.....		
0x020F	1	GPIO 16 Mode		

KONFIGURATIONSPARAMETER FÜR STEIN

Die folgende Liste enthält die Konfigurationsparameter für das StEin Modul.
Die angegeben Werte dienen als Parameter für die Config Modul Value Datagramme.

EINSTELLUNGEN FÜR SEKTIONEN

Object	Item	Beschreibung	Min.	Max.
0x1000		Abschnitt/Section		
	1	Abschnitt Art / Section Type: Bit 0: Besetzmeldung Bit 1: HLU Bit 2: ZACK Bit 3: RailCom Bit 8: Kehrschleife	1	
	2	Stein Modulnummer (=NID)		
	3	Globale Abschnittnummer (1 … 8)		
	4	Betriebsform / work type	0	
	5	HLU-Einstellung / speed setting		
	6	HLU Funktions Bits (F0 … F15)	0	255
	7	HLU Funktions Bits (F16 … F28)		
	10	Besetztschwelle normal	1	20
	11	Besetztschwelle feucht	20	50
	12	Besetztschwelle regen	50	100
	13	Abschalt-Stromlimit nieder in mA	500	2000
	14	Zeitdauer für Stromlimit ms def 500	0	5000
	15	Zeitdauer für Wiedereinschaltung ms def 500	0	5000
	16	Einschaltungslimit nieder ms	0	20
	17	Abschalt-Stromlimit hoch in mA	1500	4000
	18	Zeitdauer für Stromlimit ms def 500	0	5000
	19	Zeitdauer für Wiedereinschaltung ms def 500	0	5000
	20	Einschaltungslimit hoch	0	20
	22	MX9 Adresse	901	963
	24	Aktuell gültige Besetztschwelle	1	3
0x1001 … 0x1008		Gültige Abschnittsnummer		
	8	Position Code		
	9	Gleiseinfahrt		
	21	Connection point		
	22	MX9 Adresse		
	23	MX9 Abschnittnummer		

GPIO CONTROL

Object	Item	Beschreibung	Min.	Max.
0x0200	1	GPIO 1 Mode		
....		.....		
0x020F	1	GPIO 16 Mode		

GLOSAR

Begriff	Erklärung
UID	Weltweit eindeutig 32 Bit Nummer (Unique Identifier). Wird typischerweise während dem Anmeldeprozess verwendet.
NID	Network ID, 16 Bit Nummer welche im laufenden Betrieb zur Adressierung der Module, Fahrzeuge, Dekoder, …. Verwendet wird.
TSE	Track Signal Engine. Jener Programmteil, welcher die logischen Befehle in die jeweiligen Schienen Befehle (DCC, MM2, mfx, …) umsetzt. Ebenso ist dieser Programmteil für die Synchronisierung des Schienen Empfangs (RailCom, ZACK, mfx) zuständig.
OBJECT	Der Begriff Objekt bezeichnet eine allgemeine Datenstruktur, welche Daten unterschiedlicher Module, Fahrzeuge, Dekoder, Enkoder, etc. enthält. Diese Struktur kann dabei in abstrakter Form oder als konkreter Eintrag in einer Datenbank verwendet werden.
OBJDB	Die Objekt Datenbank beruht auf OBJECT's (siehe oben). Die konkreten Werte der Objekte werden in der OBJDB koordiniert verwaltet und je nach Bedarf permanent gespeichert.

HISTORY

In der folgenden Liste sind die bisher implementierten Befehle angeführt.

Gruppe	Befehl	Bezeichnung	MX10	MX32
0x0A	0x00	ItsMe	27.03.2012	27.03.2012
0x0A	0x02	Login 1	27.03.2012	27.03.2012
0x0A	0x03	Login 2	27.03.2012	27.03.2012
0x0A	0x04	Login 3		
0x02	0x02	Fahrzeug Speed		
0x02	0x03	Fahrzeug Funktionen		
		Anpassung ‚Legacy' NId's für MX1, MX8, MX9	01.01.2014	01.01.2014
0x0A	0x10	Funk Prozessor Kommunikation	12.02.2014	12.02.2014
0x0A	0x11	Funk Prozessor Kommunikation	12.02.2014	12.02.2014
0x01	0x02	Accessory GPIO für DCC ‚Basic' Dekoder	03.03.2014	03.03.2014
0x01	0x02	Accessory GPIO für MX9 Signale	03.03.2014	03.03.2014

REFERENZ CODE IN C# FÜR PC ANBINDUNG

Im folgenden befinden sich einige Beispiele und Hilfsfunktionen für die PC Kommunikation:

UMWANDLUNG VON 16BIT ZAHLEN:

Da das interne Protokoll im Little Endian Format arbeitet, PC's jedoch das Big Endian Format verwenden ist eine Umwandlung zwischen diesen Formaten erforderlich.

Um eine Zahl aus einem Byte Stream (typischerweise Empfangsdaten vom System) in eine 16 Bit Zahl für den PC umzuwandeln ist folgende Funktion sinnvoll:
Die Funktion geht davon aus, das die Empfangsdaten in einem Byte Buffer mit Namen iData[..] vorliegen. Durch _iByte wird angegeben ab welchem Byte die Zahl in diesem Bytearray liegt. Die jeweils 'umgewandelte' Zahl wird dem Aufrufer zurückgegeben.

```
public UInt16 DataI16Get(int _iByte)
{
       UInt16 iTemp;
       iTemp  = (UInt16)((iData[_iByte + 0] >> 0) & 0x00FF);
       iTemp |= (UInt16)((iData[_iByte + 1] << 8) & 0xFF00);
       return (iTemp);
}
```

Natürlich ist auch eine Umkehrung zum senden von 16Bit Zahlen erforderlich, dies kann mit folgender Funktion geschehen:

Als Parameter sind _iByte und _iData zu übergeben, dabei bestimmt _iByte ab welcher Position die Zahl in den Byte Stream (Buffer) einzutragen sind. _iData ist dabei die Zahl, welche entsprechend umzuwandeln ist.

```
public void DataI16Set(int _iByte, UInt16 _iData)
{
       iData[_iByte + 0] = (byte)((_iData >> 0) & 0xFF);
       iData[_iByte + 1] = (byte)((_iData >> 8) & 0xFF);
}
```

VERBINDUNGSAUFBAU PC SOFTWARE MX10:

Folgender Ablauf sollte von einer PC Software zum Aufbau der MX10 Verbindung eingehalten werden:

PC		**MX10**	
Datagramm	**Mode**	**Datagramm**	**Tx/Rx**
‚Z22Z'	Tx	Ping	
Warten auf ‚Ping'			
Abfrage ‚Modul Info Software'	Req	Software Version Zentrale	Ack
Der weitere Ablauf kann sich je nach Software Version unterscheiden Der beschriebene Ablauf gilt ab Version 0.14.0311			
Kommando ‚Interface Option Software Typ'	Cmd	‚Interface Option Software Typ'	Ack
Kommando ‚Interface Option Features'	Cmd	‚Interface Option Features'	Ack
Kommando ‚Modul Info Datum'	Cmd	‚Modul Info Datum'	Ack
Folgender Befehl ist optional sollte jedoch ausgeführt werden. Der Name (sofern angegeben) wird in diversen Anzeigen verwendet und hilft damit dem Anwender!!			
Senden des Application Names	Cmd	Bestätigung für Name	Ack
Abfrage der Geräte Gruppen Gruppe 0x0000 ➔ Fahrzeuge Gruppe 0x3000 ➔ Zubehör (DCC, MM1, …) Siehe Tabelle ‚Legacy Devices'	Req	Anzahl/Status der Geräte Gruppe	Ack

EMPFANGS FUNKTION:

SENDE FUNKTION: