WICHTIGE HINWEISE

Diese Beschreibung enthält bereits geprüfte, noch ungeprüfte und geplante Datagramme.
Zwecks Unterscheidung werden Farben verwendet:
Alle normal/schwarz geschriebenen Datagramme sind durch das MX32 und eine interne PC Testsoftware geprüft und funktionieren gemäß dieser Dokumentation.
Alle rot geschriebenen Angaben sind geplante/vorbereitete Datagramme, welche derzeit noch nicht oder fehlerhaft implementiert sind.

Ausnahme davon sind nur jene Datagramme und System Verhalten, welche in der ÖFFENTLICHEN Doku bekanntgeben sind.

Alle grün kursiv beschrieben Funktionen sind in Planung!!

LAN VERBINDUNG

Ab MX10 Version 1.18.0090 stellt das MX10 auch eine Ethernet/LAN Schnittstelle zur Kommunikation zur Verfügung. Die Version 4.00 des Schnittstellen Protokolls enthält einen ersten Entwurf, wie diese zu nutzen ist.

INHALTSVERZEICHNIS

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 erweitertes Protokoll verwenden.
Das MX10 unterstützt beide Protokolle an seinen beiden CAN Buchsen.
Wobei die mit ZIMO beschrifteten CAN Buchsen Default mäßig das ‚alte' Protokoll nutzen, die Fremdgeräte Buchse das neue CAN Protokoll. Dies kann aber im MX10 Menu jederzeit nach Bedarf geändert werden.

ZCAN20 LAYER-7 DEFINITION

Damit aber eine Modellbahn tatsächlich gesteuert werden kann, muss auch ein Layer-7 (Application/Anwendung) definiert werden.
Die Layer-7 Definition enthält die Definition der 29Bit ID's und der 8 Datenbytes. Damit ein solches CAN Protokoll einwandfrei funktioniert, muss sichergestellt werden, dass die 29Bit ID's des CAN Busses systemweit eindeutig ist.
Zusätzlich ist zu bedenken, dass 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, dass das jeweilige Gerät nur jene Nachrichten verarbeiten muss, welche tatsächlich relevant sind. Ein Weichendecoder bekommt so beispielsweise 'Lok Befehle' gar nicht mit. Umgekehrt braucht sich ein Booster nicht um 'Schaltbefehle' kümmern.

WESENTLICHE BEGRIFFE:

Im Folgenden werden einige in dieser Anleitung genutzten Begriffe beschrieben.

DER BEGRIFF OBJECT:

Im Folgenden wird häufig der Begriff 'Object' verwendet. Unter diesem Begriff kann man sich jegliches 'adressierbare' Ding auf einer Modellbahnanlage vorstellen. Konkret ist es also vorerst unerheblich, ob es sich um ein Fahrzeug, eine Weiche oder einem USB Stick handelt. Wichtig ist nur, dass solch ein Objekt im aktiven System eindeutig Identifizierbar ist und einige wenige Basiseigenschaften (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 Decoder, ...) 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 Decoder 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' Decoder und andere Geräte nach außen hin wie ZCAN20 Objects.

DER BEGRIFF UID:

Im Sinne des ZIMO Systems ist eine UId eine 32 Bit Zahl, welche über alle ZIMO Systemgeräte hinweg eindeutig ist.
Jedes ZCan 2.x Gerät sendet diese UId im Ping Event, womit der Zusammenhang zwischen UId und NId hergestellt wird.

DER BEGRIFF NID:

Da im CAN System jedes Datagramm eindeutig sein muss, erfüllt die 16 Bit NId hauptsächlich diese Aufgabe.
Darüber hinaus ist sie auch eine deutliche Vereinfachung zur eindeutigen Identifizierung von Fahrzeugen und Modul Nummern. Typischerweise besteht so eine NId immer aus der jeweiligen Adresse/Modul Nummer und einem Offset Wert. Anhand des Offset Wertes kann auch leicht die jeweilige Klasse abgeleitet werden. Eine exakte Auflistung befindet sich im Abschnitt ‚Translate Tabelle für Legacy Devices'.

DER BEGRIFF PIN:

Unter Pin ist EIN Anschluss eines Moduls, Decoders zu verstehen. Dies kann EIN Schienenausgänge der Zentrale/Boosters sein, EIN Gleisabschnitt, EIN Schaltausgang (z.B.: für Weichen) oder EIN Eingang (z.B.: Balisen) sein. Da das Protokoll ja einen abstrakten Transportlayer (Siehe ISO Layer Definition) obliegt die physische Realisierung den jeweiligen Endgeräten und ist auch in deren Detailbeschreibung zu finden.

DER BEGRIFF PORT:

Ports haben nur den Zweck mehrere Pins (=Anschlüsse), sofern möglich, mit einer Abfrage einzulesen. Ports als solches können keine Stellbefehle übermitteln.
Man kann sich ein Port vorstellen wie früher die Parallel Ports an einem PC. Dort waren
25 Pins in einem Stecker (eben dem Parallel Port) zusammengefasst.

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' Zubehördecoder
0x0000	0x3200	0x39FF	2058	DCC ‚eXtended' Zubehördecoder
0x0000	0x3A00	0x3DFF	1024	MM1 Zubehördecoder
0x0000	0x4000	0x43FF	1024	S88 Rückmelder
0x0000	0x4400	0x45FF	1024	X-Net Decoder
0x0000	0x4600	0x47FF	1024	X-Net FeedBack
0x0000	0x4800	0x4FFF	2048	Frei [2]
ZIMO Gerätegeneration 1				
0x0000	0x5000	0x503F	64	MX1
0x0000	0x5040	0x507F	64	MX8 Module, Channel 1
0x0000	0x5080	0x50BF	64	MX9 Module, Channel 1
0x0000	0x50C0	0x50CF	16	CSA Module
0x0000	0x50D0	0x50DF	16	MX31
0x0000	0x5100	0x513F	64	MX8 Module, Channel 2
0x0000	0x5140	0x517F	64	MX9 Module, Channel 2
0x0000	0x5800	0x5800	128	I2C eXtender, Unterscheidung siehe SubCmd
	0x5A00	0x5AFF	256	Blockstellen
ZIMO System Database				
	0x6000	0x60FF	256	Panels / GBS / Stellwerke
	0x6100	0x63FF	768	Routes (Fahrstraßen)
	0x6400	0x65FF		
	0x6600	0x66FF	256	Sound Projects
	0x6700	0x7FFF		RESERVIERT
mfx Adressen				
0x0000	0x8000	0xBFFF	16384	Mfx Loks
ZIMO CAN 2.xx Geräte (Auch von nicht ZIMO-Herstellern)				
	0xC000	0xC0FF	256	MX10 Zentralen
	0xC100	0xC1FF	256	MX10 Booster
	0xC200	0xC2FF	256	Spezialgeräte (IF, ...)
	0xC300	0xC3FF	256	Fahrpulte
	0xC400	0xC4FF	256	MX32 Funkmodule
	0xD000	0xDFFF	4096	Module
	0xE000	0xEFFF	4096	Objekte
	0xF000	0xFFFF	4096	Files

TABELLE STEIN MODUL NUMMERN:

StEin Nr.	NId	StEin Nr.	NId	StEin Nr.	NId	StEin Nr.	NId
0	0xD000	1	0xD001	2	0xD002	3	0xD003
4	0xD004	5	0xD005	6	0xD006	7	0xD007
8	0xD008	9	0xD009	10	0xD00A	11	0xD00B
12	0xD00C	13	0xD00D	14	0xD00E	15	0xD00F
16	0xD010	17	0xD011	18	0xD012	19	0xD013
20	0xD014	21	0xD015	22	0xD016	23	0xD017
24	0xD018	25	0xD019	26	0xD01A	27	0xD01B
28	0xD01C	29	0xD01D	30	0xD01E	31	0xD01F
32	0xD020	33	0xD021	34	0xD022	35	0xD023
36	0xD024	37	0xD025	38	0xD026	39	0xD027
40	0xD028	41	0xD029	42	0xD02A	43	0xD02B
44	0xD02C	45	0xD02D	46	0xD02E	47	0xD02F
48	0xD030	49	0xD031	50	0xD032	51	0xD033
52	0xD034	53	0xD035	54	0xD036	55	0xD037
56	0xD038	57	0xD039	58	0xD03A	58	0xD03B
60	0xD03C	61	0xD03D	62	0xD03E	63	0xD03F
64	0xD040	65	0xD041	66	0xD042	67	0xD043
68	0xD044	69	0xD045	70	0xD046	71	0xD047
72	0xD048	73	0xD049	74	0xD04A	75	0xD04B
76	0xD04C	77	0xD04D	78	0xD04E	79	0xD04F
80	0xD050	81	0xD051	82	0xD052	83	0xD053
84	0xD054	85	0xD055	86	0xD056	87	0xD057
88	0xD058	89	0xD059	90	0xD05A	91	0xD05B
92	0xD05C	93	0xD05D	94	0xD05E	95	0xD05F
96	0xD060	97	0xD061	98	0xD062	99	0xD063

TABELLE MX8 MODUL NUMMERN:

MX8 Nr.	NId	MX8 Nr.	NId	MX8 Nr.	NId	MX8 Nr.	NId
0	0x5040	1	0x5041	2	0x5042	3	0x5043
4	0x5044	5	0x5045	6	0x5046	7	0x5047
8	0x5048	9	0x5049	10	0x504A	11	0x504B
12	0x504C	13	0x504D	14	0x504E	15	0x504F
16	0x5050	17	0x5050	18	0x5052	19	0x5053
20	0x5054	21	0x5055	22	0x5056	23	0x5057
24	0x5058	25	0x5059	26	0x505A	27	0x505B
28	0x505C	29	0x505D	30	0x505E	31	0x505F
32	0x5060	33	0x5061	34	0x5062	35	0x5063
36	0x5064	37	0x5065	38	0x5066	39	0x5067
40	0x5068	41	0x5069	42	0x506A	43	0x506B
44	0x506C	45	0x506D	46	0x506E	47	0x506F
48	0x5070	49	0x5071	50	0x5072	51	0x5073
52	0x5074	53	0x5075	54	0x5076	55	0x5077
56	0x5078	57	0x5079	58	0x507A	58	0x507B
60	0x507C	61	0x507D	62	0x507E	63	0x507F

TABELLE MX9 MODUL NUMMERN:

MX9 Nr.	NId	MX9 Nr.	NId	MX9 Nr.	NId	MX9 Nr.	NId
0	0x5080	1	0x5081	2	0x5082	3	0x5083
4	0x5084	5	0x5085	6	0x5086	7	0x5087
8	0x5088	9	0x5089	10	0x508A	11	0x508B
12	0x508C	13	0x508D	14	0x508E	15	0x508F
16	0x5090	17	0x5090	18	0x5092	19	0x5093
20	0x5094	21	0x5095	22	0x5096	23	0x5097
24	0x5098	25	0x5099	26	0x509A	27	0x509B
28	0x509C	29	0x509D	30	0x509E	31	0x509F
32	0x50A0	33	0x50A1	34	0x50A2	35	0x50A3
36	0x50A4	37	0x50A5	38	0x50A6	39	0x50A7
40	0x50A8	41	0x50A9	42	0x50AA	43	0x50AB
44	0x50AC	45	0x50AD	46	0x50AE	47	0x50AF
48	0x50B0	49	0x50B1	50	0x50B2	51	0x50B3
52	0x50B4	53	0x50B5	54	0x50B6	55	0x50B7
56	0x50B8	57	0x50B9	58	0x50BA	59	0x50BB
60	0x50BC	61	0x50BD	62	0x50BE	63	0x50BF

GENERELLER AUFBAU DER 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, dass die jeweiligen CAN Geräte diese als Filterkriterium 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 genaue 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 Kommandogruppe. Gibt die jeweilige Command Group an (Sys, FeedBack, Loco, ...)
Cmd	Dieses 6 Bit Feld enthält das jeweilige Command
Mode	0b00: Req (Abfragen) 0b01: Cmd (Steuerbefehle, Wert setzen, ...) 0b10: Evt (Events = Ungefragte Informationen) 0b11: ACK (Bestätigung)
NetworkID	Identifikationsnummer des 'Absenders'. Primär notwendig um Kollisionen am Bus zu vermeiden.
ID Feld Aufteilung bei File Transfer	
Flag	Immer '1', dient zur Unterscheidung anderer Protokolle
Group	0x0F (File Transfer)
Counter	8 Bit Telegrammzähler
File ID	File ID

COMMAND GROUP'S (BEFEHLS GRUPPEN):

Alle Befehle sind in folgenden Gruppen zusammengefasst:

Gruppe	Code	Verwendung/Inhalt
System	0x00	systemkritische Aufgaben (Ein/Aus, Notstopp, ...)
Zubehör	0x01	Befehle zum Steuern des Zubehörs. Damit sind sowohl Encoder/Rückmelder wie auch Decoder gemeint.
Fahrzeuge	0x02	Befehle zum Steuern der Fahrzeuge (Mobile Decoder)
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	Statusmeldungen, meist ungefragte Meldungen
Frei	0x09	Darf von Fremdsystemen je nach Bedarf verwendet werden.
Network	0x0A	Network Management, Modulanmeldung, ...
File Control	0x0E	Steuer Befehle für File Transfer wie z.B.: File Open, Close, ..
File Transfer	0x0F	File 'Inhalt' (Daten) Telegramme
PC Datagramme	0x1n	Um den Datendurchsatz zwischen einer PC Anwendung und dem MX10 zu optimieren, gibt es einige spezifische Datagramme.

PC INTERFACE:

ETHERNET/UDP INTERFACE

Das Ethernet Interface nutzt grundsätzlich die gleiche Methode zur Daten Übertragung.
Der App-Layer Datentransfer erfolgt im Ethernet (LAN/W-LAN) über IP/UDP Frames.
Eine PC Software (bzw. App) sendet Ihre Anfragen/Befehle über der UDP Port 14520 an das MX10, die Antworten des MX10 kommen am PC, Tab, … am Port 14521 an.

Hinweis: Die Ports können am MX10 auch auf andere Werte gestellt werden,
bitte Anleitung MX10 beachten.

Um die Verbindung zu initiieren muss die Anwendung ein Port ‚Open' ([0x0A.0x06 bzw. 0x1A.0x06]) an das MX10 senden.

AUFBAU DER DATENTELEGRAMME FÜR DAS ZIMO 2.X FORMAT PER UDP:

Für die Datagramm Übertragung im Ethernet sind keine Delimiter erforderlich (‚Z2' … ‚2Z') da dies ja durch dir Ethernet Framelogik abgedeckt ist. Wie auch bei der USB (VCom-) Schnittstelle werden im Ethernet die Daten 1:1 wie am CAN Bus übertragen. Allerdings gibt es einige zusätzliche Ethernet Datagramme, welche deutlich mehr Daten an das System übertragen können bzw. kann das MX10 auch deutlich mehr Daten in einem Datagramm an den PC senden. Diese LAN Spezialbefehle sind gesondert angeführt.

Size (DLC)	[Counter]	Group	Cmd+Mode	NID	Data 0 ... x
16 Bit	16 Bit	8Bit	8Bit	16 Bit	
	Nur für Debug				

Da ein Ethernet Frame ja bis zu 1536 Byte umfasst, ist die Längenangabe gegenüber dem CAN Bus auf 16 Bit angewachsen. Zusätzlich gibt es, ein derzeit ungenutztes, 16 Bit Feld. Dieses ist für spätere Erweiterungen vorgesehen.

HINWEISE ZUR PROTOKOLLIMPLEMENTIERUNG

1. Grundsätzlich werden alle Befehle durch ‚ACKs' bestätigt, diese Bestätigung kann jedoch, je nach Befehl, einige Zeit dauern.
2. Eine PC Software MUSS auch ‚ACKs' 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-Software, die abweichenden Daten entsprechend zu verarbeiten und darzustellen.
4. Eine PC Software **muss** davon ausgehen, dass Sie nicht die einzige Befehlsquelle im System ist, dass also z.B.: Fahrpulte, eventuell andere Programme oder systeminterne Abläufe Befehle generieren.
5. Die ‚ACKs' repräsentieren **immer** den MX10 internen Zustand. Dieser kann aus diversen Gründen vom gewünschten **Sollzustand** 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 (CMDs) werden vom MX10 sofort im jeweils betroffenen Objekt (Fahrzeug, Weiche, MX8, MX9, …) abgebildet.
8. Jedes Objekt verfügt über eigene, asynchron laufende State Engine, welche den gewünschten **Sollzustand** umsetzt.
9. Die Objekt State Engines verarbeiten die Befehle mit der jeweils möglichen Geschwindigkeit. DCC Befehle benötigen ca. 20mS bis sie einen Fahrzeugdecoder zum ersten Mal erreichen. Befehle an MX8 oder MX9 Module im Mittel nur ca. 5mS.
10. Daraus ergibt sich, dass MX8/MX9 Befehle z.B.: Schienenbefehle (Fahrzeuge, Weichen) ‚überholen' können. Der PC **muss** also davon ausgehen, dass er ein ACK für ‚später' gesendete Befehle bekommen kann.
11. Insbesondere bei Fahrzeugbefehlen ist auf Folgendes zu achten:
    Wenn an ein und dasselbe Fahrzeug Geschwindigkeitsänderungen und Funktionsbefehle gesendet werden, so haben die Geschwindigkeitsbefehle **Vorrang** vor den Funktionsbefehlen und werden daher zuerst verarbeitet und auch bestätigt.
12. Sollten per Interface fortlaufend Geschwindigkeitsbefehle gesendet werden, in schnellerer Folge, 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. 1.5 Sekunden!
14. Wie oben angeführt, wird normalerweise jeder Befehl durch ‚sein' ACK bestätigt, also im wesentlichen das jeweilige Kommando 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 als ‚Sammel' ACK, in welchem der Zustand aller 32 Funktionen enthalten ist.
15. Eine Anwendung muss zumindest alle 60Sec. eine Nachricht an das MX10 senden, ansonsten wird die Verbindung vom MX10 ‚beendet'. Dies ist eine Schutzvorkehrung für Anwendungen, welche z.B.: ‚abstürzen', oder den W-LAN Bereich verlassen.

OPTIMALE IMPLEMENTIERUNG FÜR EINE PC SOFTWARE:

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

1. Alle von Ihr gesendeten Befehle sollte die PC-Software 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 ‚ACKs' 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 **Sollwert** abweichen.
5. Bei Abweichung MUSS die Software sinnvoll reagieren, z.B. durch Senden von:
   - Eigenen **Sollwert**,
   - Benutzer Info,
   - Automatische Anpassung von Stellwerken,
   - Geschwindigkeiten anderer Fahrzeuge … etc.
6. ACKs, 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.

VERBINDUNGSAUFBAU

Hinweis:
Doku noch nachtragen!

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 STATE [0X00.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x00	0b00		3	ObjectNId		Pin					
0x00	0x00	0b01		4	ObjectNId		Pin	Mode				
0x00	0x00	0b1x		4	ObjectNId		Pin	Mode				

[0x00.0x00], Type Definition für System Power

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 Gerät (Spezifiziert durch ObjectNId, Z.B.: MX10, StEin, …)) in den jeweils gewünschten Mode, dieser wird aber erst NACHDEM die internen Regelschleifen den Wechsel ausgeführt und durch Messungen verifiziert ist auch gemeldet.
Dieser Vorgang kann je nach gewünschtem Wechsel mehrere 100ms dauern.

Mit Cmd=0x00/M=0b01 kann der Pin 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 Objekt Ports werden binär kodiert, Kombinationen sind erlaubt:

Port	Ausgang
0b00000001	Schiene 1
0b00000010	Schiene 2
0b0…….. 00	Schiene 3 … 7 (Weitere MX10 im Booster Mode)
0b10000000	Booster Ausgang, bzw. Gleisausgang 8 beim MX9/StEin

Um ALLE Ausgänge mit einem Befehl zu schalten ist daher als Port 255 (=0xFF, =0b11111111) zu verwenden.

ACHTUNG:
Wenn mehrere Ausgänge gleichzeitig geschalten werden, so erfolgt die Bestätigung trotzdem jeweils einzeln für die ‚vorhandenen' Pins. Wenn also z.B.: kein weiteres MX10 im Booster Mode vorhanden ist, so gibt es KEIN ACK für diese nichtexistenten Schienen!

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

Power Modes:

Mode	Zustand (Allgemein, MX10)	MX10	MX9	StEin
0	Als Command ungültig. Wird bei Request im ACK verwendet, wenn das MX10 einen Zustandswechsel ausführt, dieser aber zum Zeitpunkt der Abfrage noch unklar/bzw. noch nicht stabil ist.	N.A.	N.A.	N.A.
1	Der ‚Pin' wird in Normalbetrieb geschalten	Ja	Ja	Ja
2	Der ‚Pin' wird in Sammelstopp mit Fahrstufe ‚0' geschalten (SSP0)	Ja	n.a.	n.a.
3	Der ‚Pin' wird in Emergency Sammelstopp geschalten (SSPe)	Ja	n.a.	n.a.
4	Der ‚Pin' wird ‚AUS' geschalten	Ja	Ja	Ja
5	Der ‚Pin' wird in Service Mode geschalten	Ja	n.a.	n.a.
6		n.a.	n.a.	n.a.
7		n.a.	n.a.	n.a.
8		n.a.	n.a.	n.a.
9		n.a.	n.a.	n.a.
10	Überstrom	Ja	Ja	Ja

SYSTEM RESET [0X00.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x02	0b01		4	SystemNID		ResetCode					
0x00	0x02	0b10		4	SystemNID		ResetCode					
0x00	0x02	0b11		4	SystemNID		ResetCode					

Mit diesen Datagrammen wird ein System Reset an alle Bus Teilnehmer gesendet.

Aktuell wird dies von einem MX10 Master unmittelbar nach seinem Reset ausgesendet, damit alle Booster und andere Module sich neu mit diesem Synchronisieren. Im einfachsten Falle in dem sie selber einen Reset ausführen.

Reset Code:
0x0100: Master hat Reset ausgeführt, Neusynchronisation durchführen
0x0900: StEin Modul hat Reset ausgeführt.

SYSTEM ERROR [0X00.0X10]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x10	0b10		6	SystemNID		Pin		Error Code			

Mit diesem Event meldet das System Fehlerzustände.

Die Bedeutung der Pin Nummern und Error Codes sind Geräte abhängig.
Die derzeit festgelegten Werte befinden sich im Anhang, Tabellen.

TSE ERROR (0X11)

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x00	0x11	0b10		4	SystemNID		Code	Pin				

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

ACCESSORY COMMAND GROUP [0X01]

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

ACCESSORY STATE [0X01.0X00]

Jedes Steuersystem (Fahrpult, PC-Software) sollte den Zubehörstatus immer als erste Initialabfrage ausführen. Insbesondere für die MX8 und MX9 Module.

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	

[0x01.0x00], Type Definition für Accessory State

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 wird die Anzahl der mS gesendet, welche seit dem letzten Steuerbefehl vergangen sind.

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

Hinweis:
Sollte ein Steuergerät 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.

HINWEIS, STEIN:

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

Error	Verwendung	Data 1/2
0x0000	Kein Fehler	
0x0002	Keine Gleisspannung	
0x0003	Keine Zubehör Versorgung	
0x0004	Kein DCC Signal	
0x0005	Keine CAN Spannung	
0x0006	Keine +20V	
0x0007	Keine +5V	

ACCESSORY MODE [0X01.0X01]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
Kurzform für Schienen Decoder (DCC/MMx) bzw. einfache Module												
0x01	0x01	0b00		2	ZubehörNID							
0x01	0x01	0b1x		4	ZubehörNID		Mode					
Langform, für komplexe Module												
0x01	0x01	0b00		3	ZubehörNID		Type					
0x01	0x01	0b1x		6/8	ZubehörNID		Type	Pin/Port	Mode1		Mode2	
Langform, für komplexe Module												
0x01	0x01	0b00		4	ZubehörNID		Type	Pin/Port				
0x01	0x01	0b1x		6/8	ZubehörNID		Type	Pin/Port	Mode1		Mode2	

Dieses Datagramm dient der Abfrage und der Einstellung der Zubehörbetriebsart.

HINWEIS, DCC BASIC DECODER:

Für DCC bzw. MMx Decoder gilt die Kurzform des Datagrammes.
DCC Basic Decoder haben eine Zubehör NID im Bereich 0x3000 … 0x31FF (Adresse 1 … 512).
Die Standard DCC Zubehör Decoder kennen 2 Betriebsarten:
Mode ‚0‘: Default Mode (bzw. Betriebsart unbekannt)
Mode ‚1‘: Paarbetrieb (Typischerweise Weichendecoder)
Mode ‚2‘: Einzelbetrieb (Jeder Ausgang kann getrennt geschalten werden).

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

HINWEIS, MX8 MODULE:

Für die MX8 Module gilt die Kurzform des Datagrammes.
MX8 Module haben eine Zubehör NID im Bereich 0x5040 … 0x507F (Adresse 0 … 63).
Die MX8 Module kennen folgende Betriebsarten:
Mode ‚0‘: Default Mode (bzw. Betriebsart unbekannt)
Mode ‚1‘: Beide Ausgangsgruppen im Paar/Paar Betrieb
Mode ‚2‘: Ausgangsgruppe 1: Paarbetrieb, Ausgangsgruppe 2: Einzelbetrieb
Derzeit nicht unterstützt
Mode ‚3‘: Ausgangsgruppe 1: Einzelbetrieb, Ausgangsgruppe 2: Einzelbetrieb
Derzeit nicht unterstützt

HINWEIS, MX9 MODULE:

Für die MX9 Module gilt die Kurzform des Datagrammes.
MX9 Module haben eine Zubehör NID im Bereich 0x5080 … 0x50BF (Adresse 0 … 63).
Die MX8 Module kennen keine besonderen Betriebsarten:
Mode ‚0‘: Default Mode (bzw. Betriebsart unbekannt)
Mode ‚1‘: MX9 Modul vorhanden

HINWEIS, MX10 ZENTRALE:

Für das MX10 gilt die jeweilige MX10 NId.

Type	Verwendung	Mode1	Mode2
0x00			
0x10	ABA Ausgänge	0x0000 = unbekannt 0xnnn1 = Ausgang kann ‚offen' sein 0xnnn2 = Ausgang kann ‚GND' schalten 0xnnn4 = Ausgang kann ‚+5V' schalten 0x100n = Einzelbetrieb 0x110n = Service Mode Relais 0x8001 = Signale 0x8000 = CZ Signale 0x9nn0 = S88 0xAnn0 = I2C	
0x80 ... 0xFF	I2C eXtender, 3Byte Abfrage	Chip Type	Next I2C Addr
0x80 ... 0xFF	I2C eXtender, 4 Byte Abfrage		

HINWEIS, STEIN MODULE:

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

Type	Verwendung	Mode1	Mode2
0x00			
0x00	Gleisabschnitt 1 … 16, Besetztmeldungen	Schwelle mA ab wann Besetzt gilt	Timing mS wie lange Besetzt gilt
0x10	Output Ports	0x0000 = unbekannt 0xnnn1 = Ausgang kann ‚offen' sein 0xnnn2 = Ausgang kann ‚GND' schalten 0xnnn4 = Ausgang kann ‚+5V' schalten 0xnnn8 = Ausgang kann ‚+20V' schalten 0x100n = Einzelbetrieb 0x200n = Paarbetrieb 0x300n = Mehrfachausgang 0x310n = Matrix Output 0x8200 = CZ Signale, Seriell	 Bitmuster für 2. Ausgang Bitmuster für Ausgänge HLU Stufe, wenn Balise Anzahl Stellungen
0x20	Input Ports	0x0000 = unbekannt 0x02nn = Schwelle für ‚Low' 0x04nn = Schwelle für ‚High' 0x300n = Eingang 0x310n = Matrix Input 0x5Tnn = Balise für Abschnitt ‚nn', ‚T' gibt den ‚Umschaltzustand' an.	 Bitmuster für Ausgang Bitmuster für Eingang HLU Stufe, wenn Balise (0xFF00 = Keine Vorgabe)

Beispiel für Balisen Mode (0x5Tnn):
T='0': Umschaltung erfolgt wenn Balisen Eingang von ‚High' nach ‚Low' geht.
T='1': Umschaltung erfolgt wenn Balisen Eingang von ‚Low' nach ‚High' geht.
‚nn' gibt den Abschnitt an, auf welchen die Balise wirkt.

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 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 PORT [0X01.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x02	0b00		4	ZubehörNID		Type					
0x01	0x02	0b1x		8	ZubehörNID		Type		PORT States			

[0x01.0x02], Type Definition für Accessory Port

Diese Datagramme dienen der effizienten Statusabfrage von simplen Ein-/Ausgängen. Es werden je Gruppe bis zu 32 Ein-/Ausgangszustände übertragen.
Durch M=0b00 kann vom Gerät ‚ZubehörNID' der Port Bereich ‚Type' abgefragt werden.
Diese werden mit M=0b11 (ACK) beantwortet. Sollte irrtümlicherweise ein Command (0b01) an die Zentrale gesendet werden, so wird die mit einem ‚Command Error' beantwortet.

ACHTUNG:
Diese Datagramme sind nur als ‚Request' erlaubt und haben NUR den Zweck einer schnellen Informationsbeschaffung.
ECHTE Schaltvorgänge MÜSSEN über die Port Datagramme abgewickelt werden.

HINWEIS, DCC BASIC DECODER:

Die DCC Basic Decoder sind in den NID Bereich 0x3000 bis 0x31FF gemappt.
Diese Datagramme werden NUR für DCC Decoder im Einzel Betrieb (Mode=2) unterstützt. Diese Betriebsart MUSS VOR Verwendung entsprechend gesetzt werden.

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

ACHTUNG:
Ein Request wird mit einem Acknowledge aus dem internen MX10 Speicher beantwortet!!
Gleichzeitig wird an das jeweilige MX8 eine Anfrage gesendet.
Es kann daher bei fehlerhaften Weichen bzw. MX8 Modulen zu falschen Antworten kommen.
Sobald ein MX8 die Statusanfrage beantwortet wird die als PIN EVENT an den PC weitergemeldet.
Eine PC Software muss daher die Differenzen zwischen ACK und EVT berücksichtigen und entsprechende Maßnahmen setzen. Z.B.: Befehl wiederholen, Anwender Informieren, …
Je nach verwendet MX8 und/oder Weichenantrieb sind Fehlmeldungen (Fehlerhafte ACK's) mehr oder minder wahrscheinlich. Bei Motor, Servo Antrieben stimmen die Stellungen typischerweise immer, bei ‚Klick-Klack' Antrieben ist dies extrem vom jeweiligen Antrieb abhängig.

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. Da das MX9 die Zustände der ALA nicht speichert und auch beim MX9 nicht abgefragt werden können, erhält man nur den gültigen Status von zuvor gesetzten MX9ALA Zuständen, weil nur das MX10 diese speichert.

HINWEIS, STEIN:

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

Type	Verwendung	Value
0x0000		
0x0002	Lokale StEin Tasten NUR EVENT!!	Bit ‚0' = Taste 1, Bit ‚1' = Taste 2, Bit ‚2' = Taste 3,
0x0100	Gleisabschnitt 1 … 8, Besetztmeldungen	0b0xxx: Frei, 0b1xxx: Besetzt 0bx0xx: Keine Zugnummern, 0bx1xx: Zugnummern 0bxx0x/0bxx1x: Reserviert 0bxxx0: Normal Last, 0bxxx1: Überlast
0x0200	Ausgänge	0b00 = N.A., 0b01=Offen, 0b10= Low, 0b11=High
0x0300	Eingänge	0b00 = N.A., 0b01=Offen, 0b10= Low, 0b11=High

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GA8				GA7				GA6				GA5				GA4				GA3				GA2				GA1			
Out 16		Out 15		Out 14		Out 13		Out 12		Out 11		Out 10		Out 09		Out 08		Out 07		Out 06		Out 05		Out 04		Out 03		Out 02		Out 01	
Inp 16		Inp 15		Inp 14		Inp 13		Inp 12		Inp 11		Inp 10		Inp 09		Inp 08		Inp 07		Inp 06		Inp 05		Inp 04		Inp 03		Inp 02		Inp 01	

ACCESSORY PIN4 [0X01.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x04	0b00		3	ZubehörNID		Pin					
0x01	0x04	0b0x		4	ZubehörNID		Pin	Value				
0x01	0x04	0b1x		4	ZubehörNID		Pin	Value				

[0x01.0x04], Type Definition für Accessory Pin4

Wenn M = 0b00, DLC = 3, dann wird der Zustand des Ein/Ausganges (Pin) vom Zubehör NID abgefragt.
Durch M = 0b01, DLC = 4 wird der Ausgang des Zubehörs (NID) auf den angegeben Wert eingestellt.
Die Abfrage wird durch 0b1x beantwortet, ebenso Änderungen welche durch andere Einflüsse verursacht werden (z.B.: Manuelles Verstellen, Zeitablauf, ... oder andere Events)
Jedes Zubehör (Egal ob 'Schienen' gebunden oder am Bus System) darf bis zu 128 Ein/Ausgänge haben. Jeder Ausgang darf bis zu 256 'Stellungen' haben. Wie viele dieser Möglichkeiten genutzt sind, ist vom jeweiligen Modul abhängig.
Bit 7 der Pin Nummer gibt an ob der Pin Wert gültig (Valid) oder nur ‚virtuell' (gespeichert) ist.

HINWEIS, DCC BASIC DECODER:

DCC Basic Decoder haben eine Zubehör NID im Bereich 0x3000 … 0x31FF (Adresse 1 … 512).
Die Pins werden von 0 … 7 gezählt, bei Weichendecodern sind nur die geraden Pin Nummern gültig (0 = Weiche 1; 2 = Weiche 2; 4 = Weiche 3; 6 = Weiche 4;).
Ein Value von ‚0' bedeutet, dass der jeweilige Ausgang (Pin) abgeschaltet sein soll, ein Value von ‚1', das dieser eingeschalten sein soll.

ACHTUNG:
Die tatsächliche Funktion bei DCC Decodern ist extrem vom jeweiligen Decoder und dessen Konfiguration abhängig. Im Grunde bewirkt dieser 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 (siehe NMRA Norm 9.2.1: http://www.nmra.org/sites/default/files/s-9.2.1_2012_07.pdf)
A = 9 Bit Adresse des Decoders
D = Pin Nummer
C = Pin Zustand

HINWEIS, DCC EXTENDED DECODER:

DCC Extended Decoder haben eine Zubehör NID im Bereich 0x3200 … 0x39FF (Adresse 1 … 2048) gemäß NMRA. Die Port Nummer ist bei diesen Decodern immer ‚0', mit Value wird der jeweilige Aspekt angegeben.
ACHTUNG:
Die tatsächliche Funktion bei DCC Decodern ist extrem vom jeweiligen Decoder und dessen Konfiguration abhängig. Im Grunde bewirkt dieser 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 Decoders
X = Pin 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. Dabei gilt 0x5040 ist der Offset, die jeweilige Modul Nummer ist hinzuzuaddieren (Siehe Tabelle MX8 Modul Nummern)

Die MX8 Module haben 32 Ausgänge, welche je nach MX8 Konfiguration getrennt oder paarweise angesteuert werden können.
Im Paar Betrieb gelten für die Ansteuerung jeweils die geraden Pin Nummern (0, 2, 4, 6, …).
Die Antwort über die Schnittstelle kommt mit der gleichen Pin Nummer (0, 2, 4, …) und jeweils einem Bit für die beiden Ausgänge. Dabei bedeutet 0b00 und 0b11 eine fehlerhafte Stellung, 0b01 bzw. 0b10 die jeweils gültige Stellung.

HINWEIS, MX9:

Die MX9 Module werden in den NID Bereich 0x5080 bis 0x50BF gemappt. Dabei gilt 0x5080 ist der Offset, die jeweilige Modul Nummer ist hinzuzuaddieren (Siehe Tabelle MX9 Modul Nummern)
Die Pin Nummer wird für die verschiedenen MX9 Funktionen wie folgt genutzt:

Pin	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. Bit 0 … 5 ➔ HLU Speed.

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

ACCESSORY DATA [0X01.0X05]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x05	0b00		4	ZubehörNID		Pin	Type				
0x01	0x05	0b01		8	ZubehörNID		Pin	Type	Data	Data	Data	Data
0x01	0x05	0b11		8	ZubehörNID		Pin	Type	Data	Data	Data	Data

[0x01.0x05], Type Definition für Accessory Data

Mit diesen Datagrammen können Objektdaten abgefragt und gesetzt werden.
In einigen Fällen sind die Objektdaten ‚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 Pin Nummer wird fortlaufend von ‚0' bis ‚15' gezählt. Der ‚alte' Hauptabschnitt 1 hat daher die Pin Nummer ‚0' und ‚1', usw.

Type	Verwendung	Data (DB5 … DB8)
0x00	MX9 HLU Geschwindigkeit	
0x10	MX9 Fahrzeug ‚Sammelmeldung' ACHTUNG: Nur für PC Interface	
0x11	Erste, zweite erkannte Fahrzeugadresse an Pin	Erste Fahrzeug Adresse in DB 5&6, Zweite Fahrzeug Adresse in DB 7&8.
0x12	Dritte, vierte erkannte Fahrzeugadresse an Pin	Dritte Fahrzeug Adresse in DB 5&6, Vierte Fahrzeug Adresse in DB 7&8.

Wenn das MX9 nur eine Fahrzeug Adresse kennt, dann ist DB7&8 = 0x000, bei 3 Adressen dann wieder die Bytes 7&8 im Datagramm mit Type 0x12.

HINWEIS, STEIN:

Die StEin Module werden in den NID Bereich 0xD000 bis 0xDFFF gemappt.
Die Pin Nummer wird fortlaufend von ‚0x01' bis ‚0x0F' gezählt.
Fahrzeugliste endet mit Fahrzeugadresse ‚0'.

Hinweis: StEin, Roco10808 Melder!
Diese benutzen das MSB der Adressen um die jeweilige Aufgleisrichtung der betroffen Adresse zu melden.

Type	Verwendung	Data (DB5 … DB8)
0x00		
0x01		
0x0E	Anzahl Fahrzeuge gesamtes Modul/Objekt	
0x0F	Anzahl Fahrzeuge für Pin	
0x11	Erste, zweite erkannte Fahrzeugadresse an Pin	Erste Fahrzeug Adresse in DB 5&6, Zweite Fahrzeug Adresse in DB 7&8.
0x12	Dritte, Vierte erkannte Fahrzeugadresse an Pin	
….		
0x1F	Erkannte Fahrzeugadressen 31, 32	

ACCESSORY PIN6 [0X01.0X06]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x06	0b00		4	ZubehörNID		Pin	Type				
0x01	0x06	0b1x		6	ZubehörNID		Pin	Type	Value			

[0x01.0x06], Type Definition für Accessory Pin6

Wenn M = 0b00, DLC = 4 dann wird der Zustand des Ein/Ausganges (Pin) vom Zubehör NID abgefragt.
Durch M = 0b01, DLC = 6 wird der Ausgang des Zubehörs (NID) auf den angegeben Wert eingestellt.
Die Abfrage wird durch 0b1x beantwortet, ebenso Änderungen welche durch andere Einflüsse verursacht werden (z.B.: Manuelles Verstellen, Zeitablauf, ...)

HINWEIS, MX10:

Die ABA Ein-/Ausgänge des MX10 sind über die NId des MX10 ansprechbar.
Die Pin Nummer gibt den Ein- bzw. Ausgang an.

Pin	Type	Verwendung
0x00 … 0x07	0x20	ABA Eingänge Value enthält den jeweiligen Analog Wert des Eingangs
0x00 … 0x06	0x21	ABA Ausgänge Value ‚0x00' ➔ Ausgang ‚offen', ‚0x10' ➔ Ausgang ‚Low', ‚0x11' ➔ Ausgang ‚High'
0x00 … 0xFF	0x8n	I2C eXtender

HINWEIS, STEIN:

<table>
<tr><th>Pin</th><th>Type</th><th>Verwendung</th></tr>
<tr><td>0x00</td><td>0</td><td>Spezial Fälle</td></tr>
<tr><td>0x01</td><td>0</td><td>StEin Display, Nur COMMAND, Antwort ACK
DB5 = Digit 1, DB6 = Digit 2.</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>0x00 … 0x07</td><td>0x01</td><td>Block, Status:
Value = 0x0000 ➔ Fahrspannung AUS, Anzeigezustand Frei
Value = 0x0100 ➔ Fahrspannung Ein, Anzeigezustand Frei
Value = 0x1000 ➔ Fahrspannung AUS, Anzeigezustand Besetzt
Value = 0x1100 ➔ Fahrspannung Ein, Anzeigezustand Besetzt

Value = 0x1201 ➔ UESL- temporär, Anzeigezustand Besetzt
Value = 0x1202 ➔ UESL- temporär, Anzeigezustand Frei
Value = 0x1203 ➔ UESL- abgeschaltet, Anzeigezustand, Besetzt

Hinweis:
Kurzschluss Zustand kommt über System Gruppe (0x00) wegen CAN Priorität</td></tr>
<tr><td>0x00 … 0x07</td><td>0x02</td><td>
<table>
<tr><td>15</td><td>14</td><td>12 … 13</td><td>8 … 11</td><td>7</td><td>6</td><td>4 … 5</td><td>0 … 3</td></tr>
<tr><td>A2</td><td>OW/FR</td><td>Dir 2</td><td>HLU 2</td><td>A1</td><td>OW/FR</td><td>Dir 1</td><td>HLU 1</td></tr>
</table>
HLU 1/2:
0x0 ➔ AUS, 0x2 ➔ Halt, 0x4 ➔ UH, 0x6 ➔ U,
0x8 ➔LU, 0xA ➔ L, 0xC ➔ FL, 0xE ➔ Fahrt
Dir 1/2:
0x00 ➔ KEIN Richtungsbit, 0x01 ➔ Ost, 0x02 ➔ West
FR/OW: ‘0’ ➔ Ost/West, ‘1’ ➔ Vorwärts/Rückwärts.
A (Aktiv):
Bei Command muss A1 immer mit ‚1‘ gesendet werden.
Wenn im Command A2 ‚0‘ ist, so werden die Anweisungen für den Kontaktmelder ignoriert.
Wenn A2=‘1‘ so gelten die Vorgaben für den Kontaktmelder, sobald dieser ausgelöst wird.
Im ACK auf das Command wird immer mit A1=1 geantwortet, sollte jedoch der Kontaktmelder zum Zeitpunkt des Command’s aktiv sein, dann wird unmittelbar nach dem Command ACK das Umschalten per Event gemeldet.
Bei Abfrage bzw. Event ist je nach aktuellem Status A1 oder A2 = ‚1‘.</td></tr>
<tr><td>0x00 … 0x07</td><td>0x03</td><td>HLU Funktion</td></tr>
<tr><td>0x00 … 0x07</td><td>0x08</td><td>Aktueller Anschluss Strom, Value = Strom in mA</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>0x00 … 0x0F</td><td>0x10</td><td>Output Pins:
Value = 0x0000 ➔ Unbekannt
Value = 0x0001 ➔ AUS/OFFEN
Value = 0x0002 ➔ Masse
Value = 0x0004 ➔ VCC
Value = 0x10nn ➔ Mehrfach Stellungen 1 …. Bis Anzahl Mode Command</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>0x00 … 0x0F</td><td>0x20</td><td>Input Pins:
Value = 0x0000 ➔ Unbekannt (Z.B. Wechselspannung, Pulse, …)
Value = 0x0001 ➔ AUS/OFFEN
Value = 0x0002 ➔ Masse
Value = 0x0004 ➔ Positiv (> 3V3)</td></tr>
</table>

Pin	Type	Verwendung
0x00	0	Spezial Fälle
0x00 … 0xFF	0x40	LED Signal Ausgänge auf Basis PCA9955 (16 Fach konstant Strom) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Auf-/Ab Blendzeit (10mS Unit!!) | PWM
0x00 … 0xFF	0x80 … 0xFF	I2C Adresse im Type 0 … 127 + 0x80, Value gibt Timing und PWM Wert. 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Auf-/Ab Blendzeit (10mS Unit!!) | PWM

StEin Anzeige Logik.
Mit Pin 0x01, Type 0x00 können die beiden Anzeige Digits vom StEin für ‚externe' Anzeigen verwendet werden. Jede Digitbyte ist dazu in 3 High Bits und 5 Low Bits unterteilt.
Die 5 Low Bits ergeben das gewünschte Zeichen: 0 = ‚0', 1 = ‚1', …. 9 = ‚9', 10 = ‚A', 11='b', 12='C', 13='d', 14='E', 15='F', 16='H', 17='h', 18='I', 19='L', 20='n', 21='P', 22='S', 23='U', 24='u', 25='_', 26='-,, 27='-,

Wenn Bit ‚7' bei einem der beiden Digit Bytes gesetzt ist, so wird die ‚remote' Anzeigefunktion beendet (Gilt immer für beide Digits!!!), wenn Bit ‚6' gesetzt ist, blinkt das jeweilige Digit, Bit ‚5' zeigt den Dezimalpunkt an.

HINWEIS, STEIN, SERVO PLATINE:

Pin	Type	Verwendung
		Servo Platine
0x20 … 0x3F 0x40 … 0x5F	0x10	Platine 1: Output Pins 1 … 32 (Nur 8 physische) Platine 2: Output Pins 1 … 32 (Nur 8 physische) Value = 0x0000 ➔ Unbekannt Value = 0x0001 ➔ AUS/OFFEN (N.A.) Value = 0x0002 ➔ Masse Value = 0x0004 ➔ VCC
0x20 … 0x3F 0x40 … 0x5F	0x12	Platine 1: Servo 1 … 32 (Nur 8 physische) Platine 2: Servo 1 … 32 (Nur 8 physische) Value = 0x0000 ➔ unbekannt (Fehler) Value = 0x000n ➔ Stellung 1 … n laut Config Value = 0x1xxx ➔ Servo Analog Position 1 … 256
0x20 … 0x2F 0x40 … 0x4F	0x20	Platine 1: Input Pins Platine 2: Input Pins Value = 0x0000 ➔ Unbekannt (Z.B. Wechselspannung, Pulse, …) Value = 0x0001 ➔ AUS/OFFEN Value = 0x0002 ➔ Masse Value = 0x0004 ➔ Positiv (> 3V3)

HINWEIS, STEIN, I2C EXTENSIONS, LICHTPLATINE:

<table>
<tr><th>Pin</th><th>Type</th><th>Verwendung</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>0x00 … 0xFF</td><td>0x60
0x61</td><td>StEin Servo Erweiterungsplatine 1 [0x60], Erweiterungsplatine 2 [0x61].
Durch diesen Befehl können bis 2x256 (512) Servos angesteuert werden.
Welcher Servo an welchem physischen Anschluss vorhanden ist entscheidet die StEin Config.

Einfachster Fall: Servo fährt an Position 1 … 32 gem. Config (StEin Excel)
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="5">Position</td></tr>
</table>
Servo fährt an Position ‚x‘:
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="10">Position / Winkel</td></tr>
</table>
Servo fährt an Position ‚x‘ (1..32) mit Geschwindigkeit ‚y‘:
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="7">Geschwindigkeit (x10mS)</td><td colspan="5">Position</td></tr>
</table>
</td></tr>
<tr><td>0x00 … 0xFF</td><td>0x64
0x65</td><td>StEin Weichen Erweiterungsplatine 1 [0x64], Erweiterungsplatine 2 [0x65].
Durch diesen Befehl können bis 2x256 (512) Weichen angesteuert werden.
Welche Weiche an welchem physischen Anschluss vorhanden ist entscheidet die StEin Config.</td></tr>
<tr><td>0x00 … 0xFF</td><td>0x70
0x71</td><td>Stein Eingangs Erweiterungsplatine 1 [0x70], Erweiterungsplatine 2 [0x71]
Durch diesen Befehl können bis 2x256 (512) Eingänge abgefragt werden..
Welche Eingang an welchem physischen Anschluss vorhanden ist entscheidet die StEin Config.</td></tr>
</table>

HINWEIS, STEIN, ERWEITERUNGSPLATINEN:

<table>
<tr><td colspan="3">Bedeutung von Pin und Value für StEin Erweiterungsports</td></tr>
<tr><td>Pin</td><td>Verwendung</td><td>Bedeutung von Value</td></tr>
<tr><td>0x00 … 0x3F</td><td>Servo Pins</td><td>Einfachster Fall: Servo fährt an Position 1 … 32 gem. Config (StEin Excel)
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="5">Position</td></tr>
</table>
Servo fährt an Position ‚x‘:
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="10">Position / Winkel</td></tr>
</table>
Servo fährt an Position ‚x‘ (1..32) mit Geschwindigkeit ‚y‘:
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="7">Geschwindigkeit (x10mS)</td><td colspan="6">Position</td></tr>
</table></td></tr>
<tr><td>0x40 … 0x7F</td><td>Digital Outputs</td><td>
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8"></td><td colspan="8">‚0‘=Low, ‚1‘ = High, ‚2‘=Open</td></tr>
</table></td></tr>
<tr><td>0x80 … 0xBF</td><td>Actual Free</td><td></td></tr>
<tr><td>0xC0 … 0xFF</td><td>Digital Inputs</td><td>
<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8"></td><td colspan="8">‚0‘=Low, ‚1‘ = High, ‚2‘=Open</td></tr>
</table></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

ACCESSORY LOCOREQ [0X01.0X08]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x08	0b00		4	ZubehörNID		Pin	0	LocoNId			

Dieses Datagramm existiert nur als Abfrage.
Es wird der Stein mit ZubehörNId, Pin gefragt ob er das Fahrzeug LocoNId ‚kennt‘

Wird als ZubehörNId 0xD000 und Pin 0xFF gesendet, so ist dies eine Broadcast Frage an alle StEin Module und alle Gleisanschlüsse.
Sollte das Fahrzeug in mehreren Abschnitten erkannt sein, so werden alle Abschnitt gemeldet.
Die Antwort erfolgt über die 0x01.0x05 Datagramme.

Hinweis:
Jenes Gerät, welches diesen Request sendet, muss ein Timeout von 500mS implementieren, sollte in dieser Zeit kein Accessory Data Ack empfangen werden, so ist die aktuelle Position des Fahrzeuges unbekannt.

ACCESSORY SIGNAL [0X01.0X0A]

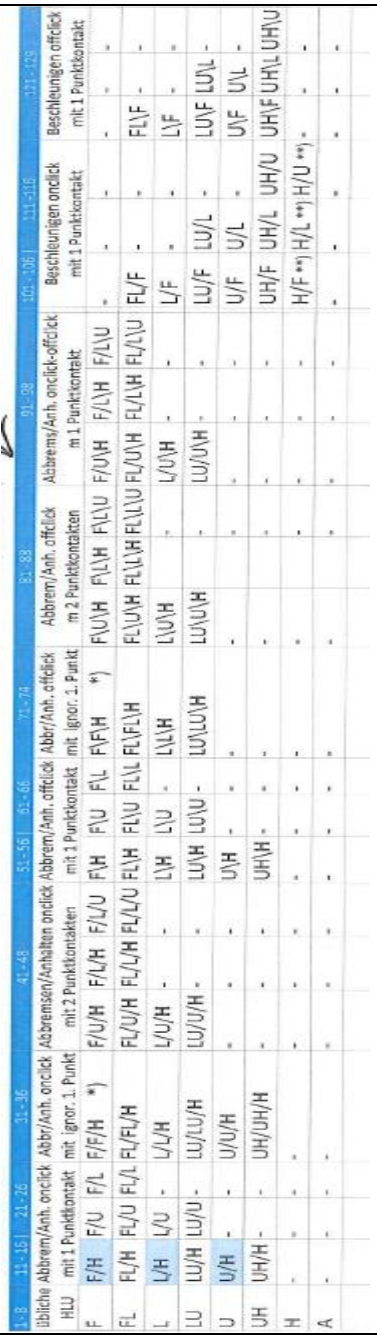
Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x0A	0b1x		8	ZubehörNID		I2C-Adr.	An.Pkt.	CRC32			

HINWEIS:
Dieser Befehl gilt für die StEin ICA Signal Platinen.

Durch M = 0b01, DLC = 6 wird das Signal des Zubehörs (NID) auf den angegeben CRC32 gestellt.
I2C-Adr. ➔ Adresse des I2C Chips, an welchem sich der erste Anschluss Punkt des Signales befindet
An.Pkt. ➔ Anschlusspunkt, Erste ‚rote' Lampe des Signals lt. Excel Config Sheet
CRC32 ➔ Ist der StEin Dokumentation zu entnehmen.

ACCESSORY TRACK MULTILIMIT [0X01.0X09]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x09	0b00		7	AccessoryNId		Track					
0x01	0x09	0b01		8	AccessoryNId		Track	HLU-Code	O/W-Code			
0x01	0x09	0b1x		8	AccessoryNId		Track	HLU-Code	O/W-Code			



Diese Datagramme arbeiten ähnlich dem Pin6 Datagramm, jedoch können bis zu 2 Folge HLU Stufen (Zusätzlich zur ‚Basis' HLU Stufe) und der Ost/West Code an einem Gleisabschnitt gesendet werden.
Die HLU Codes bitte nebenstehender Tabelle entnehmen.

Per Datagramm mit Mode ‚0b00' kann der aktuelle HLU und O/W Code für den jeweiligen Gleisabschnitt (Track) abgefragt werden.

Per Mode ‚0b01' wird ein neuer HLU und O/W Code an das Modul (StEin) gesendet. Dieser wird per Ack (0b10) bestätigt, ebenso Abfragen.

Sobald der StEin im ‚Betrieb' autonom (eben aufgrund der Balisen Meldungen) die HLU Stufen wechselt, erfolgt die Meldung per Event (0b11).

Hinweis:
Wenn der StEin per Multilimit Datagramm [0x01.0x09] gesteuert wird, so:

- Erfolgen die Rückmeldungen ebenfalls durch dieses Datagramm
- Balisen Meldungen sind in dem Falle NICHT mehr vorhanden,
  da ja intern schon abgearbeitet.

OBJECT COMMANDS (0X1N), EINFÜHRUNG MIT STEIN-MODUL

Vorbereitung für die Objektsteuerung, 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 Objektstatus abgefragt bzw. beantwortet werden.

OBJECT COMMAND (0X12)

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x12	0b00		3	ObjNID		Prop.					
0x01	0x12	0b1x		8	ObjNID		Prop.	Object Data				

Vorbereitung für die Objektsteuerung, 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	Gleisabschnittsobjekte
0xE800	Broadcast-Adresse für Ein-/Ausgänge
0xE801 … 0xEFFF	Ein/Ausgangsobjekte

STEIN, GLEISABSCHNITT

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

Property	Verwendung	Data (DB5 … DB8)
0x01	Besetztmeldung	DB5: ‚0' = frei, ‚1' = besetzt DB6: frei DB7, DB8: 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, DB8: Meldungsinhalt
0xF0	Fehler: Überstrom	DB5: 0xFF DB6: frei DB7, DB8: 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, DB8: Ausgangsstrom in mA

ACCESSORY PIN6 [0X01.0X16]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x01	0x16	0b00		4	ModulNr		Pin	Type				
0x01	0x16	0b1x		6	ModulNr		Pin	Type	Value			

Wenn M = 0b00, DLC = 4 dann wird der Zustand des Ein/Ausganges (Pin) vom Zubehör ‚ModulNr' abgefragt.
Durch M = 0b01, DLC = 6 wird der Ausgang des ‚ModulNr' auf den angegeben Wert eingestellt.
Die Abfrage wird durch 0b1x beantwortet, ebenso Änderungen welche durch andere Einflüsse verursacht werden (z.B.: Manuelles Verstellen, Zeitablauf, ...)

Hinweis:
Dies ist wieder ein STP Sonderbefehl.

FAHRZEUG CONTROL GROUP [0X02]

Die Command Group 0x02 fasst alle Fahrzeugsteuerbefehle zusammen und muss von allen Boostern und Fahrpulten implementiert werden. Diese Gruppe enthält jedoch KEINE Programmierbefehle.

FAHRZEUG STATE [0X02.0X00]

Mit diesen Datagrammen kann ein Gerät (Fahrpult, PC-Software, …) den aktuellen Status eines Fahrzeuges abfragen. Dies ist insbesondere sinnvoll um Steuer-Konflikte zu erkennen.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x00	0b00		2	FahrzeugNID							
0x02	0x00	0b11		8	FahrzeugNID		StateFlags (*)		CtrlNID		CtrlTick	
0x02	0x00	0b11		8	FahrzeugNID		Speed		F0 … 15		F16 ... 32	

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 Steuergerät 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.

Hinweis:
Wenn das Fahrzeug dem MX10 unbekannt ist, so sendet es:
Status Flags=0xFFFF, CtrlNId=0xFFFF, CtrlTick=0x0000.
In diesem Falle muss das Steuernde Gerät (PC Programm, Fahrpult, …) die Fahreigenschaften per Mode Command festlegen.

STATE FLAGS

Bit	Info Command
0 .. 7	Consist NId & 0xFF
8	BiDi
9	ZACK
10	Richtungs Anweisung (Client an MX10)
11	Aktuelle Gleisrichtung (MX10 an Client)
12	
13	
14	
15	E-Stopp

[0x02.0x00], Type Definition für Fahrzeug Stat

FAHRZEUG MODE [0X02.0X01]

Diese Datagramme dienen der Abfrage bzw. dem Einstellen der Fahrzeug-Betriebsart.
Damit das MX10 ein Fahrzeug steuern kann, müssen Ihm die Betriebsparameter bekannt sein.

Eine PC-Software kann/muss die Fahrzeug Betriebs-Parameter festlegen!!!
Dazu kann sie entweder diese zuerst abfragen und nur ‚unbekannte' Parameter ergänzen, oder schlicht und einfach ‚Ihre' Parameter rücksichtslos als Command senden.
In jedem Falle arbeitet das MX10 mit den zuletzt definierten (empfangenen) Fahrzeugparametern und speichert diese bei Betriebsende.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x01	0b00		2	FahrzeugNID							
0x02	0x01	0b01		5	FahrzeugNID		M1	M2	M3			
0x02	0x01	0b11		5	FahrzeugNID		M1	M2	M3			

Durch Mode=0b00 kann das Steuergerät die aktuellen Betriebsparameter für ein Fahrzeug abfragen. Bei einer Antwort mit M1=0x00 und M2=0x00 ist dem MX10 das jeweilige Fahrzeug unbekannt.

MODE 1 FLAGS

Bit	Bedeutung
0 .. 3	Speed Steps: 0: 'unbekannt' 1: 14FS 2: 27FS 3: 28FS 4: 128FS 5: 1024FS 6 - 7: nicht definiert
4 .. 7	Schienen Format: 0: unbekannt 1: DCC 2: MM2 3: nicht definiert 4: mfx 7: System Use

MODE 2 FLAGS

Bit	Bedeutung
0 .. 7	Max. Anzahl an Funktionen: Keine (0) bis derzeit max. 32

MODE 3 FLAGS

Bit	Info Command
0	Puls Fx (Funktionen werden also Pulskette gesendet, LGB)
1	Analog Fx (Analog Funktionen)
2 .. 3	Speed Limit ZIMO / Speed Limit NMRA 0b00 = Kein Speed Limit aussenden, 0b01 = NMRA Speed Limit senden 0b10 = ZIMO Speed Limit senden
4	nicht definiert
5	nicht definiert
6	nicht definiert
7	nicht definiert

FAHRZEUG SPEED [0X02.0X02]

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' 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 Anhang 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.

SPEED & FLAGS

Bit	Bedeutung
0 .. 9	Fahrzeug Geschwindigkeit. Hinweis: Für alle Schienenformaten ist der ZIMO Wert 1008 die Max. Fahrstufe, Werte > 1008 werden automatisch auf 1008 skaliert.
10	Richtungs Anweisung (Client an MX10) ‚0' ➔ Vorwärts, ‚1' ➔ Rückwärts
11	Aktuelle Gleisrichtung (MX10 an Client) ‚0' ➔ Vorwärts, ‚1' ➔ Rückwärts
12	
13	
14	
15	Fahrzeug E-Stopp

FAHRZEUG BASIS FUNKTION ABFRAGEN [0X02.0X03]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x03	0b00		2	FahrzeugNID							
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 = 0b11, DLC = 6, dann antwortet die Lok auf eine Statusabfrage.

Dieses Datagramm dient nur der ‚schnellen' Abfrage des aktuellen Fahrzeug Funktionszustandes.
Ein ‚Cmd' (Befehl) ist NICHT vorgesehen!!
Die 32 Bits enthalten 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 [0X02.0X04]

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-Decoder abhängig, für 'normale' DCC und MM Lok Decoder werden diese als Funktion 'Ein' interpretiert.

Die ‚FxNr' ist für diesen Befehl in mehrere Bereiche aufgeteilt:

Von FxNr	Bis FxNr	Beschreibung	Gültige Werte
0	31	Die bekannten ‚normalen' Funktionen, die maximal Fx Nummer ist dabei vom jeweiligen Format abhängig.	Ein/Aus
256	512	256 Analog Funktionen	0 … 255
32768	65536	Binäry States (siehe NMRA 9.2.1)	Ein/Aus

FAHRZEUG SONDER-FUNKTION SCHALTEN [0X02.0X05]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x05	0b00		4	FahrzeugNID		SxNr					
0x02	0x05	0b01		6	FahrzeugNID		SxNr		SxVal			
0x02	0x05	0b11		6	FahrzeugNID		SxNr		SxVal			

Von SxNr	Bis SxNr	Beschreibung	Gültige Werte
1		Manual	0 = ‚AUS' 1 = ‚EIN'
2		Rangierfunktion	0 = ‚AUS' 1 = ‚AzBz' 2 = ‚Half'
3		ZIMO Ost/West Vorgabe	0 = Keine Änderung 1 = Richtung ‚Ost' 2 = Richtung ‚West'

[0x02.0x05], Type Definition für Fahrzeug Special Function

FAHRZEUG AKTIV [0X02.0X10]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x10	0b00		2	Fahrzeug-NID							
0x02	0x10	0b01		4	Fahrzeug-NID		Mode					

Jedes Steuergerät (Fahrpult/Computer) sollte diesen Befehl für ‚aktive' Fahrzeug etwa alle 500 … 1000ms aussenden. Dieses Kommando bewirkt, dass dieses Fahrzeug zumindest in Priorität 1 im MX10 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 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 [0X02.0X11]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x11	0b11		8	Fahrzeug1NId		Fahrzeug2NId		Fahrzeug3NId		Fahrzeug4NId	

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 (siehe MX10 Prioritätslogik) weiter ausgesendet.

FAHRZEUG LAST CONTROLLER [0X02.0X12]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x12	0b00		4	FahrzeugNId		Type					
0x02	0x12	0b01		8	FahrzeugNId		Type		ControllerNId		Sekunden	

Mit diesem Befehl kann ein Gerät abfragen, welches andere Gerät das Fahrzeug zuletzt gesteuert hat.

TYPE

Type	Controller
0	
1	Es wird der ‚aktuellste' Controller gemeldet (Egal ob MX32 oder Computer) ControllerNId ist die NId des jeweiligen Controllers bzw. ‚0' wenn das Fahrzeug bisher nicht gesteuert wurde.
2	Es wird das 'letzte' System Gerät gemeldet (MX10, MX32, StEin) Die ControllerNId ist die NetworkNId jenes MX10, MX32, StEin, welches das Fahrzeug zuletzt gesteuert wurde.
3	Es wird der 'letzte' Computer gemeldet. Die ControllerNId ist die NetworkNId jenes Computers, welches das Fahrzeug zuletzt gesteuert wurde.

In der Antwort wird auch die Zeitspanne seit dem letzten Steuerkommando gemeldet. Diese Zeitspanne in Sekunden ist '0', sofern noch kein Gerät das Fahrzeug gesteuert hat.

FAHRZEUG LÖSCHEN [0X02.0X14]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x02	0x14	0b01		4	ObjectNId		FahrzeugNId		Mode		CheckSum	
0x02	0x14	0b10		4	ObjectNId		FahrzeugNId		State			

Mit diesen Datagrammen kann ein Fahrzeug in der Zentrale gelöscht werden.
Dabei gilt:
ObjectNId ist die NId jener Zentrale, in welcher das Objekt zu löschen ist. Wenn 0xC000 angegeben wird, so gilt dies für alle Zentralen im System.
FahrzeugNId ist die NId jenes Fahrzeuges, welches zu löschen ist.
CheckSum = ObjectNId XOR (NOT FahrzeugNId).

Dieser Befehl wird per ACK NACH Abschluss der Operation in der angegebenen Zentrale bestätigt.
State bedeutet:

- 0x0001 ➔ Gelöscht
- 0xFFFF ➔ Löschen nicht möglich

FREE GROUP [0X03]

Derzeit frei für Protokoll Erweiterungen

TRAIN CONTROL GROUP [0X05]

Diese Datagramm Gruppe dient der Verwaltung von Zügen (Verbund Objekten).
So wie bei allen anderen Gruppen kann die Anzahl der im System gespeicherten Züge über den Group Count (Data Group Datagramm [0x07.00]) abgefragt werden. Ebenso können die NId's der vorhandenen Züge mit den Datagrammen Item List bei Index/NId abgefragt werden.
Züge bestehen aus zumindest 2 Dekodern (typischerweise Fahrzeug Dekoder, keine Funktionsdekoder sind jedoch ebenso möglich). Jeder Zug hat einen ‚Eigentümer', dies kann ein bestimmtes MX32 sein, ein PC oder eine andere Steuereinheit, welche Zugriff auf Fahrzeuge hat.
Das MX10 kann bis zu 256 solcher Züge verwalten und speichern. Jeder Zug darf im Grunde aus allen im MX10 vorhandenen Fahrzeug/Funktionsdekodern bestehen. Sinnvoll sind jedoch nur Züge mit bis zu maximal 16 FAHR Dekodern, da sonst schon alleine die im Gleissignal (DCC) vorhandene (besser nicht vorhandene) Bandbreite zu einem sichtbaren ruckeln der Fahrzeuge führt.

TRAIN PART LIST [0X05.0X01]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x01	0b00		3	ZugNId		Idx					
0x05	0x01	0b11		7	ZugNId		DecoderNId		OwnerNId		Flag	Idx

[0x05.0x01], Type Definition für Zug Fahrzeug Suche

Mit diesem Datagramm können die ‚Teile' eines Zuges abgefragt werden.
Die Antwort enthält die ‚DecoderNId' (Fahrzeug/Funktionsdecoder Adresse) und wer den jeweiligen Zug angelegt hat.

TRAIN PART FIND [0X05.0X02]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x02	0b00		2	DecoderNId							
0x05	0x02	0b11		6	DecoderNId		ZugNId		OwnerNId		State	

[0x05.0x02], Type Definition für Fahrzeug Zug Suche

Mit diesen Datagrammen kann abgefragt werden, ob ein Fahrzeug zu einem Zug (Traktion) gehört.
Wenn das Fahrzeug zu keinem Zug gehört, so ist im ACK die ZugNId = 0xFFFF.
Unter Decoder sind sowohl Fahrzeug Decoder als auch reine Funktionsdecoder zu verstehen.
Die jeweilige NId ist entsprechend der schon definierten Regeln anzugeben.

TRAIN CREATE [0X05.0X04]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x05	0b10		2	ZugNum							
0x05	0x05	0b11		6	ZugNum		OwnerNId		ZugNId			

Durch diese Datagramme kann ein ‚neuer' Zug angelegt werden.

Damit kann ein Bediengerät einen neuen Zug anlegen.
Die ZugNum ist dabei jene Nummer, welche das Endgerät nutzen will, die ZugNId ist Systemweit eindeutig.
Hinweis: Es sind nur ‚CMD' und ‚ACK' erlaubt.

TRAIN OWNER CHANGE [0X05.0X03]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x03	0b10		8	ZugNId		NewOwnerNId		ZugNumFlag		Flag's	
0x05	0x03	0b11		8	ZugNId		NewOwnerNId		ZugNumFlag		Flag's	

[0x05.0x03], Type Definition für Zug 'Eigentümer'

TRAIN PART HEAD [0X05.0X05]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x05	0b00		2	ZugNId							
0x05	0x05	0b10		4	ZugNId		DecoderNId					
0x05	0x05	0b11		4								

Durch diese Datagramme kann:

- Abgefragt werden, welches Fahrzeug den ‚Kopf' des Zuges bildet
- Oder dieser gesetzt bzw. gelöscht werden.

TRAIN PART SET [0X05.0X08]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x08	0b01		4	ZugNId		DecoderNId					
0x05	0x08	0b11		6	ZugNId		DecoderNId		State			

Durch dieses Datagramm wird ein Decoder (Fahrzeug, Funktionsdecoder) einem Zug hinzugefügt.
(Es sind nur ‚CMD' und ‚ACK' erlaubt).

Hinweise:

- Wenn das MX10 den Zug nicht kennt, so wird dieser automatisch angelegt.
- Wenn das MX10 den Decoder nicht kennt, so wird das hinzufügen abgelehnt (State = 0xFFFF)

TRAIN PART DATA [0X05.0X09]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x09	0b00		6	ZugNId		DecoderNId		Type			
0x05	0x09	0b01		8	ZugNId		DecoderNId		Type		Value	
0x05	0x09	0b11		8	ZugNId		DecoderNId		Type		Value	

Mit diesem Datagramm können die Eigenschaften für ein Fahrzeug im Zug abgefragt bzw. geändert werden.

Fahrzeug Eigenschaften (Type):

Type	Beschreibung	Werte
0x0001		
0x0002	‚Eigentümer' des Zuges ist	NId des Eigentümers
0x0003	Zug Nummer des ‚Eigentümers'	Zug Nummer
0x0010	Richtungsverhalten	0 = Richtung 1:1 1 = Richtung ‚verdreht'
0x0011	Relative Geschwindigkeit in Prozent	100 = 100% ➔ Keine Änderung 110 = 110% ➔Fahrstufe +10% 90 = 90% ➔ Fahrstufe -10%
0x10nn	Einstellungen für Funktionen NICHT Implementiert	

TRAIN PART DEL [0X05.0X0F]

Grp	Cmd	D	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x05	0x0F	0b01		6	ZugNId		DecoderNId		0			
0x05	0x0F	0b11		6	ZugNId		DecoderNId		State			

Mit diesen Datagrammen wird ein Fahrzeug aus der Traktion entfernt.

TRAIN STATUS BITS

Diverse Train Datagramme haben ein Status Feld. Dieses ist wie folgt aufgebaut:

Status Bits															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Zugnummer des Eigentümers								Richtungsumkehr	Geschwindigkeitskorrektur: 1 – 100%						

Eine Status 0xFFFF bedeutet in jedem Falle eine negative Antwort.

TRACK SIGNAL ENGINE GROUP [0X06/0X16]

In dieser Gruppe sind alle Schienensignal-Funktionen zusammengefasst.
Insbesondere sind dies die verschiedenen Methoden um Decoder zu programmieren, Firmwareupdates durchzuführen und Soundprojekte zu laden.
Die Gruppen Kennung 0x06 wird CAN Bus intern genutzt, 0x16 hingegen ist für die Verwendung per Ethernet Schnittstelle vorgesehen. Der Hauptunterschied besteht schlicht darin, dass am Ethernet kein 8 Byte Limit, wie am CAN notwendig ist. Dadurch kann eine geeignete PC Software komplexere Befehle nutzen.

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 Schienenformat abhängig.
Bei DCC werden die ersten 1024 Adressen gemäß NMRA in CV#1 (bzw. CV#17, CV#18 bei langen Adressen) 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 TRACK MODE [0X06.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x00	0b00		2	NID		Pin					
0x06	0x00	0b01		4	NID		Pin	Mode				
0x06	0x00	0b1x		4	NID		Pin	Mode	Voltage		Current	

Durch M=0b00 kann der aktuelle Status der Track Signal Engine abgefragt werden.
Der Pin gibt an welcher MX10 Anschluss in den gewünschten Mode geschalten werden soll.

ACHTUNG:
Nicht jeder Pin kann alle Betriebsmodi ausführen.
Durch Mode wird der jeweilige Betriebsmode festgelegt.

Im Acknowledgement ist die jeweilige Spannung (in mV) und der Maximal Strom (in mA) enthalten.

Hinweis:
Genaue Beschreibung der Betriebsmodi fehlt noch.

TSE PROG CLEAR [0X06.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x04	0b01		4	ObjNId		FahrzeugNId		Options			
0x06	0x04	0b11		4	ObjNId		FahrzeugNId		Options			

Mit diesem Befehl (M=0b01) werden die im MX10 gespeicherten CV Werte des entsprechenden Fahrzeuges (FahrzeugNID) gelöscht.

TSE INFO [0X16.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5 .. DB8	DB9	DB10
0x16	0x08	0b1x		8	SysNId		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 einem Nichterfolg 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 Decoder angeschlossen
0xFn		Fehlermeldungen

Hinweis:
Dokumentation der Fehlermeldungen (0xFn) fehlt

TSE PROG READ [0X16.0X08]

Mit den TSE Read Befehlen können CV's aus einem Decoder gelesen werden.
Ob dies per POM (Default) oder Service Mode Befehlen geschehen soll entscheidet der gewählte TSE Mode (Cmd=0x00).

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5 .. DB8	DB9..DB10
0x16	0x08	0b01		8	SysNId		FahrzeugNID		Cfg Num	
0x16	0x08	0b11		10	SysNId		FahrzeugNID		Cfg Num	Val

Das Kommando (0b01) veranlasst, dass die Zentrale (NID) einen ‚Config Read' Befehl an den Schienendecoder (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 PROG WRITE [0X16.0X09]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5 .. DB8	DB9
0x16	0x09	0b01		9	SysNId		FahrzeugNID		Cfg Num	Val
0x16	0x09	0b11		9	SysNId		FahrzeugNID		Cfg Num	Val

Das Kommando (0b01) veranlasst, dass die Zentrale (NID) einen ‚Config Write' Befehl an den Schienendecoder (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 WRITE 16BIT [0X06.0X0D]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x0D	0b01										
0x06	0x0D	0b11		6								

TSE FIND [0X06.0X10]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x10	0b01		8	SysNId		Mode		0			
0x06	0x10	0b10		8	SysNId		Mode		LocoNId		Tick	

Diese Datagramme dienen der ZIMO Abkippsuche.

SysNId gibt an welches Gerät die Abkippsuche startet, und somit in weitere Folge der Empfänger ist.
Mode = ‚0' Startet die Suche.

‚Erkannte' Dekoder werden per Event gemeldet. Der Mode hat die Werte ‚1 …. X', ist eigentlich eine fortlaufende Nummer. LocoNId ist die erkannte Dekoder Adresse, Tick der Zeitstempel. ZIMO Dekoder sendet hier derzeit Werte mit 250mS Auflösung.

Mode	Funktion
0x100	Bestandssuche

TSE BIDI LOGON CONTROL [0X06.0X11]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x11	0b01		4	Action		Count					
0x06	0x11	0b10		4	Action		Count					

Action	Funktion
0x0100	Start, Fortlaufend, mit System Default Settings
0x0101	Start, Count: Anzahl der Aussendungen, Intervall [x100mS] Zeitabstand zwischen Aussendung
0x0000	Stop

TSE BIDI LOGON RESULT [0X06.0X12]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x12	0b10		8	Ctrl[4] Vendor[12]		LocoNId		Decoder UId			

In DB1, 2 ist ein Control Code und der Hersteller hinterlegt.
Die Hersteller Nummer wird in den unteren 12 Bits gesendet, der Control Code in den obersten 4 Bit.
Control Code ‚0': RailCom Logon erkannt (LocoNId = 0)
Control Code ‚1': RailCom Short Info empfangen (LocoNId = Wunschadresse)

TSE BIDI LOGON ASSIGN [0X06.0X13]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x12	0b10		8								

TSE BIDI DECODER GUI [0X06.0X14]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x14	0b01		4	LocoNId		Action					
0x06	0x14	0x11		4	LocoNId		Action/Error					
0x06	0x14	0x10		4..	LocoNId		Error					

Mit dem Befehl (Mode=0b10) startet das MX10 die Decoder GUI Abfrage.
Sofern der Befehl im MX10 implementiert ist und der angesprochene Decoder über RailCom verfügt, antwortet das MX10 mit einem ACK und der gewünschten Action, ansonsten mit einem Error Code.
Während dem Transfer sendet das MX10 eventuell Events mit einem Error Code, sofern ein Fehler vorkommt.
Hinweis:
Die Fehler Code Liste kann ergänzt werden, sofern in der Praxis weitere Fehlersituationen auftauchen.
Alle Fehler Codes starten mit 0xFnnn, somit kann eine Anwendung diese auch einfach erkennen.

Action	**Funktion**
0x0100	Abfrage ZIMO GUI Version 1 (Dont Use)
0x0101	Abfrage ZIMO GUI Version 2
0x0201	Abfrage RCN 218 Decoder GUI
Error	
0xF000	Anforderung aktuell nicht ausführbar, z.B.: MX10 führt schon mehrere abfragen aus, RailCom Empfang erfordert aktuell zu viel Prozessorleistung, …
0xF001	No Decoder RailCom
0xF010	Timeout für Init (Decoder reagiert nicht auf Abfrage Befehl)
0xF011	Timeout für ‚Daten' (Entweder RailCom Empfangsproblem, oder Decoder sendet ‚Unsinn')

TSE BIDI DECODER GUI PROGRESS [0X06.0X15]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x14	0b10		8	LocoNId		Type	Progress				

Mit diesem Datagramm vermeldet das MX10 den Fortschritt bei der Decoder GUI Abfrage.
Diese Datagramm wird vom MX10 ‚regelmäßig' versendet. Sofern der Transfer ‚normal' verläuft erfolgt dies etwa alle 16 DCC Datagramme, welche an den jeweiligen Decoder gesendet werden.

Progress	
Bit	**Inhalt**
0..11	Anzahl Tx Pakete (Gesendete DCC Pakete an die angegebene Adresse)
12..23	Anzahl Rx Pakete (Anzahl der empfangenen RailCom Pakete von der angegeben Adresse)
24..31	CRC lt. Decoder
32..39	CRC lt. Berechnung

TSE BIDI RAW DATA, BROADCAST [0X06.0X1D]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x1D	0b10		5	Loco/AccNId		Pin	Data				

Mit diesem Datagramm meldet das MX10 BiDi Roh Daten, welche sich im Broadcast Channel befinden.
Die AbsendeNId im Header gibt an, welches Modul die Daten empfangen hat.
Bei Modulen mit mehreren Pins, ist im Feld Pin anzugeben, welches der Pins die Daten empfangen hat.

TSE BIDI RAW DATA, DATA CHANNEL [0X06.0X1E]

Grp	Cmd	M	ID	DLC	DB1/DB2	DB3	DB4		DB5	DB6	DB7	DB8
0x06	0x1E	0b10		8	Loco/AccNId	Pin	L	N1	Data			
								1 .. 4	5, 6, 7, 8, …. 36			

Mit diesem Datagramm meldet das System (MX10, StEin, …) RailCom Roh Daten, welche sich im Data Channel befinden.
Die AbsendeNId im Header gibt an, welches Modul die Daten empfangen hat.
Loco/Acc NId gibt an wer die RailCom Daten gesendet hat (Fahrzeug bzw. Zubehör Dekoder)
Bei Modulen mit mehreren Pins, ist im Feld Pin anzugeben, welches der Pins die Daten empfangen hat.

Im oberen Nibbel von DB4 (L) befindet sich die Anzahl GÜLTIGER 6 Bit Gruppen,
Diese ergeben sich aus der RailCom Bit Anzahl / 6 (abgerundet).
Somit sind folgende Werte gültig:

12	Bit RailCom Datagramm	➔ 2
18	Bit RailCom Datagramm	➔ 3
24	Bit RailCom Datagramm	➔ 4
36	Bit RailCom Datagramm	➔ 6
12+12	Bit RailCom Datagramm	➔ 4
12+18	Bit RailCom Datagramm	➔ 5
12+24	Bit RailCom Datagramm	➔ 6
18+18	Bit RailCom Datagramm	➔ 6
12+12 +12	Bit RailCom Datagramm	➔ 6

Das untere Nibbel von DB4 (N1) enthält das erste RailCom Roh Daten Nibbel, welches die erste RailCom Id darstellt.
Alle weiteren RailCom Roh Daten Bits werden von links nach rechts in den Datenbytes 5, 6, 7, 8 eingefügt.
Alle von RailCom nicht benutzten Bits sind mit ‚0' zu senden.

Beispiele:
12 Bit Datagramm:
L=2, N1 enthält die Id, DB5 die 8 Datenbits (N1 + DB5 ➔ 12 Bit).
DB6, DB7, DB8 sind mit 0 aufzufüllen.

36 Bit Datagramm:
L=6, N1 enthält die Id, DB5…8 die restlichen 32 Bit der gesamten 36 Bit Meldung (N1+DB5, 6, 7, 8)

TSE BIDI RAW DATA, ACK/NACK [0X06.0X1F]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x06	0x1F	0b10		8	Loco/AccNId		Pin	[N]ACK				

Mit diesem Datagramm meldet das MX10 BiDi ACK's bzw. NACK's.
Allerdings sind reine ACK's bzw. NACK's gemäß der aktuellen RailCom Spezifikation praktisch irrrelevant.
Wenn ein Decoder RailCom Datagramme sendet, so hat der damit schon implizit auch ein ‚ACK' gesendet.
Ein dezidiertes NACK wurde per 2015.03.19 aus der Spezifikation entfernt.

DATA GROUP [0X07]

GROUP COUNT [0X07.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x00	0b00		4	SrcNID		ObjType					
0x07	0x00	0b11		4	ObjType		Count					

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

Anmerkung: ScrNID ist die Sourcequellen-NID.

Group Count für MX8, MX9 Module liefert bei unbekannter Anzahl (z.B. weil die Anfrage zu früh, Autoscan off, Rückmeldefehler liegt vor,…) das Ergebnis 0xFFFF.

OBJTYPE CODES:

ObjType	
0x0000	Fahrzeuge
0x2F00	Züge
0x3000	Zubehör, DCC ‚simpel'
0x3200	DCC ‚eXtended' Zubehördecoder
0x5040	MX8 Module
0x5080	MX9 Module
0x9100	*Anzahl I2C Adressen am I2C Bus gesamt*
0x910n	*Anzahl I2C Adressen am I2C Bus 'n' (n=1 … 16max)*

ITEM LIST BY INDEX [0X07.0X01]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x01	0b00		6	SrcNId		GroupNId		Index			
0x07	0x01	0b11		6	Index		NID		ItemState			
0x07	0x01	0b11		8	Index		NID		ItemState		LastTick	

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

Die SrcNId ist jene NId, welche Antworten soll, typischerweise ist das die Zentrale (MX10), könnte aber auch ein Booster, Fahrpult sein.

Fall 1: Gerät vorhanden
Wenn unter dem abgefragten Index ein Objekt im MX10 vorhanden ist, so liefert es in der Antwort den Index, die NId des Gerätes und die Anzahl der mS seit der letzten Kommunikation mit dem Gerät.

Fall 2: Gerät nicht vorhanden/unbekannt
Wenn das MX10 unter dem angegeben Index kein Gerät kennt, so liefert es in der Antwort den abgefragten Index, als NId=0xFFFF und ebenso als letzten Kommunikation-Tick 0xFFFF.

Ebenso, wenn der Index außerhalb der ‚Objektgruppe' liegt (z.B.: Bei MX8/MX9 sind nur Indexe von 0 … 63) erlaubt), oder die ‚Objektgruppe' als solches unbekannt ist.

STEIN ERWEITERUNGEN:

Um die Adressen der StEin Erweiterungen (I2C) zu finden gilt:
SrcNId=StEin NId, GroupNId= 0x4800, Index=1 … 128.
Ansonsten arbeitet dieser Befehl wie im MX10, die Antworten erfolgen vom jeweiligen StEin Modul

ITEM LIST BY NID [0X07.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x02	0b00		4	SrcNID		NId					
0x07	0x02	0b11		6	NId		Index		ItemState			
0x07	0x02	0b11		8	NId		Index		ItemState		LastTick	

Durch M = 0b00 kann ein Gerät jene NId abfragen, welche nach der angegeben NId gespeichert ist.
Dieser Befehl ist insbesondere für Zubehör Module/Decoder hilfreich.
Die Antwort (M = 0b11) enthält die ‚nächste' NId, den jeweiligen Index und sofern vorhanden den letzten ‚Kommunikationstick'.

Die SrcNId ist jene NId, welche Antworten soll, typischerweise ist das die Zentrale (MX10), könnte aber auch ein Booster, Fahrpult sein.

Ähnlich wie bei ‚Item List by Index' [0x07.0x01] gibt es auch hier 2 Antwortmöglichkeiten:

1. Das MX10 findet ein ‚nächstes' Gerät nach der angegeben NId in der gleichen Objektgruppe. In dem Falle liefert es die gefundene NId, den Index und die letzten Kommunikationstick.
2. Dem MX10 sind keine weiteren Geräte in der Objektgruppe bekannt, die NId verweist auf eine unbekannte Objektgruppe, … In dem Falle antwortet das MX10 mit NId=0xFFFF, Index=0xFFFF und LastTick=0xFFFF.

HINWEIS ZU VERWENDUNG VON 0X07.0X01/0X07.0X02:

Beide Befehle erfüllen sehr ähnliche Aufgaben und liefern auch ähnliche Antworten.
Mit ‚Item List by Index' wird jedoch ein ‚direkter' Speicher Zugriff ausgeführt. An der jeweiligen Speicherstelle können sich Daten (ein Objekt) befinden oder auch nicht. Wenn eine Abfrage auf Index z.B.: Index 10 ‚keine Daten' liefert, so können bei Index 11 durchaus noch welche vorhanden sein.
Mit ‚Item List by NId', liefert das MX10 solange ‚positive' Antworten, wie es weitere Daten findet. ‚Leere' Speicherplätze werden dabei übersprungen.
Die Antwort (M = 0b11) enthält die ‚nächste' NId, den jeweiligen Index und sofern vorhanden den letzten ‚Kommunikationstick'.

ITEM STATE:

Bit		
0 .. 3	Track Format	
4 .. 7	Track Steps	
8	Direction Cmd	
9	Direction Ack	
10 .. 19	Speed	
20 .. 21	East/West	
22	BiDi	
23	ZACK	
24	E-Stop	
25	Cab E/W	
26	Train	
27 .. 30	Free	
31	Deleted	

SUB-ITEM LIST BY NID [0X07.0X03]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x02	0b00		6	NId		Sub-Part					
0x07	0x02	0b11		8	NId		Sub-Part		Type		Data1	Data2

Durch M = 0b00 können ‚Teil' Funktionen eines Gerätes abgefragt werden.
Dies ist z.B.: Sinnvoll um die I2C Platinen eines StEin Modules abzufragen, oder um die Teilnehmer (Dekoder) eines Verbundes abzufragen.

NId ist die Network Id des abgefragten Moduls (z.B.: StEin)
Sub-Part ist der ‚Unter' Modul/Teil (z.B.: I2C Platine)

Für die StEin Module gilt:

Bus	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Intern	0	0	0	0	0	0	0	0	I2C Adresse							
Extern 1	0	0	0	0	0	1	0	0	I2C Adresse							
Buchse 1	0	0	0	0	1	0	0	0	I2C Adresse							
Buchse 2	0	0	0	0	1	1	0	0	I2C Adresse							

DATA CONTROL [0X07.0X06]

Dieses Datagramm dient der Steuerung der (anschließenden) Daten Transfers

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x06	0b01		6	SrcNID		NID		Action			
0x07	0x06	0b11		6	SrcNID		NID		Action		Status	

Der ‚Absender' legt per Command fest, welche Datenoperation er ausführen möchte.
Diese Operation bezieht sich immer auf ein Objekt (NID) in der Datenbank eines bestimmten Gerätes (SrcNID).
In erster Linie handelt es sich dabei um die Daten, welche ein MX10 verwaltet, allerdings erlaubt dieses Datagramm auch den direkten Zugriff auf Daten in den MX32ern oder anderen Geräten mit eigener Datenverwaltung (App's).

Das angesprochene Gerät beantwortet (ACKed) mit einem Status Code.

DATA CONTROL ACTIONS

Action	Beschreibung	Status
0x1000	Data Transfer Open	0x0000: Ok 0xFFFF: Abgelehnt
0x1F00	Data Transfer Close	0x0000: Ok 0xFFFF: Abgelehnt
0x2000	Data Group 1 Item Bitflags für die vorhandenen Daten in Gruppe 1	Siehe Data Group 1

DATA GROUP 1 ITEM

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

DATA FLAGS [0X07.0X07]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x07	0b00		6	SrcNID		NID		Type			
0x07	0x07	0b11		8	SrcNID		NID		Type		Flags	

DATA VALUE [0X07.0X08]

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		6	SrcNID		NID		Type			
0x07	0x05	0b01		8	NID		Type		Value			
0x07	0x05	0b11		8	NID		Type		Value			

Die ‚SrcNID' (Source/Quell Net ID) gibt an welches Gerät die Frage beantworten soll. Typischerweise ist die die NID jener Zentrale, mit welcher der PC verbunden ist. Kann aber vor allem in einem Netzwerk auch ein anderes Gerät (z.B.: Tab, anderer PC) sein.
Mit der ‚NID' und dem ‚Type' wird angegeben, für welches Gerät und welche Art von Daten abgefragt werden.

DATA VALUE TYPES FÜR FAHRZEUGE

Type	Beschreibung	
0x0010	Gruppe	Value: 0 … 15, Festlegung in Pult Conf
0x0011	Epoche	
0x0012	Antrieb	Value=0b00000000: unbekannt Value=0b00000001: Dampf Value=0b00000010: Diesel Value=0b00000100: Elektro Value=0b00010000: Getriebe Mechanisch Value=0b00100000: Getriebe Hydraulisch Value=0b00110000: Getriebe Elektrisch
0x0013	Land	Value: 2 Zeichen ISO Code z.B.: Österreich = ‚AT' ➔ 0x41540000 Deutschland = ‚DE' ➔ 0x44450000
0x0014	Gesellschaft	Value: CRC32 über Name

DATA VALUE TYPES FÜR ZUBEHÖR

Type	Beschreibung
0x0010	Gruppe

DATA NAME (0X10)

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

Dieser Befehl dient nur der CAN internen Kommunikation.
Für die PC Kommunikation ist der Befehl ‚Data Name eXtended' zu verwenden.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x10	0b00		6	NID		SrcNID		Pin	Pos		
0x07	0x10	0b01		8	NID		Pin	Pos	Z1	Z2	Z3	Z4
0x07	0x10	0b11		8	NID		Pin	Pos	Z1	Z2	Z3	Z4

Durch M = 0b00 kann ein Gerät vom Gerät ‚SrcNID' den Namen vom Gerät ‚NID' abfragen. Mit ‚Pos' gibt es an ab welcher Position es die nächste Zeichengruppe haben will.
Durch M = 0b01 kann ein Gerät einen neuen Namen für das Gerät ‚NID' festlegen, dieser wird dann gesendet.
Eine Abfrage (M=0b00) oder Namensänderung (M=0b01) wird durch M=0b11 beantwortet.

ITEM IMAGE CONFIG (0X12)

Mit den ‚Data Image Config' Datagrammen können für jedes Fahrzeug, Zubehör, Gerät im System die Bilder übertragen werden.

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.

Hinweis:
Fahrzeug Bilder können auch per ImageCRC32 abgerufen bzw. festgelegt werden, die Umsetzung auf die ImageID muss im jeweiligen Gerät berechnet werden. Die ‚Quelle' für die ImageCRC32 ist dabei immer das ‚große' Fahrzeugbild (279x92 Pixel, 16 (wenn MX32/FU berechnet) bzw 24 (wenn PC berechnet) Bit Farben).

IMAGE TYPES

Type	Beschreibung
0x01	Fahrzeug Bild: Image ID
0x02	Fahrzeug Bild: Image CRC32
0x03	Fahrzeug Tacho: ImageID
0x04	
0x10	Fahrzeug Instrument: Bremsbalken
0x1F	Fahrzeug Instrument:

(FAHRZEUG) FUNKTION MODES [0X07.0X14]

Mit diesem Datagramm können die Modi der Funktionstasten blockweise abgefragt und gesetzt werden.
Für die typischen 28 Funktionen reichen somit 4 Transfers (2x Req/Cmd + 2xAck).
Sollten zusätzliche Parameter erforderlich sein (z.B.: Zeit) müssen die per Detail Req/Cmd Angefragt bzw. Gesetzt werden.
Ebenso muss im Falle des Modes 0b11 der exakte Mode und eventuell nötige Parameter transferiert werden.
Hinweis:
Per UDP Protokoll können alle 64 Funktionen inkl. des 1. Parameters mit dem Datagramm 0x17.0x14 übertragen werden.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x14	0b00		3	FahrzeugNID		Grp					
0x07	0x14	0b01		8	FahrzeugNID		Grp	0	0 … 3 16 ... 19	4 … 7 20 … 23	8 … 11 24 … 27	12 … 15 28 … 31
0x07	0x14	0b11		8	FahrzeugNID		Grp	0	0 … 3 16 ... 19	4 … 7 20 … 23	8 … 11 24 … 27	12 … 15 28 … 31

Durch das Datagramm M = 0b00, DLC = 3, wird der Funktionsmode für das Fahrzeug ‚FahrzeugNId' und der Gruppe ‚Grp' abgefragt. Mit dem Datagramm M = 0b00, DLC = 8, werden die Funktionsmodi für das für die Tastengruppe ‚Grp' festgelegt. Eine Abfrage wird per Datagramm M=0b11, DLC=8 beantwortet.

Funktionstasten Gruppe:
0 = Tasten 0 … 15
1 = Tasten 16 ... 31
2 = Tasten 32 … 47
3 = Tasten 48 … 63

Die Datenbyte's 4 … 8 enthalten für jede Taste jeweils 2 Bit der den Funktionsmode darstellt. Dabei bedeutet:
0b00: Einfache Schaltfunktion (Ein/Aus)
0b01: Moment Taste
0b10: Auto Timeout, Zeit muss über Datagramm 0x02.0x09 Abgerufen werden.
0b11: eXtended Function, Details müssen über Datagramme 0x02.0x09 Abgerufen werden.

(FAHRZEUG) FX INFO [0X07.0X15]

Detail Einstellungen je Funktionstaste

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x15	0b00		8	(Fahrzeug)NId		FxNum		Base	Type	SrcNId	
0x07	0x15	0b01		8	(Fahrzeug)NId		FxNum		Base	Type	Data	
0x07	0x15	0b11		8	(Fahrzeug)NId		FxNum		Base	Type	Data	

Details noch unklar.

DATA SPEED MAX [0X07.0X18]

Mit den ‚Data Speed' Datagrammen können für jedes Fahrzeug, Zubehörteil, Gerät im System die Geschwindigkeiten festgelegt werden. Wobei dies natürlich für Fahrzeuge wesentlich ist, bei Zubehör- oder Systemgeräten wird dies eher keinen Sinn machen.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x18	0b00		6	SrcNID		Ziel-NID		Type	Entry		
0x07	0x18	0b01		6	NID		Type	Entry	Value			
0x07	0x18	0b11		6	NID		Type	Entry	Value			

Durch M = 0b00 kann ein Gerät von der Quelle (SrcNID, typischerweise die NID der jeweiligen Zentrale) die für Ziel-NID gespeicherten Speed Daten abfragen.
Mit M = 0b01 werden die ‚Speed Daten' gesendet, jeder Empfänger muss selbständig entscheiden ob die jeweiligen Daten für Ihn relevant sind.
Mit M=0b11 wird jede Anfrage beantwortet.

SPEED TYPES

Type	Beschreibung
0x0010	V.max für normale Fahrt Vorwärts
0x0020	V.max für normale Fahrt Rückwärts
0x0030	V.max für Rangierfahrt
0x0080	System Beschleunigung
0x0081	System Verzögerung

DATA SPEED TAB [0X07.0X19]

Diese Datagramme dienen dem abbilden einer Geschwindigkeitkurve.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x18	0b00		6	SrcNID		Ziel-NID		Type	Entry		
0x07	0x18	0b01		8	NID		Type	Entry	Step		Speed	
0x07	0x18	0b11		8	NID		Type	Entry	Step		Speed	

Durch M = 0b00 kann ein Gerät von der Quelle (SrcNID, typischerweise die NID der jeweiligen Zentrale) die für Ziel-NID gespeicherten Speed Daten abfragen.
Mit M = 0b01 werden die ‚Speed Daten' gesendet, jeder Empfänger muss selbständig entscheiden ob die jeweiligen Daten für Ihn relevant sind.
Mit M=0b11 wird jede Anfrage beantwortet.

SPEED TYPES

Type	Entry	Beschreibung
0x01	0x00 … 0x0F	normale Fahrt Vorwärts: Jeder Entry enthält ein Step/Speed Pärchen Dabei bedeutet ‚Step' die jeweilige Fahrstufe im 1024 (ZIMO) Format, Speed der jeweils dazu passende Wert für kmh oder mph. Die Interpretation ob kmh oder mph muss das jeweilige GUI Gerät vornehmen.
0x02		normale Fahrt Rückwärts
0x03		Rangierfahrt
0x08	0x00 … 0x0F	RailCom Tabelle Vorwärts
0x09	0x00 … 0x0F	RailCom Tabelle Vorwärts

DATA SAVE [0X07.0X1A]

Diese Datagramme speichert die aktuellen GUI/Speed Tab Daten.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x18	0b10		8	NID		Option					
0x07	0x18	0b11		8	NID		Option					

NId: Object NId, normaly a Loco
Option: 0x01➔ Save to System Memory
Option: 0x02➔ Save to Lan Memory

DATA BLAUE NADEL [0X07.0X20]

Diese Datagramme dienen dem abbilden einer Geschwindigkeitkurve.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x20	0b00		5	SrcNID		Loco-NId		Idx			
0x07	0x20	0b01		8	NID		Idx	W1	W2	W3	W4	W5
0x07	0x20	0b11		8	NID		Idx	W1	W2	W3	W4	W5

Neues Datagramm für ‚Blaue' Nadel Tabelle.

Damit kann ein Gerät die 16 Werte für die Blaue Nadel Tabelle abfragen bzw. festlegen.
Idx = 1: Werte 1, 2, 3, 4, 5
Idx = 2: Werte 6, 7, 8, 9, 10
Idx = 3: Werte 11, 12, 13, 14, 15
Idx = 4: Wert 16

Die Werte sind gemäß der ‚alten' LENZ kmh Logik kodiert

DATA CLEAR [0X07.0X1F]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x1F	0b01		4	SrcNID		NId					
0x07	0x1F	0b11		4	NId		State					

Durch M = 0b01 kann ein Gerät Daten im angegeben Gerät löschen.
Die ‚SrcNId' gibt an welches Gerät das Object löschen soll.

Dieser Befehl setzt unmittelbar das Löschkennzeichen, dies arbeitet vergleichbar mit dem File Löschen bei einem PC.

PC ONLY: DATA NAME EXTENDED (0X21)

Dieser Befehl steht nur am PC Interface zur Verfügung (USB/LAN).
Damit kann eine App Texte mit bis zu 192 Zeichen in einem Befehl übertragen.
Etliche Einträge sind aber mit 32 Zeichen limitiert, bzw. gibt es in der GUI-Darstellung Limitierungen.
Hinweis: Namen und andere Zeichenketten sind 0x00 Terminiert zu senden!

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 7 .. 10	DB 11 .. 14
0x07	0x21	0b00		14	SrcID	NID	SubID	Value 1	Value 2

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 8	DB 9 .. 12	DB 13 .. 204
0x07	0x21	0b01		12 ..	NID	SubID	Value 1	Value 2	Z1 .. Z[x]
0x07	0x21	0b11		12 ..	NID	SubID	Value 1	Value 2	Z1 .. Z[x]

Die ‚NID' gibt an für welches Gerät der Text gilt.
Wenn ‚NID' z.B.: die NID eines Fahrzeuges ist, so werden die Texte mit diesem Fahrzeug verknüpft.
Auch alle anderen Texte können mit diesem Befehl übertragen werden.

NID	SubID	Value 1	Value 2	Verwendung	Max. Länge
Fahrzeug	0	0	0	Fahrzeugname	32 Zeichen
	1	0	0	Bahngesellschaft	32 Zeichen
Zubehör	Pin	0	0	Bezeichnung für Anschluss	
0x7F00	Vendor	0	0	Hersteller Namen	32 Zeichen
0x7F02	Decoder	0	0	Decoder Name/Typen	32 Zeichen, GUI MX32 max. Platz für 8 Zeichen
0x7F04	CfgName	Type (8Bit), CgfNum (24Bit)	0	CV Bezeichnung	32 Zeichen
0x7F06	CfgDb			CfgDb	32 Zeichen
0x7F10	Icon			Icon	32 Zeichen
0x7F11	Icon			Icon	32 Zeichen
0x7F18	ZIMO Partner			ZIMO Partner	32 Zeichen
0x7F20	Land			Land	32 Zeichen
0x7F21	Gesellschaft			Bahngesellschaft	32 Zeichen
		Type, CgfNum		CV Bezeichnung Vendor = Herstellernummer lt. NMRA Type = Decodertyp (8Bit, Wert der CV#250) CfgNum = CV-Nummer (24Bit)	
0xC2nn				Bezeichnung für PC Software	32 Zeichen

Hinweis:
Value2 ist für Gruppen gedacht.

INFO / CONFIG GROUP [0X08]

In der Info Group sind diverse Informationsabfragen und -meldungen zusammengefasst.

MODUL POWER INFO [0X08.0X00]

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

Pin:

Wert	Schiene / Booster
1	Schiene 1
2	Schiene 2
3 …	Booster 1 …

Status:

Bit	
0	‚1' = AUS
1	‚1' = Unterspannung
2	‚1' = Überstrom
3	‚1' = Netzteil Spannung/Strom
4 … 7	‚0' = Run, ‚1' = SSP, ‚2' = Service Mode ‚3' = frei ‚4' = Decoder Update ‚5' = Sound Laden
8 … 9	Frei
10	Zugnummern Impulse
11	RailCom®
12	mfx®

LOCO INFO [0X08.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x04	0b00		2	Ziel-NID							
0x08	0x04	0b10		6	Ziel-NID		Cmd Count		BiDi Count			
0x08	0x04	0b11		6	Ziel-NID		Cmd Count		BiDi Count			

Mit diesen Datagrammen können die aktuellen Fahrzeug Statistik Werte abgefragt werden.
Ebenso wird jede Fahrzeug Aktiv Meldung (Grp=0x02, Cmd=0x10) mit dem Loco Info ACK beantwortet.

BIDI INFO [0X08.0X05]

Grp	Cmd	M	ID		DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x04	0b00			4	SrcNId		Loco/AccNId		Type			
0x08	0x04	0b10			8	Loco/AccNId		Type		Data			
0x08	0x04	0b11			8	Loco/AccNId		Type		Data			
0x08	0x04	0b10			8	Loco/AccNId		Type		DataH		DataL	
0x08	0x04	0b11			8	Loco/AccNId		Type		DataH		DataL	

Mit diesem Datagramm meldet das MX10 BiDi Informationen
Die ‚SrcNID' (Source/Quell Net ID) gibt an welches Gerät die Frage beantworten soll. Typischerweise ist die die NID jener Zentrale, mit welcher der PC verbunden ist. Kann aber vor allem in einem Netzwerk auch ein anderes Gerät (z.B.: Tab, anderer PC) sein.

Type:

Type	Name	Data
0x0100	BiDi Speed	DataH ➔ Geschwindigkeit in kmh
0x0101	Tilt/Curve	
0x0200	BiDi CV	
0x0300	BiDi QoS	DataH ➔ Quality of Service
0x0400	BiDi Füllstand	DataH ➔ Füllstand in %
0x0800	BiDi Direction	
0x1000	BiDi Track Voltage	DataH ➔ Schienenspannung in 100mV Einheit
0x1100	BiDi Alarms	Bit 0: Alarm Motor Bit 1: Alarm Ausgänge Bit 2: Alarm Temperatur Bit 3 … 15: Unbekannt/Ungenutzt

ZACK INFO [0X08.0X06]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x04	0b11		6	Ziel-NID		Type		Value			

Mit diesem Datagramm meldet das MX10 ZACK Informationen

MODUL INFO [0X08.0X08]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x08	0b00		4	NID		Type					
0x08	0x08	0b01		8	NID		Type		Info			
0x08	0x08	0b11		8	NID		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 Read Only/Write 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	Info Byte 1 = Tag Info Byte 2 = Monat Info Byte 3/4 = Jahr
4	Software Build Time	RO	Info Byte 1 = ‚0' Info Byte 2 = Sekunde Info Byte 3 = Minute Info Byte 4 = Stunde
5	Realtime Clock Date	RW	Info Byte 1 = Tag Info Byte 2 = Monat Info Byte 3/4 = Jahr
6	Realtime Clock Time	RW	Info Byte 1 = ‚0' Info Byte 2 = Sekunde Info Byte 3 = Minute Info Byte 4 = Stunde
7	Frei		
8	MiWi Hardware Version	RO	
9	MiWi Software Version	RO	
10	MiWi Channel Zentrale	RO	Aktueller MiWi Kanal der Zentrale Kommt auch ungefragt als ‚ACK', wenn an der Zentrale geändert
20	Modul Nummer	RO	‚Logische' Nummer des Moduls
100	Modul Art	RW	Info Byte 1/2: 0x2105 ➔ Booster 10806 Info Byte 1/2: 0x2106 ➔ Booster 10807 Info Byte 1/2: 0x9001 ➔ Stein, Version 1 Info Byte 1/2: 0x9002 ➔ Stein, Version 2 Info Byte 1/2: 0x9201 ➔ RoCo Detector
		StEin	
0x0201	*Aktive Config: Gleisabschnitte*	*RO*	*Nummer der aktiven Config im StEin für Gleisabschnitte*
0x0202	*Aktive Config: Weichen*	*RO*	*Nummer der aktiven Config im StEin für Weichen*
0x0203	*Aktive Config: Signale*	*RO*	*Nummer der aktiven Config im StEin für Signale*
0x021x	*Aktive Module Anschluss 1*	*RO*	Info Byte 1: Anzahl Erweiterungsplatinen Info Byte 2: Aktuelle Erweiterungsplatine (Sollte ='X') Info Byte 3/4: 0x9100: Servo Platine Typ 1
0x022x	*Aktive Module Anschluss 2*	*RO*	Info Byte 1: Anzahl Erweiterungsplatinen Info Byte 2: Aktuelle Erweiterungsplatine (Sollte ='X') Info Byte 3/4: 0x9100: Servo Platine Typ 1

Hinweis:
Beschreibung Hardware und Softwareversion ergänzen.

SYSTEM MODUL CONFIG

Die Grundidee der System Modul Config besteht darin, das ein ‚Nutzer' gar nicht wissen muss, welche Einstelloptionen das jeweilige Gerät hat. Per 0x08.0x0C Datagramm[en] kann der ‚Nutzer' alle Einstellmöglichkeiten des jeweiligen Gerätes abrufen und sich so seine ‚eigene' Liste an Optionen und Einstellungen aufbauen.

Ein PC per LAN Anbindung bekommt dazu nicht nur den Einstell Code (Schlüssel) sondern auch:

- Min./Max. Werte
- Einheit
- Namen / Bezeichnung

Natürlich sind eine Reihe von Einstellwerten ‚fix' definiert und können auch ohne Abfrage der Schlüssel Codes genutzt werden. Aufgrund der stetigen Weiterentwicklung SOLLTE!! Eine PC-Software trotzdem immer den aktuellen Katalog (Liste) aus dem System abfragen.

Folgendes sollte in der ZCan Doku ergänzt werden.

MODUL CONFIG [0X08.0X0A]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x0A	0b00		6	NId		Tag					
0x08	0x0A	0b01		8	NId		Tag				Value	
0x08	0x0A	0b11		8	NId		Tag				Value	

Per Modul Config Datagramm können die Parameter der ZIMO Module abgefragt (Req) und geändert (Cmd) werden.
Jeder Parameter besteht aus einem Pärchen mit 32 Bit Schlüssel (Tag) und einem 16 Bit Wert.
Die Tags können per Tag List abgefragt werden. Die Tag List liefert für jeden Tag eine Bezeichnung, den Minimal / Maximal Wert und die passende Einheit.

Insbesondere ist zu beachten, dass der Wert die passende Einheit hat.

Hinweis:
Aktuell sind noch nicht alle Parameter Werte implementiert, dies ist bei Verwendung der Tag Liste unerheblich, weil diese dort als ‚N.A.' gekennzeichnet sind. Spricht man diese trotzdem an, so antwortet das MX10 mit dem Tag ‚0xFFFFFFFF' und dem Wert ‚0xFFFF'.

MODUL TAG LIST [0X08.0X0C]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x0C	0b00		8	NId		Cmd	Dev	Tag			
0x08	0x0C	0b11		8	NId		Unit	Mul	Tag			

MODUL OBJECT/PROPERTY CONFIG [0X08.0X0A]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x0A	0b00		6	NId		Tag					
0x08	0x0A	0b01		8	NId		Tag				Val	
0x08	0x0A	0b11		8	NId		Tag				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' (z.B. ID von StEin, Objekt,…) das jeweils angesprochen Gerät anzugeben. Die Parameter 'Num' , 'Idx' und 'Val' sind vom jeweiligen Gerät abhängig und sind der Beschreibung des Gerätes zu entnehmen.

MODUL TAG LIST [0X08.0X0C]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x08	0x0C	0b00		8	NId		Cmd	Dev	Tag			
0x08	0x0C	0b11		8	NId		Unit	Mul	Tag			

PC ABFRAGE/ANTWORT

Grp	Cmd	M	ID	DLC	1/2	3	4	5/8	9/10	11/12	13/14	15/16	17/18	19/22	23…
0x08	0x0C	0b00		8	NId	Cmd	Dev	Tag	0x9A	Content	Min.	Max.	Val	Unit	Text
0x08	0x0C	0b11		14…	NId	Cmd	Dev	Tag	0x9A	Content	Min.	Max	Val	Unit	Text

Durch dieses Datagramm kann ein Gerät die verfügbaren Objects und Properties eines Gerätes abfragen.
Somit können ‚Fremd' Geräte ohne konkretes Wissen feststellen, was in einem ‚anderen' Gerät überhaupt einstellbar ist.

Kurzbeschreibung der notwendigen Vorgangsweise:
Gerät ‚A' möchte die möglichen Einstellwerte von Gerät ‚B' wissen:
1.
Somit sendet Gerät ‚A' einen Request an Gerät ‚B' (Per NId angegeben) mit Aggregat ‚0x0000' und Item ‚0x0000', Next = ‚1'.
Gerät ‚B' Antwortet daraufhin mit dem seinem ersten Aggregat und der ersten Eigenschaft dieses Aggregates.
Ebenso welche Einheit diese Eigenschaft hat.
2.
Gerät ‚A' widerholt nun seine Abfragen mit dem zuletzt empfangenen Aggregat/Item, und erhält so die nächste Geräte Eigenschaft. Dies ist solange zu wiederholen, bis vom Gerät ‚B' die Antwort: Aggregat ‚0xFFFF' und Item ‚0xFFFF' kommt.

Command (Cmd):
Wenn ‚0' so wird die Definition vom Angegebenen Tag zurückgeliefert. Ist der Tag unbekannt, so wird in den Datenbytes 3/4 0xFFFF zurückgegeben.
Wenn ‚1' so wird die Information vom nächsten Tag geliefert. Sollte es keine weiteren Tags mehr geben, so sind die Datenbyte 3 … 8 0xFF.

Device (Dev):
Wird für MX8 (=8) bzw. MX9 (=9) verwendet, da für diese Geräte das MX10 diese Funktionalität simuliert.

HINWEIS PC:

Der PC bekommt eine deutlich Umfangreichere Antwort, sofern ‚lange' Datagramme ‚Eingeschalten' sind.
Zu den 8 ‚Basis' Bytes kommen noch:

- Byte 9/10: Platzhalter/Reserve
- Byte 11/12: Content
  - Bit ‚0': ni (Nicht implementiert)
    Bit ‚1': RO, Wert kann nur gelesen werden
- Byte 13/14: Kleinster einstellbarer Wert
- Byte 15/16: Größter einstellbarer Wert
- Byte 17/18: Aktueller Wert
- Byte 19..22: Optional! Text für Einheit (4 Byte, fix)
- Byte 23ff: Optional! Bezeichnung des Wertes, wie am MX32 angezeigt (Bis 32 Bytes variabel).

Die Einheit und Bezeichnung sind optional, das heißt sie werden nur gesendet, wenn im System entsprechende Texte vorhanden sind. Wenn ein Text vorhanden ist, so wird auch immer eine Einheit gesendet, diese könnte aber schlicht aus 4 x 0x00 bestehen. Texte beginnen daher IMMER ab Byte 23.
Je nachdem ob ein Text vorhanden ist oder nicht, bekommt der PC daher entweder Datagramme mit 18 Bytes, oder maximal 48 Bytes.

NETWORK GROUP [0X0A]

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

PING [0X0A.0X00]

Grp	Cmd	M	NID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x00	0b00		2	NId/GrpId							
0x0A	0x00	0b10		8	Master-UID				Type		Session	

Durch einen Ping Request können Geräte andere Geräte am CAN Bus auffordern mit einem Ping Ack zu antworten.
Dadurch kann jedes Gerät prüfen ob das / die andere(n) Gerät(e) vorhanden ist.
Beim Request kann die NId eines bestimmten Gerätes angegeben werden. In dem Falle antwortet das Gerät unmittelbar, der Sender kann daher nach ca. 100mS davon ausgehen, das das Gerät antwortet oder eben zur Zeit nicht verfügbar ist.
Wenn statt einer konkreten NId jedoch eine Gruppen NId (z.B.: 0xD000 für neue Module) abgefragt wird, so antworten alle Geräte dieser Gruppe mit einer zufälligen Verzögerung von bis zu 1000mS.

Alle Geräte sollten regelmäßig einen Ping senden!
Dabei sollten folgende Kriterien eingehalten werden:

- Nach dem Start des Gerätes zumindest ein Ping innerhalb der ersten Sekunde
- Die Zeitintervalle sollten durch einen Zufallswert einen bewussten Jitter aufweisen.
- Bei geringer CAN Bus Auslastung sollte jedes Modul zwischen 500 und 1000mS zumindest einen Ping senden.
- Ausnahmen (langsameres Ping senden):
    - Bei hoher CAN Bus Auslastung (> 500 Nachrichten/Sekunde) sollten Pings ausgelassen werden, In jedem Falle muss einer alle 5000mS gesendet werden.
    - Wenn das Gerät selber viele Nachrichten zu senden hat, kann der Ping ebenfalls ausgelassen werden.
    - In jedem Falle jedoch MUSS jedes Gerät zumindest EINE Nachricht alle 5000mS am CAN Bus GÜLTIG senden!!

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, dass 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, dass es mit einer ‚unbekannten' Zentrale verbunden ist, so muss es einen Anmeldevorgang initiieren.

TEST [0X0A.0X02]

Grp	Cmd	M	NID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x02	0b01		8	ZielNId		Counter				TimeStamp	
0x0A	0x02	0b11		8	ZielNId		Counter				TimeStamp	

Kommunikation/Latenzzeit Test.
Ein Gerät kann dieses Datagramm an das Gerät mit der ZielNId senden.
Dieses Antwortet per ACK.
Dadurch kann die Durchlaufzeit von Datagrammen geprüft werden.

RAILWAY CONTROL SYSTEM [0X0B]

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/FU) und einer PC Stellwerks Software.
Diese Befehle sind erst nach einer primären Identifikation der Stellwerkssoftware verwendbar.
Diese Identifikation läuft über die Interface Options (Grp=0x0A, Cmd=0x0A, Type=0x0000, 0x0001) ab.

Ebenso ist es sinnvoll (aber nicht zwingend erforderlich), wenn ein PC Software Ihren Namen dem System mitteilt (Data Group 0x07, CMD=0x21, NID=0xC199 ... 0xC1FF).

Das System kann bis zu 64 Stelltische mit maximal 8192 Tischfeldern verwalten.
Die Aufteilung der Tischfelder zu den Stelltischen erfolgt dynamisch, es kann also z.B.: ein Monster Tisch mit 8192 Felder definiert werden, oder eben bis zu 64 Tische mit entsprechend weniger Felder.

Hinweis:
Im Folgenden wird ein Stelltisch meist als ‚Tab' bezeichnet.
Viele Stellwerks- bzw. PC Steuerprogramme verwenden Tab's um Stelltische übersichtlich darzustellen bzw. um Module je nach Bedarf zusammenzustellen und/oder zu organisieren. In einigen Fällen wird dies auch als ‚Z' Ebene bezeichnet.

RCS STATUS [0X0B.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x00	0b01		2	Tab	State						
0x0B	0x00	0b11		2	Tab	State						

Durch diese Datagramme kann eine Stellwerkssoftware Ihren Status ans System weitergeben.
Aktuell sind nur zwei States definiert:
‚0' ➔ Stellwerk AUS, Offline
‚1' ➔ Stellwerk aktiv.

RCS OPTIONS [0X0B.0X01]

Grp	Cmd	M	ID	DLC	DB1/DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x01	0b00		2	Option						
0x0B	0x01	0b01		3 .. 8	Option						
0x0B	0x01	0b1x		3 .. 8							

Durch dieses Datagramm kann eine Stellwerks-Software diverse Optionen setzen.

Option	**Verwendung**
0x01nn	RGB Wert für Ausleuchtung, nn=Stellung 0x0100: Ausleuchtung ‚Gerade' 0x0101: Ausleuchtung ‚Abzweig' DB3= Helligkeit für Rot, DB4= Helligkeit für Grün, DB5= Helligkeit für Blau

RCS POSITION [0X0B.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x02	0b00		2	FahrzeugNID							
0x0B	0x02	0b01		6	FahrzeugNID		RCS PosNID		Pin	Speed		
0x0B	0x02	0b11		6	FahrzeugNID		RCS PosNID		Pin	Speed		

Mit diesen Datagrammen kann die Fahrzeugposition abgefragt bzw. gesetzt werden.
Üblicherweise wird ein Fahrpult (MX32), welches ja keine Ahnung vom Stellwerk hat, die Position von der Stellwerkssoftware 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.

Hinweis:
Beschreibung RCS PosNID ergänzen

Die Geschwindigkeit ist in Schritten zu je 5km/h anzugeben.

RCS LOCK [0X0B.0X03]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x03	0b00		4	ObjectNId		Type	Pin				
0x0B	0x03	0b01		6 - 8	ObjectNId		Type	Pin	LockNId			
0x0B	0x03	0b11		6 - 8	ObjectNId		Type	Pin	LockNId			

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	Verwendung
0x01	Sperre für gesamtes Objekt (Zubehör Decoder, MX8, MX9, …), Pin wird ignoriert
0x02	Sperre für gesamtes Objekt aufheben Pin wird ignoriert
0x11	Sperre für Objekt ‚Ausgang' setzen Hinweis: Funktion vom jeweiligen Modul abhängig, für Zubehör Decoder, MX8/MX9 NICHT möglich
0x12	Sperre für Objekt ‚Ausgang' aufheben Hinweis: Funktion vom jeweiligen Modul abhängig, für Zubehör Decoder, MX8/MX9 NICHT möglich

LockNId dient als ‚Sperr' Referenz, darf NICHT ‚0' sein!!!
Sperren können nur aufgehoben werden, wenn ‚LockNId' exakt gleich mit dem Wert beim setzten ist.

RCS SPEED LIMIT [0X0B.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x04	0b00		3	FahrzeugNID		Tab					
0x0B	0x04	0b11		4	FahrzeugNID		Tab	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 Stellwerkssoftware 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 5km/h anzugeben.

RCS LOOK AHEAD [0X0B.0X05]

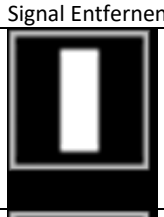
Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x05	0b00		2 [4]	FahrzeugNID		Tab	StTyp				
0x0B	0x05	0b10		8	FahrzeugNID		Tab	StTyp	Signal/Stellung		Distanz	
0x0B	0x05	0b11		8	FahrzeugNID		Tab	StTyp	Signal/Stellung		Distanz	

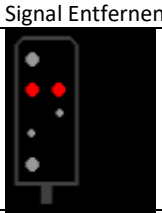
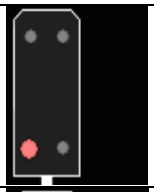
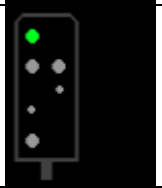
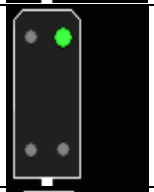
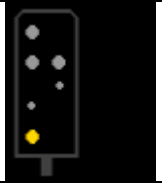
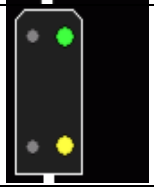
Über die Abfrage kann ein Fahrpult das Stellwerk Fragen, auf welches Signal das Fahrzeug zufährt.
Die Stellwerks Software kann diese Information entweder ‚ungefragt' als Event melden, oder per ACK eine entsprechende Abfrage beantworten.

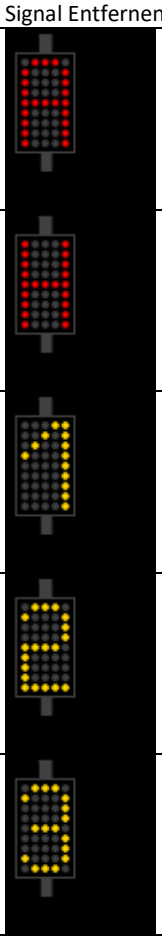
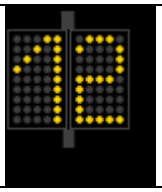
Sofern eine Stellwerks Software diese Funktion implementiert, besteht die typische Anwendung darin, dass die PC-Software mithilfe dieses Kommandos dem Lokführer (=System/MX32) das nächste Signal anzeigt.

Im Feld ‚Tab' befindet sich die Stellwerks Kennung (Max. 7 Bit bzw. 0 … 128).
Im Feld ‚StType' befindet sich der Stellwerks Type (Max. 0 … 11).
Im Feld , Signal/Stellung' wird die Signal Art und Stellung an das System übermittelt.
Von den 16 verfügbaren Bits, werden die höchsten 10 Bit für die Signal Art verwendet und verbleibende 6 Bit dienen der Stellungsinformation.
Die Entfernung sind einfach die aktuellen Meter zum Signal (14 Bit, 0 … 16383).

Derzeit definierte Signale:

Signal	**Bild**	**Stellung**	**Signal Bild (StType=16)**	**Signal ÖBB (StType=15)**	
Keines	0	0	Signal Entfernen		
	1				
	1				
	1				

Signal	Bild	Stellung	Signal Bild (StType=16)	Signal ÖBB (StType=15)
Keines	0	0	Signal Entfernen	
Hp0	2	1		
Hp1	2	2		
Hp2	2	3		
	90	1		
	90	2		
	95	1		
	95	2		
	95	3		

Signal	Bild	Stellung	Signal Bild (StType=16)	Signal ÖBB (StType=15)
Keines	0	0	Signal Entfernen	
	3	1		
	3	2		
	3	3		
	3	4		
	3	5		
	3	...		
	3			

RCS SHUNTING [0X0B.0X06]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x08	0b00		2 [4]	FahrzeugNID		[Stellwerk]					
0x0B	0x08	0b10		4	FahrzeugNID		Stellwerk					
0x0B	0x08	0b11		4	FahrzeugNID		Stellwerk					

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

RCS BLOCK [0X0B.0X08]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x08	0b00		4/6	Ziel Block NId		Aktion/Cmd		[Sende Block NId]			
0x0B	0x08	0b01		4/6	Ziel Block NId		Aktion/Cmd		[Sende Block NId]		Zug	
0x0B	0x08	0b10		8	Ziel Block NId		Aktion/Cmd		Sende Block NId		Step/Error	
0x0B	0x08	0b11		8	Ziel Block NId		Aktion/Cmd		Sende Block NId		Step/Error	

Diese Datagramme dienen der Stellwerk-Stellwerk Kommunikation.
Primär gedacht ist die Erlaubnis Abgabe/Annahme zwischen Stellwerken.

DB1/2 enthält hierzu den ‚Zielblock', darunter ist jener zu verstehen, welcher auf dieses Datagramm zu reagieren hat.
DB5/6 enthält dazu den ‚Absendeblock', also jenen, welcher die jeweilige Aktion veranlässt.
DB3/4 legt die jeweilige Funktion fest.

Anfrage (Request) Command ‚1' ➔ Zustandsabfrage
Bei Inbetriebnahme der Anlage nötig, um den aktuellen Ist Zustand der Anlage zu ermitteln.
Je nach Implementierung auch im laufenden Betrieb, zwecks Überwachung ob Blockstelle noch betriebsbereit, vorhanden, etc. Alternativ könnte eine Blockstelle dies auch per ‚Ping' fortlaufend selbständig melden.

Anfrage (Request) Command ‚2' ➔ Vor-Block
Der ‚Zielblock' antwortet mit seinem jeweiligen Vor-Block, ist an sich nicht zwingend, da dieser ja im Stellwerk eigentlich bekannt sein sollte.
Anfrage (Request) Command ‚3' ➔ Rückblock
Der ‚Zielblock' antwortet mit seinem jeweiligen Rück-Block, ist an sich nicht zwingend, da dieser ja im Stellwerk eigentlich bekannt sein sollte.

Command ‚0x12': Erlaubnisabgabe vom ‚Vor' Block
Command ‚0x13': Erlaubnisabgabe vom ‚Rück' Block

Command ‚0x22': Erlaubniserhalt vom ‚Vor' Block
Command ‚0x23': Erlaubniserhalt vom ‚Rück' Block

RCS ROUTING [0X0B.0X0C]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x0B	0b00		6	Action		Src/Route		[Destination]			
0x0B	0x0B	0b10		8	Source		Destination		RouteNId		Step/Error	
0x0B	0x0B	0b11		8	Source		Destination		RouteNId		Step/Error	

Dieses Datagramm dient dem Abrufen und bestätigen von Fahrstraßen (Route).
Ein Gerät kann mit dem ‚Route' Command einer Stellwerkssoftware die ‚Start-/Ziel- Codes' senden.

Ablauf:
Das System (MX10/MX32) sendet das Route Command an die jeweilige Stellwerkssoftware.
Dabei hat ‚Action' folgende Bedeutungen:
Action=0x0000: Rangierstraßen: Src/Route ist die Nummer, Destination ist in dem Falle 0.
Action=0x0001: Rangierstraßen: Src/Route ist der ‚Start Code Taste', Destination ist der ‚Ziel Code/Taste'.
Action=0x0002: Fahrstraßen: Src/Route ist die Fahrstraßen Nummer, Destination ist in dem Falle 0.
Action=0x0003: Fahrstraßen: Src/Route ist der ‚Start Code Taste', Destination ist der ‚Ziel Code/Taste'.
Action=0x8FF0: Auflösen der Rangier-/Fahrstraße mit der Nummer lt. Src/Route
Action=0x8FF1: Auflösen der Rangier-/Fahrstraße mit den Source/Destination Codes

Die Stellwerkssoftware beantwortet das Command durch ein ACK mit RouteNId und Step/Error.
Dabei bedeutet RouteNId=0 das die Stellwerkssoftware die Fahrstraße nicht kennt (Also schichtweg ‚keine Ahnung hat').
Wenn die gewünschte Fahrstraße prinzipiell möglich ist, so muss die Stellwerkssoftware dieser Fahrstraße eine NId im Bereich 0x6100 … 0x61FF vergeben. Die konkrete NId ist dabei egal, allerdings DARF sich diese während EINER Session (Also von System Power On bis System Power Off) nicht ändern.

Im weiteren Ablauf kann die Stellwerkssoftware den Aufbau der Fahrstraße als Event ans System melden.
Hier gilt prinzipiell das ein Step/Error = 0xFxxx einen Fehler kennzeichnet.

RCS TAB [0X10]

Damit kann ein Stelltisch (Tab) abgefragt bzw. angelegt werden.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x10	0b00		1	Tab							
0x0B	0x10	0b10		6	Tab	Type	X-Size		Y-Size			
0x0B	0x10	0b11		6	Tab	Type	X-Size		Y-Size			

Mit dem Request kann abgefragt werden ob das System einen Tisch kennt.
Mit dem Command kann ein Stelltisch modifiziert bzw. angelegt werden.
Jeder Stelltisch sollte ebenfalls einen Namen (Grp=0x07, Cmd=0x21) bekommen.
ERST nachdem ein Stelltisch (Tab) angelegt wurde, können die Tischfelder festgelegt werden.

Type	Beschreibung
0	Ungültig
1	Simpel Panel für MX32
2 …	Stellwerk Art. Derzeit sind nur die Symbole lt. Anhang vorhanden, daher werden alle ‚Stellwerke' gleich dargestellt. In Zukunft sind aber mehrere Typen machbar.

RCS FIELD ICON [0X11]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x11	0b00		4	Stellwerk, X-, Y- Position							
0x0B	0x11	0b10		6	Stellwerk, X-, Y- Position				Icon			
0x0B	0x11	0b11		6	Stellwerk, X-, Y- Position				Icon			

Dieses Datagramm dient zum abfragen bzw. festlegen der Stellwerksfeld Icons.

Bit Definition für ‚Stellwerk, X-, Y- Position':

Bit	Beschreibung	Gültige Werte
00 … 09	10 Bit für Y Koordinate des Feldes	0 … 1023
10 … 19	10 Bit für X Koordinate des Feldes	0 … 1023
20 … 23	Derzeit frei	
24 … 29	Nummer für Stellwerk, Z-Ebene, Tab, …	0 … 63
30	Derzeit frei	
31	Feld Löschen	0='Setzen', 1=Löschen

Hinweis RocRail:
RocRail verwendet die Stellwerknummer als ‚Z' Ebene bzw. als Tab-Nummer.

RCS FIELD ACTUATOR [0X12]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x12	0b00		4	Stellwerk, X-, Y- Position							
0x0B	0x12	0b10		8	Stellwerk, X-, Y- Position				NId		Pin	State
0x0B	0x12	0b11		8	Stellwerk, X-, Y- Position				NId		Pin	State

Damit kann festgelegt werden, welcher/welche Aktuatoren einem Tischfeld zugeordnet sind.
Mit Aktuatoren sind Aktive Elemente wie Weichen, Signale, etc. gemeint. Jedes Feld darf bis zu 16 dieser Aktuatoren haben.

RCS FIELD FEEDBACK [0X13]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0B	0x13	0b00		4	Stellwerk, X-, Y- Position							
0x0B	0x13	0b10		7	Stellwerk, X-, Y- Position				NId		Pin	
0x0B	0x13	0b11		7	Stellwerk, X-, Y- Position				NId		Pin	

Z[IMO]PROGRAMMABLE[SCRIPT] [0X0C]

Die Datagramme der ZPS Gruppe dienen der Kommunikation mit der Z[imo]Programmierbaren[Scripts].

ZPS STATE [0X0C.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x00	0b00		2	TaskNId							
0x0C	0x00	0b10		6	TaskNId		State		Step			
0x0C	0x00	0b11		6	TaskNId		State		Step			

Per Request Datagramm kann der aktuelle Status eines ZPS abgefragt werden.

Per Command kann das jeweilige Script gestartet bzw. angehalten werden.

ZPS MODIFY [0X0C.0X02]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x02	0b00		8	ScriptNId		ScriptNum		KeyS		KeyX	
0x0C	0x02	0b01		8	ScriptNId		ScriptNum		KeyS		KeyX	
0x0C	0x02	0b1x		8	ScriptNId		ScriptNum		KeyS		KeyX	

Mit diesen Datagrammen kann die Basis Info für ein Script abgefragt werden bzw. ein neues angelegt werden.

Egal ob Abfrage oder Command, die Antwort enthält immer die Script NId, die Script Nummer und die Start/Ziel Tasten
Abfrage:
Version 1: Per Script NId
Version 2: Per Script Nummer
Version 3: Per Start/Ziel Tasten (Signale, Nummern)

Command:
Version 1: Per Script NId,
in dem Falle sind Num, KeyS und KeyX mit ‚0' zu senden
Version 2: Per Script Nummer,
in dem Falle ist NId=0xFFFF, KeyS und KeyX mit ‚0' zu senden.
Version 3: Per Start/Ziel Tasten (Signale, Nummern)
In dem Falle ist NId=0xFFFF, Num=0 zu senden.

Hinweis:
Script Nummern sind nur von 1 … 255 erlaubt. Die Nummer ‚0' ist ungültig.

SCRIPT DELETE [0X0C.0X03]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x03	0b10		2	ScriptNId							
0x0C	0x03	0b11		8	ScriptNId		ScriptNum		KeyS		KeyX	

Mit diesem Datagramm wird ein komplettes Script gelöscht.

SCRIPT OPTIONS [0X0C.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x04	0b01		4	ScriptNId		Option					
0x0C	0x04	0b10		6	ScriptNId		Option		Value			
0x0C	0x04	0b11		6	ScriptNId		Option		Value			

Mit Hilfe dieser Datagramme können diverse Einstellungen für Abläufe abgefragt bzw. Festgelegt werden.
Option = 0x0001: Zeitbasis Multiplikator.

SCRIPT ITEM: COMMAND SWITCH [0X0C.0X08]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x08	0b00		4	ScriptNId		Step					
0x0C	0x08	0b10		8	ScriptNId		Step	Wait	ObjNId		Pin	Pos
0x0C	0x08	0b11		8	ScriptNId		Step	Wait	ObjNId		Pin	Pos

Dieses Datagramm dient zum simplen modifizieren von Schaltfolgen/Weichenstraßen.

‚ScriptNId' ist der jeweilige Task, dieser MUSS zuvor angelegt werden!! Sollte er nicht vorhanden sein, antwortet das Zs100 mit Step = 0xFFFF, ObjNId=0xFFFF, Pin = 0xFF, Pos=0xFF.
‚Step' gibt an, welcher Schritt der Schaltfolge zu ändern ist.
‚Wait' gibt an wie viele Ticks zum ‚vorigen' Schritt abzuwarten sind.
‚ObjNId' ist jene NetworkId, welche den Befehl auszuführen hat (z.B.: DCC Dekoder, MX8, StEin, …).
‚Pin' ist der jeweilige Anschlusspin, welcher in den Zustand ‚Pos' zu schalten ist.

TASK COMMAND ‚A' [0X0C.0X0A]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x08	0b00		4	ScriptNId		Step					
0x0C	0x08	0b10		8	ScriptNId		Step		Cmd		ObjNId	
0x0C	0x08	0b11		8	ScriptNId		Step		Cmd		ObjNId	

Diese Datagramme dienen dem Zugriff auf den Command Teil ‚A'.
Bei Abfrage (Req=0b00) erhält man das Command Objekt und den Befehl, den das Objekt ausführen soll.
Dies kann per Command Datagramm (Cmd=0b10) geändert werden.
Wenn im Command als Step Nummer 0xFFFF angegeben wird, so wird das Task-Command an den genannten Task angefügt.

TASK COMMAND ‚B' [0X0C.0X0B]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0C	0x08	0b00		4	ScriptNId		Step					
0x0C	0x08	0b10		8	ScriptNId		Step		Item		Value	
0x0C	0x08	0b11		8	ScriptNId		Step		Item		Value	

PORT OPEN [0X0A.0X06]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x06	0b01		0								

Mit diesem Datagramm kann ein Gerät die Ethernet Schnittstelle des MX10 ‚öffnen'.
Als Antwort bekommt das Gerät ein ‚Ping' vom MX10.

LOGOFF / PORT CLOSE [0X0A.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 Kommunikationssport (USB oder Ethernet) geschlossen. Das jeweilige Gerät muss zur Wiederaufnahme der Verbindung wieder die jeweiligen Initialschritte abarbeiten.
Als (Ziel-) NID ist dabei die NID jenes Gerätes anzugeben, von welchem sich der Schnittstellen Benutzer (PC Software) abmelden will.

INTERFACE OPTION [0X0A.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 und 0x0001 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 Fahrstraßeninformationen zur Verfügung)
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	MX9 ‚0' Zugnummern senden
	17	Full Power Message siehe [0x00.0x00]

INFO VALUES FÜR TYPE 0X0001

Type	Value	Verwendung
0x0001	0x0000	ZIMO Intern
	0x0010	Kennung für ESTWGJ
	0x0018	RocRail
	0x0020	Kennung für STP
	0x0021	Kennung für Pfusch
	0x0030	Kennung für TrainController
	0x0031	Kennung für TrainProgrammer
	0x0040	RailManager
	0x0050	Win-Digipet
	0x0070	iTrain

INTERFACE ERROR [0X0A.0X0F]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0A	0x0F	0b11		8	NID		Grp	Cmd	Value			

Dieses Datagramm wird vom MX10 gesendet, wenn ein Befehl fehlerhafte Parameter enthält oder aus anderen Gründen nicht ausführbar ist.

RF CONNECT MAKE/KILL [0X0A.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		Chn.
0x1x	Verbindungsaufbau: x' gibt das Land an: ‚1' : Amerika ‚2': Europa ‚F' : Alle restlichen Länder	Funk Kanalnummer
0x80	Funk Fehler Meldungen	Fehler Meldungen 0x00 ➔ Kein Fehler, Error Clear 0x01 ➔ Funkmodul Fehler
0xFx	Verbindung abbauen	

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					

Statusabfrage und -meldungen des Funkmodules.
Während des Auf-/Abbaus meldet das Funkmodul periodisch den jeweiligen Schritt/Zustand des Auf- und Abbaus 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
0x0010	Buffer State	0x0000 = Buffer verfügbar 0xFFFF = Voll
0x0011	Buffer Clear	0 = Alle Buffer löschen NId = Alle Buffer für NId löschen

FILE CONTROL [0X0E]

Mit Hilfe der 'File Control [0x0E]' Group wird der File Transfer gesteuert.

FILE TYPES

File Type	Beschreibung
0x010x	GUI Sprache 0 … F
0x011x	Bezeichnungen in Sprache 0 … F
0x1501	MX10 Firmware
0x1502	MX10 XILINX
0x1503	MX10 Funkprozessor
0x3001	MX32, Rev. 5 Firmware
0x3301	MX32, Rev. 7 Firmware
0x3303	MX32, Rev. 7 Funkprozessor
0x9001	StEin Firmware
0x9002	StEin XILINX
0x9004	StEin Config
0x9005	StEin Sound
0x9201	Roco Melder 10808 Firmware
0x9204	Roco Melder 10808 Config
0x9206	Roco Booster 10806/10807

Hinweis:
Diese Liste wird nach Bedarf ergänzt!!!

FILE CONTROL, STATE [0X0E.0X00]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x00	0b00		4								
0x0E	0x00	0b1x		4								

Abfrage für Datentransfer Zustand
Derzeit noch nicht implementiert

FILE CONTROL, LIST [0X0E.0X01]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x01	0b01		4	FileType		Index					
0x0E	0x01	0b11		4	FileType		Index					

File Liste für File Type Abfragen.

FILE CONTROL, OPEN [0X0E.0X04]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x04	0b01		6	FileId		FileType		DestNId			
0x0E	0x04	0b11		8	FileId		FileType		SrcNId		State	

Mit dem File Open Command wird ein File Transfer Stream geöffnet.
Der File Absender sendet ein Open Command [0b01] an das Gerät mit der ‚DestNId', dem File Type und einer Stream/File Id.
Das Betroffene Modul antwortet mit ‚ACK'. Die ‚SrcNId' ist die NId des Absenders (Aus dem CAN des Commands Identifier).
Im ACK müssen FileType und FileId mit dem Command übereinstimmen, mit ‚State' gibt der Empänger des Streams seinen Status an.

State	Beschreibung
0x0000	Ungültig, Modul unterstützt die Funktion nicht
0x0001 …. 0xFFFE	Statusmeldung, nach Bedarf zu ergänzen
0xFFFF	Open abgelehnt / Cancel

FILE CONTROL, INFO [0X0E.0X05]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x05	0b00		4	FileId		InfoType					
0x0E	0x05	0b01		8	FileId		InfoType		InfoData			
0x0E	0x05	0b11		8	FileId		InfoType		InfoData			

Das File Control Info Req, Cmd, Ack dient dem Austausch von File Informationen.

FILE CONTROL INFO TYPES

Info Type	Beschreibung	
0x0001	Gesamt Länge der Daten	Mandatory
0x0002	File CRC32	Mandatory
0x0003	Block Size (Für CAN typisch 2048Bytes)	Mandatory
0x0011	File Version	Optional
0x0011	File Date	Optional
0x0012	File Time	Optional
0xFFFF	Negativ Response	Optional

Sollte der Empfänger Infos ablehnen, so kann er dies entweder durch Angabe seiner Grenzen machen, oder durch Antworten mit einer negativen Response.
Beispiel:
Sender möchte 5Mbyte übertragen, der Empfänger hat aber nur für 4Mbyte Platz ➔ ACK mit Data = 4Mbyte
Sender will File Version 1.2.3, welche der Empfänger nicht hat/kennt ➔ ACK mit negativ Response.
Hinweis:
Diese Liste wird nach Bedarf ergänzt!!!

FILE CONTROL, CLOSE [0X0E.0X07]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x07	0b00		6	FileId		FileType		DestNId			
0x0E	0x07	0b1x		8	FileId		FileType		SrcNId		State	

Durch diesen Befehl wird ein File Transfer beendet.
Wenn der Empfänger zur Verarbeitung der Daten mehr als 100mS benötigt, so muss er dies per Event etwa alle 500mS wiederholen.
Dabei gilt ‚State = 0x0000' endgültig abgeschlossen, State 0x0001 … 0xFFFE Close läuft, 0xFFFF Cancel.
Wenn der Empfänger das Close ‚verzögern' will (Weil er noch Verarbeitungszeit benötigt), so sollte er im State die Restlaufzeit in mS angeben.
Erst EIN ACK beendet den Transfer endgültig.

FILE CONTROL, DATA INIT [0X0E.0X11]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x11	0b01		8	FileId		Block Size		File Pos			
0x0E	0x11	0b10		8	FileId		Block Size		File Pos			

Mit diesem Command legt der Absender fest ab welchem Byte der folgende Stream startet.
Ebenso wie viele Bytes in dem folgenden Stream übertragen werden.
BEVOR der Absender den Stream senden darf, muss er das ACK des Empfängers abwarten.
Dieser muss klarerweise mit der gleichen FileId und File Pos antworten.

Sobald dieses Handshake abgeschlossen ist, überträgt der Sender die Daten mit der Gruppe 0x0F (siehe dort).

Am CAN Bus beträgt die maximale Block Size 2048Byte.
Sollte die vom Absender angegebene Block Size grösser sein, als jene, welche der Empfänger verarbeiten kann, so muss der Empfänger mit seiner Maximal Block Size antworten.
In diesem Falle hat der Absender zu entscheiden, ob er die kleinere Block Size verwenden will (Sollte im Normalfall kein Problem sein) und das Data Init erneut mit der angepassten Block Size zu senden.
Oder, wenn dies nicht möglich ist, muss er den Transfer beenden.

Ähnliches gilt für die File Pos, wenn diese außerhalb des gültigen Bereiches liegt, so muss der Empfänger mit File Pos = 0xFFFFFFFF antworten. Der Sender kann erst nach erneuten senden mit korrekter File Pos weiterarbeiten.

ERST wenn der Data Init Handshake von Sender und Empfänger gültig abgeschlossen ist, dürfen tatsächlich Daten gesendet werden.

FILE CONTROL, DATA ACK [0X0E.0X12]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x12	0b00		8	FileId		Reserved		CRC32			
0x0E	0x12	0b10		2	FileId							
0x0E	0x12	0b11		8	FileId		Reserved		CRC32			

Sobald der Empfänger die Daten Empfangen/Verarbeitet hat meldet er dies durch ein ACK an den Absender.
Eine Abfrage ist daher im Normalfall nicht notwendig!!!

HINWEIS:
Der Empfänger kann/darf das ACK bis zu einer Sekunde verzögert senden, wenn er den Block länger offenhalten will, so muss er zumindest 1x/Sekunde ein Evt senden.

FILE CONTROL, DATA NACK [0X0E.0X13]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0E	0x13	0b10		4	FileId		Error Code					

Sollte beim Datentransfer ein Fehlerzustand auftreten, so kann der Empfänger dies JEDERZEIT per Data NACK Event melden.
Error Code = 0x0000 ➔ Fehlerzustand beendet
Error Code = 0x0001 ➔ Datagramm out of Sequenz
Error Code = 0x0002 ➔ Pause (Empfänger wünscht Pause),
Muss durch ein Event mit Code = 0x0000 beendet werden.

Hinweis:

Diese Liste wird nach Bedarf ergänzt!!!

FILE TRANSFER [0X0F]

FILE TRANSFER [0X0F.NN]

Grp	Counter	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x0F	Counter	FileId	0 .. 8								

Nach dem Data Init wird der eigentliche Daten Stream mit diesem Datagramm übertragen.
Der Absender überträgt im Counter die Datagramm Nummer INNERHALB des jeweiligen Blockes.
Startet also nach dem Block Init mit Counter=0 und zählt bei jedem Datagramm hoch.
Somit kann ein Block maximal 256x8=2048 Bytes übertragen.

Sollte der Empfänger einen Fehler erkennen (z.B.: Counter out of Sequenz), die Daten nicht mit der Sendegeschwindigkeit verarbeiten können, oder aus sonstigen Gründen den Transfer unterbrechen wollen, so kann er dies jederzeit durch das Data NACK Command machen.

ZUSATZ BEFEHLE FÜR PC ANBINDUNG ÜBER LAN:

Um die PC Kommunikation zu optimieren, stehen einige Datagramme in einer lang-Form zur Verfügung.
Damit kann ein PC mehr Informationen in einem Datagramm abfragen und senden.

Die Ausgangsbasis dieser ‚Lang' Datagramme sind die jeweiligen CAN Datagramme in den Gruppe 0x00 bis 0x0F.

Beispiel Modul Power Info:
Das CAN Datagramme hat die Gruppe 0x08, Command 0x00.
Die PC (lang) Form hat die Gruppe 0x18, Command 0x00.
Auch Inhalts mäßig sind diese Datagramme extrem ähnlich. Da am CAN nur 8 Datenbytes zur Verfügung stehen, werden der Zustand und die aktuellen Strom/Spannungswerte abwechselnd je Schiene übertragen. Da UDP jedoch 1536 Nutzdatenbytes haben (genutzt werden jedoch maximal 1024) können beide Schienen Zustände und die beide Strom/Spannungswerte in einem Datagramm gesendet werden (lang Form).

Ähnliches gilt auch für das Übertragen von Namen, Zugnummern, usw. letztendlich transportieren die lang Datagramme den gleichen Inhalt wie die CAN Datagramme, fassen diese jedoch sinnvoll zusammen und reduzieren damit den Trafik.

UDP ONLY: DATA GROUP [0X11]

PC ONLY: ACCESSORY DATA EXTENDED [0X11.0X05]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).
Damit kann der Gleiszustand von einem MX9 oder StEin abgefragt werden.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 7..10	Objekt Data DB 11…
0x11	0x05	0b01		10	SrcNId	ObjNId	SubId	Pin‘s	
0x11	0x05	0b10/0b11		(52)	SrcNId	ObjNId	SubId	Pin‘s	

Die ‚SrcNID‘ (Source/Quell Net ID) gibt an welches Gerät die Frage beantworten soll. Typischerweise ist die die NID jener Zentrale, mit welcher der PC verbunden ist. Kann aber vor allem in einem Netzwerk auch ein anderes Gerät (z.B.: Tab, anderer PC) sein.

Gleisabschnittsdaten (SubId = 0x00):

DB	Len	Inhalt
5..8	4	8x Gleis Stati gem. ‚Simpel‘ Datagramm 0x01.0x02
9..12	4	HLU Info
13..16	4	16x Eingang
17..20	4	16x Ausgang
21..52	32	Je 2 Byte für die Erkannte Zugnummern

UDP ONLY: eXtended TSE Group [0x16]

PC ONLY: LOCO SIGNUP CONTROL [0X16.0X11]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).
Steuerung der Fahrzeug Anmeldung Fahrzeug Anmeldung gesteuert werden (RCN218 + ZIMO Eränzungen)

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 7..10
0x16	0x11	0b01		6	Action	Repeat	Intervall	
0x16	0x11	0b10		6	Action	Repeat	Intervall	
0x16	0x11	0b11		6	Action	Repeat	Intervall	

Action=0x0000: Anmeldung beenden
Action=0x0001: Anmeldung starten
Wenn ‚Anmeldung starten‘, dann geben Repeat die Anzahl der ‚Logon Enable‘ Aussendungen an und Intervall das Zeitintervall der Aussendungen.
Hinweis: das Intervall hat eine Zeitbasis von 100mS, sprich Intervall ‚1‘ bedeutet alle 100mS wird ein Logon Enable gesendet (Etwa jeder 10. DCC Befehl, bzw. ca. 10x je Sekunde)

Der jeweilige ‚Repeat‘ wird per Event (0b10) gemeldet.

UDP ONLY: LOCO SIGNUP INFO [0X16.0X12]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).

Mit dem folgenden Datagrammen erfolgt die Anmeldung:

Grp	Cmd	M	ID	DLC	DB 1	DB 2	DB 3 .. 4	DB 5 .. 6	DB 7..10
0x16	0x12	0b01		10	Action	Flags	LocoNId	Vendor	DecoderUId
0x16	0x12	0b10		6	Action		LocoNId	Vendor	DecoderUId

Sobald ein Decoder auf Logon Enable reagiert, wird der Hersteller und die Decoder UId gemeldet (Action = 0x01).
Wenn der Decoder auf ‚ShortInfoGet' Antwortet wird unter LocoNId die von Ihm gewünschte Adresse (Wunschadresse) als Event gemeldet (0x02).

Ein PC sollte auf Action 0x02 warten, in den Flags meldet das MX10 was es über den Decoder ‚weiß':
0x08 ➔ Unbekannter Decoder (Hersteller+DecoderUID), Ebenso die Wunschadresse
0x09 ➔ Unbekannter Decoder (Hersteller+DecoderUID), Wunschadresse jedoch vorhanden
0x0A ➔ Hersteller+Decoder UId sind in der MX10 Datenbank mit der Wunschadresse vorhanden
0x0B ➔ Hersteller+Decoder UId sind in der MX10 Datenbank aber mit unterschiedlicher Wunschadresse

Hinweis: Die Flags könnten mehr werden, das sind mal die absolut offensichtlichen.

Per Command kann der PC dann das Logon Assign durchführen.
Action 0x8A ➔ Simple Bestätigung das eh alles passt.
Action 0x8B ➔ Änderung der Adresse für den Deocoder (inkl. Anpassung der MX10 Datenbank).

UDP ONLY: DECODER GUI CONTROL [0X16.0X14]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).

Durch das Decoder GUI Control Command kann ein Gerät (Fahrpult, App, …) die im Decoder gespeicherte GUI abrufen.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 8	DB 8 ..
0x16	0x14	0b01		8	DecoderNId	Control	Data	
						0x0100	0	
						0x0201	0	
0x16	0x14	0b10		8	DecoderNId	Control	Data	
0x16	0x14	0b11		8	DecoderNId	Control	Data	
						0x0100	0/0xFFFFeee	
						0x0201	0/0xFFFFeee	

Decoder GUI Abruf Start:

- Command (M=0b01): Angabe der Decoder NId (Fahrzeugadresse),
  Control = 0x0100 ➔ ZIMO GUI
  Control = 0x0201 ➔ RCN218 Data Block ‚a'

Antwort (ACK) vom System:

- Decoder NId, Control 0x0100/0x0201,
  Data=0x00000000 wenn Abfrage gestartet wurde,
  Data=0xFFFFeeee wenn Abfrage aktuell nicht machbar, Ursache in ‚eeee' codiert

Events vom System während Prozess läuft:

UDP ONLY: DATA GROUP [0X17]

ITEM LIST BY INDEX [0X17.0X01]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x01	0b00		8	SrcNId		GroupNId		Index		Max	Flags
0x07	0x01	0b11		8	Idx1		NId1		Data			

ITEM LIST BY NID [0X17.0X02]

Grp	Cmd	M	ID	DLC	DB 1, 2	DB 3, 4	DB 5 .....
0x17	0x02	0b00		4	SrcNID	NId	
0x17	0x02	0b11		xx	NId	Index	Object State & Info

Sofern der NId einem Fahrzeug Objekt zugeordnet ist, werden folgende Zusatzdaten mitgeliefert:

DB	Bit	Inhalt
5 ... 8	00 ... 03	Schienen Format
	04 ... 07	Fahrstufen
	08	Richtung ‚Soll' (Vorgabe von Bediengerät)
	09	Richtung ‚Track' (Aktuell an die Schiene gesendete Richtung)
	10 .. 19	Fahrstufe im 1024 Format
	20 .. 22	Ost/West
	23	RailCom Vorhanden
	24	ZIMO Ack
	25	E-Stop
	26	Object gehört zu Zug
	27 .. 30	Unused
	31	Delete Mark

Hinweis:
Dieser Befehl wird weitere Informationen bekommen!

UDP ONLY: DATA VALUE EXTENDED [0X17.0X08]

Dieser Befehl steht nur am PC Interface zur Verfügung (Z20-LAN).
Damit können die Fahrzeug Fahr Daten abgefragt werden.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 9 .. n
0x17	0x08	0b00		6	SrcID	NId	SubID	
0x17	0x08	0b11		12 ..	NId	SubID	Object Daten	

Sofern NId ein Fahrzeug darstellt und SubId = ‚0', enthalten die Daten Bytes folgende Objekt Daten:

DB	Len	Inhalt
5..6	2	Vendor Id (gem. RCN218)
7..10	4	Decoder UId (gem. RCN218)
11..12	2	ZIMO Decoder Type
13..16	4	ZIMO Sound Project
17	1	Track Format
18	1	Track Function
19..20	2	Track Sende Counter
21..22	2	Track Empfangs Counter
23..24	2	
25..26	2	OwnerNId: NId des aktiv steuernden Gerätes
27..30	4	OwnerTick: Tick des letzten Commands
31..32	2	TSE2NId: Gerät welches das Fahrzeug zuletzt auf RüF Priorität gesetzt hat
33..36	4	TSE2Tick Tick wann das Fahrzeug zuletzt auf RüF Priorität gesetzt wurde
37..38	2	Geschwindigkeit, Richtung
39..42	4	Schaltfunktionen 0 … 31. Je ein Bit pro Funktion
43..46	4	Sonderfunktionen: Bit 0..3: Rangieren Bit 4..5: Manual
47..112	64	32 Analog Funktionen, 1 Byte Nummer, 1 Byte Wert
113..		

Sofern NId ein Fahrzeug darstellt und SubId = ‚1', enthalten die Daten Bytes folgende Objekt Daten:

DB	Len	Bit	Inhalt
1..2	2		Data NId
3..4	2		Data Idx
5..8	4		Flags
		0..30	
		31	Gelöscht
9..12	4		
		0..7	Zug Nummer
13..14	2		Hersteller Nummer
15..18	4		Decoder Id
19..20	2		Decoder Typ
21..24	4		Sound UId
25	1		Track Format
26	1		Anzahl Funktionen
27..28	2		Anzahl DCC Pakete an Decoder
29..30	2		Anzahl RailCom Rückmeldungen vom Decoder
31..32	2		RailCom Rückmelde Flags
33..34	2		Fahrzeug ‚Besitzer'
35..38	4		Letzter Steuerbefehl
39..40	2		Last P2 Command Sender NId
41..44	4		Last P2 Command Tick
45..46	2		Speed
47..50	4		Function States
51..54	4		Special Functions
		0..3	Rangier Faktor
		4..5	Manual Mode
55..	64		32 Analog Function Num/Val
	2		RailCom Speed Value
	4		RailCom Speed SysTick

UDP ONLY: (FAHRZEUG) FUNKTION MODES EXTENDED [0X17.0X14]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB …
0x07	0x14	0b00		4	FahrzeugNID		
0x07	0x14	0b01		4…	FahrzeugNID		
0x07	0x14	0b11		4…	FahrzeugNID		

Durch das Datagramm M = 0b00, DLC = 3, wird der Funktionsmode für das Fahrzeug ‚FahrzeugNId' und der Gruppe ‚Grp' abgefragt. Mit dem Datagramm M = 0b01, werden die Funktionsmodi für das für die Tastenliste festgelegt.
Eine Abfrage wird per Datagramm M=0b11.

Tasten Liste

DB	Len	Inhalt
3 … 4	2	Tasten Nummer (Normalerweise die jeweilige DCC Funtion) Mapping 0 … 63: DCC Funktion 0 … 63 Mapping 256 … 511: DCC Analog Funktion 0 … 255 Mapping 32768 … 65535: DCC Binary State 0 … 32767
5 ... 6	2	Tasten Mode 0b0000000000000000: Dauer 0b0100000000000000: Moment 0b10ttttttttttttt: Zeit, Dauer ‚t' x 10mS (10mS … 163.830mS bzw. 163Sekunden/2Min:43Sek)
7 … 10	4	Angabe für Taste 2
….	4	Angabe für Taste ‚n' Maximal 64 Tasten/Datagramm (256 Byte)

Hinweis:
Mapping > 63 derzeit nicht implementiert!

UDP ONLY: LOCO SPEEDTAB EXTENDED [0X17.0X19]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).
Damit können die Fahrzeug Geschwindigkeit Tabellen Abgefragt bzw. Festgelegt werden.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5	DB 6	DB 7 .. n
0x17	0x19	0b00		6	SrcID	NID	0	0	
0x17	0x19	0b11			SrcID	NID	0	3	3 Punkt Steps/Speed

In der Abfrage kann angegeben werden welche Geschwindigeitstabelle abgefragt oder geändert werde soll.
Tab = 0: MX3n Standard Tabelle

Diese enthält so viele Speed Pärchen wie unter ‚Items' angegeben.

DB	Len	Inhalt
7..8/11..12/15..16	2	Fahrstufe auf 1024 Stufen skaliert
9..10/13..14/17..18	2	Passende kmH

UDP ONLY: LOCO GUI EXTENDED [0X17.0X28]

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).
Damit können die Fahrzeug Basis GUI Informationen ans System gesendet werden bzw. abgefragt werden.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 9 .. n
0x17	0x28	0b00		6	SrcID	NId	SubID	
0x17	0x28	0b01		12 ..	NID	SubID	Fahrzeug GUI Daten	
0x17	0x28	0b11		12 ..	NID	SubID	Fahrzeug GUI Daten	

In der Abfrage kann angegeben werden welcher GUI Teil/Version abgefragt wird.

Wenn das Datagramm als Abfrage genutzt wird, kann angegeben werden, welches Gerät Antworten soll (SrcNId).
Wenn als SrcNId = 0x000 genutzt wird, so antwortet der MX10 Master.

Mit NId wird angegeben für welches Gerät die GUI abgefragt bzw. beantwortet wird. Derzeit ist nur für Fahrzeuge der GUI Speicher vorgesehen (Es könnte aber auch für andere Geräte ein solcher vorhanden sein).

Per SubId wird angegeben welcher Speicher angefragt bzw. geändert wird.
SubId = 0x0000: System GUI, sofortige Übermittlung an alle Geräte
SubId = 0x1000: LAN/PC/APP (Externe) GUI mit sofortiger Weiterleitung an CAN Bus Geräte
SubId = 0x1001: LAN/PC/APP (Externe) GUI ohne Weiterleitung an CAN Bus Geräte
SubId = 0x8000: Decoder GUI, READ ONLY

Antwort auf Abfrage mit SubId=0x0000:

DB	Size	Inhalt
5 … 8	4	Version Nummer
9 … 10	2	Flags, Bit 16=Delete
11 … 12	2	Group
13 … 44	32	Fahrzeug Name
45 … 46	2	Image Num
47 … 50	4	Image CRC
51 … 52	2	Tacho Num
53 … 56	4	Tacho CRC
57 … 58	2	Höchste Vorbild Zulassungsgeschwindigkeit (Nicht zwingend die System Höchstgeschwindigkeit!!!)
59 … 60	2	Höchste Vorbild Rückwärts Geschwindigkeit
61 … 62	2	Typische Rangiergeschwindigkeit (Bremsweg < 500m)
63 … 64	2	Antriebsart
65 … 66	2	Epoche Codierung: Es E: Oberes Byte, Hauptepoche S: Unteres Byte, Sub Epoche Z.B.: Epoche IIIa ➔ 0x0301
67 … 68	2	Land
69 … 132	64x2	Funktionsicons (64)
133 … 196	64x2	Funktions Mode‘s

Hinweise:
1.
Die in diesem Datagramm genutzten Geschwindigkeiten sind jene gem. Typen Zulassung des Vorbildes. Diese können, müssen aber NICHT!!, mit den Tachoeinstellungen zusammenpassen. Werte von ‚0‘ oder > 500kmH werden ignoriert.
Bei ‚brauchbaren‘ Werten wird die Default Kurve neu berechnet (Kann natürlich per Speed Tabelle in Folge überschrieben werden).
2.
Derzeit kann ein Fahrzeug bis zu 64 Funktionen enthalten, für jede Funktion kann ein Icon hinterlegt werden.
Die Themen Logik bekommt (Sobald ausreichend definiert) eine eigene Sub Id. Aktuell sind im System die Icon Nummern 700 bis 999 (Somit 299 unterschiedliche) möglich. Auch dieser Punkt wird in Absehbarer Zeit geändert.

UDP ONLY: INFO / CONFIG GROUP [0X18]

MODUL POWER INFO [0X18.0X00]

Sofern eine PC Software/Client 'lange' Datagramme aktiviert hat, sendet das MX10 seine Gleis Zustands, Strom und Spannungswerte in einem 'langen' Datagramm.

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3 …..
0x18	0x00	0b00		2	Dev-NId		
0x18	0x00	0b1n		22	Dev-NId		Daten Details siehe folgende Tabelle

Die ‚Dev-NId' ist die Network Id des abgefragten Gerätes, bzw. die Network Id des Ursprungs der Power Info Meldung.
Dies können (aktuell 2020.05.18) das Master MX10(ec), MX10 Booster oder StEin[e] sein.

Daten Bytes für erweiterte Power Meldung, wenn MX10(ec/Booster) die Absender sind:

Byte	
3, 4	Pin 1: Status
5, 6	Pin 1: Spannung (in mV)
7, 8	Pin 1: Strom (in mA)
9, 10	Pin 2: Status
11, 12	Pin 2: Spannung (in mV)
13, 14	Pin 2: Strom (in mA)
15, 16	32V: Strom (in mA)
17, 18	12V: Strom (in mA)
19, 20	Netzteil Spannung (in mV)
21, 22	Temperatur

Hinweis:
Eine PC-Software MUSS darauf vorbereitet sein, das mehr Datenbytes gesendet werden!!
Je nach zukünftigen Anforderungen wird das MX10 mehr Status Informationen senden.

MODUL TAG LIST [0X18.0X0C]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5…DB8	DB9…
0x18	0x0C	0b00		8	Ziel-NID		Cmd	Dev	Tag	
0x18	0x0C	0b1n		13..46	Ziel-NID		Cmd	Dev	Tag	Tag Info

HINWEIS PC:

Der PC bekommt eine deutlich Umfangreichere Antwort, sofern ‚lange' Datagramme ‚Eingeschalten' sind.
Zu den 8 ‚Basis' Bytes kommt noch:

- Content
  Bit ‚0': ni (Nicht implementiert)
  Bit ‚1': RO, Wert kann nur gelesen werden
- Bezeichnung für Einheit (4 Byte, fix)
- Bezeichnung des Wertes, wie am MX32 angezeigt (Bis 32 Bytes variabel).

Somit kann die Gesamtlänge bis zu 46 Bytes betragen.

Tag Info:

Byte	Inhalt
9 ..10	Tag Flags
11..12	Zulässiger Minimal Wert für Parameter
13..14	Zulässiger Maximal Wert für Parameter
15..16	Derzeit Frei
17..20	Einheit
21..52	Text

UDP ONLY: NETWORK GROUP [0X1A]

PORT OPEN CMD [0X1A.0X06.B01]

Grp	Cmd	M	ID	DLC	DB 1 … 4	DB 5 .. 8	DB 9 …
0x1A	0x06	0b01		0, 8…	Options	App. Code	App Name

Mit diesem Datagramm kann ein Gerät die Ethernet Schnittstelle des MX10 ‚öffnen'.
Als Antwort bekommt das Gerät ein ‚Ping' vom MX10.

Mit Länge ‚0' erfolgt ein simples ‚Open' der Schnittstelle, weitere Angaben sind nicht nötig.

Wenn Länge grösser 8, dann:
Enthalten die ersten 4 Byte Optionen
In den Bytes 5 … 8 folgt eine Anwendungs-Kennung (Eindeutige Programm Kennung)
Danach können bis zu 24 Zeichen als Anwendungs-Name gesendet werden.
Dieser Name wird in diversen Anzeigen am MX10 und MX32 verwendet.

Wird KEIN Name gesendet, so wird die 32 Bit Kennung angezeigt, wird auch diese nicht gesendet, so wird schlicht ‚PC-Steuerung' angezeigt. Dies ist insbesondere bei größeren Anlagen mit mehreren Stellwerken extrem Anwender Unfreundlich.

Options	
Bit	Inhalt
0	Long Messages (Lange Nachrichten)
1	Free
2..7	Multi List
8	Show Deleted Obj
9 ..20	Free
21	All Fx (Alle Fx Datagramme 1:1 an PC)
22	Long GUI (Use Long GUI Datagramms)
23	Connection Lost ➔ Stop all Loco
24	Debug Output
25..27	Free
28..31	Version (Open Ack Version)

PORT OPEN ACK [0X1A.0X06.B11]

Grp	Cmd	M	ID	DLC	DB 1 … 4	DB 5 .. 8	DB 9 …
0x1A	0x06	0b11		0, 8…	Options	App. Code	System Info

System Antwort (Ack) auf Open Command

System Info	
Byte	Inhalt
1..4	Options, Entspricht Open Cmd Options. Flags können ‚0' gesetzt sein, wenn das MX10 die Option nicht unterstützt.
5..8	AppCode
9..12	System Ip
13…14	MX10 NetWork Id (SysNId)
15…18	MX10 Unique Id (SysUId)
19…50	System Name, Max. 31 Zeichen, unbenutzte Bytes mit 0x00 auffüllen.
51…54	System Tick
55…58	Hardware Build
59…62	Software Build
63…66	Software Date
67…70	Software Time
71…74	MiWi Build

Hinweis!!
Je nach Sinnhaftigkeit werden noch weitere System Infos ergänzt

SYS STATISTIK [0X1A.0X08]

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5 … DB[n]
0x1A	0x08	0b00		8	Ziel-NID		Statistk		
0x1A	0x08	0b1n			Ziel-NID		Statistk		Statistik Daten

Z[IMO]P[ROGRAMMABLE]S[CRIPT] [0X1C], PC COMMANDS

Nachdem gerade Script Datagramme von langen Nachrichten profitieren, gibt es per Ethernet / UDP Schnittstelle eben auch lange Script Datagramme.

ZPS MODIFY [0X1C.0X02], REQUEST

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x1C	0x02	0b00		4	ObjNId		Cmd		Key 1			
0x1C	0x02	0b00		6	ObjNId		Cmd		Key 1		Key 2	

Um die Header Info für ein Script abzufragen gibt es zwei Formen, entweder mit einem eindeutigen Kriterium (NId, Num) oder durch zwei Kriterien (Start/Ziel Taste).
Welche der beiden Formen gilt, entscheidet ‚Cmd'.

ZPS MODIFY [0X1C.0X02], CMD/ACK

Grp	Cmd	M	ID	DLC	DB 1, 2	DB 3, 4	DB 5 …..
0x1C	0x02	0b10		12..	ObjNId	Cmd	Siehe Command Tabelle
0x1C	0x02	0b11		12..	ObjNId	Cmd	Siehe Command Tabelle

ZPS MODIFY COMMANDS

CMD	Verwendung	Bytes	Beschreibung
0x0000	Abfrage per NId	3, 4	Script NId
0x0001	Abfrage per Num	3, 4	Script Nummer
0x0002	Abfrage per Start/Ziel Taste	3, 4	Start Taste
		5, 6	Ziel Taste
0x0010	Clear per NId	3, 4	Script NId Das angegebene Script wird gelöscht
0x0020	Modify Script per NId	3, 4	Script NId
		5, 6	Script Num
		7, 8	Start Taste
		9, 10	Ziel Taste
		11, 12	Virtuelle Fahrzeug/Zubehör Adresse
		13, 14	Virtuelle Fahrzeug Funktion/Ausgang
		15, 16	Options (actual free)
		17, ff	Bezeichnung für Script (Max. 24 Zeichen)

Abfragen werden immer per ACK mit einheitlicher Datenfolge beantwortet (NId, Num, Start, Ziel, …).
Alle nicht vorhandenen Felder haben den Wert 0xFFFF.

Bei einem Command entscheidet die Script NId ob eine Neuanlage erfolgen muss, oder eine Änderung zu machen ist.
Schlicht aufgrund der Tatsache das die jeweils angegebene Script-NId schon verwendet ist oder nicht.
Jedes Script kann einen Namen von bis zu 24 Zeichen haben.

SYSTEM DEBUG [0X2F]

SYSTEM DEBUG TEXT

Grp	Cmd	M	ID	DLC	DB1…4	DB5…8	DB5 …. n
0x2F	0x01	0b01		8..	System Ticks	Sys UId	Debug Text

Sofern aktiviert, sendet das MX10 alle internen Debug Meldungen mit diesem Event an die Ethernet Clients.
Die ‚System Ticks' sind die internen Millisekunden seit Power On. Diese können zur Synchronisierung der Nachrichten dienen. Bytes 5 bis 8 enthalten die System UId. Aber Byte 5 bis zur im DLC angegebenen Länge befindet sich der Text.

Eine Anwendung sollte diese Nachrichten in einem eignen File speichern, oder, sofern vorhanden, im eigenen Log eintragen. Dadurch kann man die internen Aktionen vom MX10 mit der jeweiligen Anwendung abgleichen.

FUNKTIONELLE EIGENSCHAFTEN

ABLAUF FAHRZEUG ‚AKTIVIEREN'

Mit dem im Folgenden beschriebenen Ablauf aktiviert ein Fahrpult bzw. sonstiges Steuergerät (z.B.: Computer) ein Fahrzeug, um dieses zu 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 kommen, so liegt ein Fehler vor.
5a	Ab hier kann das Fahrzeug in 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- bzw. Zustandsänderungen (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 als 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 Kommandologik, 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	

Hinweis:
Die Doku ist noch nachzutragen.

OBJECT CONTROL / STEIN:

Mit dem ZIMO StEin Modul wird eine neue objektorientierte 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 Anwendungen sein.

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

Die ‚Kurznummern' dürfen dabei im Bereich 1 … 65535 liegen. Die Kurznummer ‚0' ist für Broadcasts reserviert (z.B.: Status-/Existenzabfrage 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 (z.B.: 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:

MODUL TYPE KENNUNG

Gilt für Ping und Modul Info.

Kennung	Gerät
0x01nn	MX10
0x02nn	Z21
0x0205	Roco Booster 10806
0x0206	Roco Booster 10807
0x90nn	StEin
0x92nn	RoCo Besetztmelder 10808

Das zweite Byte der Kennung (‚nn') kann zur Unterscheidung von unterschiedlichen Hardware Versionen verwendet werden.

PIN ZUORDNUNGSTABELLE

Gerät	Pin	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	Decoder-Update
MX10	0b0001,0000	Konstant Strom
MX10	0b0010,0000	Ausgeschalten
MX10	0b0100,0000	Unterspannung
MX10	0b1000,0000	Überstrom

FEHLER CODES

MX10: PIN NUMMERN UND ERROR CODES

Pin	Verwendung
0x0000	Broadcast (Alle Pins)
0x0001, 0x0002	Schiene 1, 2
0x0003	Booster
0x0004	Sniffer
0x0010 … 0x0017	ABA Inputs (analog)
0x0020 … 0x0025	ABA Outputs
0x0100, 0x0101	CAN Buchse 1, 2 (1=ZIMO, 2=eXtended)
0x0102, 0x0103	X-PressNet 1, 2

Error Code	Verwendung
0x0000	Kein Fehler, Normalbetrieb
0x0100	CAN Buffer ‚Überlauf'. Meldung wird bei ca. 75% Auslastung gesendet, ab hier sollte eine PC Software keine weiteren Befehle senden. Ausnahme sind alle System Internen Befehle (Stopp, Schiene AUS, …)
0x0101	**CAN Controller ‚Reset'.** **Dieser Fehler kommt NUR bei Nutzung der MX8 Module zum Tragen!!** **Wenn der PC diese Fehlermeldung erhält, so MUSS er bis zur Quittierung (Error Code 0x0000) warten. Außerdem muss er dann selber für den richtigen Status der MX8 Module Sorge tragen.** **Hinweis:** **Bei Verwendung von MX8 bitte entsprechendes Kapitel lesen.**

Hinweis: Die Fehlercodes und deren Bedeutung sind derzeit noch nicht definiert.

STEIN: PIN NUMMERN UND ERROR CODES

Pin	Verwendung
0x0000	Modul Global
0x0001	Kommunikation
0x0100 … 0x0107	Gleisabschnitte
0x1000 … 0x1007	Leistungsaus/Eingänge
0x2000 … 0x200F	Digitalaus/Eingänge

Error Code	Verwendung
Modul Global (Type=0x0001)	
0x0001	Bad Parameter (z.B.: Gleisabschnitt 0x0122)
0x0002	Keine Gleis Spannungsversorgung
0x0003	Keine Zubehör Spannungsversorgung
0x0004	Kein DCC (Schienen) Signal
0x0005	Keine CAN Spannung
0x0006	Keine +20V
0x0007	Keine +5V
0xFFFF	Derzeit nicht implementiert (Gilt Global, für alle Abschnitte, Ein- und Ausgänge)
Kommunikation (Type=0x0002)	
0x0001	Bad Parameter
0x0002	Befehl nicht ausführbar: Abschnittsabfrage ohne Gleisspannung
0x0003	Befehl nicht ausführbar: RailCom Abfrage ohne RailCom Lücke
Gleisabschnitt	
Leistungs Ein-/Ausgänge (Type=0x1000 … 0x1008)	
0x0001	Bad Parameter (z.B.: Ausgang 0x1037)
0x0005	Befehl nicht ausführbar: Keine +20V vorhanden
0x0006	Befehl nicht ausführbar: Keine +5V vorhanden

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 6 Bit 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	Soll Richtung	Soll Richtung
11	Ist Richtung	Ist 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

Noch ergänzen:
Was bedeutet in diesem Zusammenhang Info Command?

EPOCHEN

Fahrzeuge sind gemäß des Datums Ihres ersten Streckendienstes in Epochen eingeteilt.
In Europa gelten typischerweise die Epochen der Deutschen Eisenbahn. England verfügt über eine eigenen Einteilung.

Epoche	Jahr	Schwerpunkt Loks	Ökonomie
Epoche I	1835 bis 1920	Der Beginn der Bahnfahrt und die Entwicklung der ersten Dampflokomotiven.	Regionale Initiativen bis zur Industrialisierung
Epoche II	1920 bis 1950	Dampf hat die Vorherrschaft, die ersten elektrischen Lokomotiven werden gebaut, aber sie sind noch nicht im allgemeinen Gebrauch.	Große Bahngesellschaften entstehen und die regionalen Initiativen verlieren sich in ihnen.
Epoche III	1950 bis 1970	Dampfantrieb verschwindet langsam, E-Loks und Dieselloks sind im Vormarsch.	Wiederaufbau nach dem zweiten Weltkrieg und neue Möglichkeiten durch Entwicklung
Epoche IV	1970 bis 1990	Dampf ist komplett verschwunden, primär sind elektrisch- und dieselbetriebene Loks im Einsatz.	Internationaler Verkehr nimmt zu
Epoche V	1990 bis 2007	Hochgeschwindigkeitszüge sind im Kommen, Mehrfarbigkeit nimmt deutlich zu.	Globalisierung heißt das Zauberwort
Epoche VI	ab 2007	Private Unternehmen in einem internationalen Wettbewerb.	Private Transportunternehmen

Zusätzlich zu diesen Hauptepochen gibt es auch noch ‚Sub-Epochen', welche mit Kleinbuchstaben nach der Hauptepoche gekennzeichnet sind.

SPEZIELLE NID'S

NID Min.	NID Max	Anzahl	
0x7000	0x7000	1	Namen für Gruppen
0x7100	0x71FF	256	Herstellernamen

EINGETRAGENE MARKENZEICHEN

mfx®	Gebr. Märklin & Cie. GmbH
Motorola®	Motorola Inc., Tempe-Phoenix, USA
ZIMO	ZIMO Elektronik GmbH
HLU	ZIMO Elektronik GmbH
DCC	NMRA
RailCom®	Lenz Elektronik GmbH
LocoNet	Digitrax Inc.
Android	Google Inc.
iPad, iPhone	Apple Inc.
iOS	Apple Inc.
App Store	Apple Inc.
Google Play	Google Inc.

HAFTUNGSAUSSCHLUSS

Information von der Firma ZIMO Elektronik GmbH:

Die Firma ZIMO Elektronik GmbH erklärt ausdrücklich, in keinem Fall für den Inhalt in diesem Dokument oder für in diesem Dokument angegebene weiterführende Informationen rechtlich haftbar zu sein. Die Rechtsverantwortung liegt ausschließlich beim Verwender der angegebenen Daten oder beim Herausgeber der jeweiligen weiterführenden Information.

Für sämtliche Schäden die durch die Verwendung der angegebenen Informationen oder durch die Nicht-Verwendung der angegebenen Informationen entstehen übernimmt ZIMO Elektronik GmbH ausdrücklich keinerlei Haftung.

Die Firma ZIMO Elektronik GmbH übernimmt keinerlei Gewähr für die Aktualität, Korrektheit, Vollständigkeit oder Qualität der bereitgestellten Informationen.
Haftungsansprüche, welche sich auf Schäden materieller, immaterieller oder ideeller Art beziehen, die durch die Nutzung oder Nichtnutzung der dargebotenen Informationen verursacht wurden, sind grundsätzlich ausgeschlossen.

Die Firma ZIMO Elektronik GmbH behält es sich vor, die bereit gestellten Informationen ohne gesonderte Ankündigung zu verändern, zu ergänzen oder zu löschen.
Alle innerhalb des Dokuments genannten und gegebenenfalls durch Dritte geschützten Marken- und Warenzeichen unterliegen uneingeschränkt den Bestimmungen des jeweils gültigen Kennzeichenrechts und den Besitzrechten der jeweiligen eingetragenen Eigentümer.

Sollten Teile oder einzelne Formulierungen des Haftungsausschlusses der geltenden Rechtslage nicht, nicht mehr oder nicht vollständig entsprechen, bleiben die übrigen Teile des Haftungsausschlusses in ihrem Inhalt und ihrer Gültigkeit davon unberührt.

KONFIGURATIONSPARAMETER ALLGEMEIN

Die folgenden Parameter sollten von jedem Modul implementiert werden.

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 GES. MODUL

Object	Item	Beschreibung	Min.	Max.
0x1000	0x0000	Stein Modulnummer	1	255
	0x0001	Hardware Limit: Minimal Strom [mA]	Read Only, typ. 1mA	
	0x0002	Hardware Limit: Maximal ‚P' Strom [mA]	Read Only, typ. 8.000mA	
	0x0003	Hardware Limit: Maximal ‚N' Strom [mA]	Read Only, typ. 10.000mA	
	0x0004	Stromquelle für Gleise (MX10 Ausgang 1/2, Netzteil, …)		
	0x0005			

EINSTELLUNGEN FÜR SEKTIONEN

Object	Item	Beschreibung	Min.	Max.	Def.
	0x0020	Globale Abschnittnummer (1 … 8)	1	255	
	0x0021	Betriebsform / work type	0		
	0x0022	Abschnitt Art / Section Type: Bit 0: Besetzmeldung Bit 1: HLU Bit 2: ZACK Bit 3: RailCom Chn1 Bit 4: RailCom Chn2 Bit 8: Kehrschleife			
	0x0024	HLU-Einstellung / speed setting			
	0x0025	HLU Funktions Bits (F0 … F15)	0	255	
0x1001	0x0026	HLU Funktions Bits (F16 … F28)			
….					
0x1008	0x0030	Stromlimit nieder in mA			3000
	0x0031	Stromlimit nieder: Zeitdauer [mS]			500
	0x0032	Stromlimit nieder: Wiedereinschaltung [mS]			500
	0x0033	Stromlimit nieder: Anzahl Wiederholungen			
	0x0034	Stromlimit hoch in mA			5000
	0x0035	Stromlimit hoch: Zeitdauer [ms]			500
	0x0036	Stromlimit hoch: Wiedereinschaltung [ms]			500
	0x0037	Stromlimit hoch: Anzahl Wiederholungen			
	0x0038	Stromlimit Kurzschluss im mA		8000	
	0x0039	Stromlimit Kurzschluss: Zeitdauer [ms]	20	100	30
	0x003A	Stromlimit Kurzschluss: Wiedereinschaltung [ms]	100	1000	300

EINSTELLUNGEN FÜR SEKTIONEN

Object	Item	Beschreibung	Min.	Max.	Def.
	0x0040	Besetztschwelle normal: Strom [mA]			
	0x0041	Besetztschwelle normal: Zeit ‚Besetzt Test' [mS]	10	160	160
	0x0042	Besetztschwelle normal: Zeit ‚Frei Test' [mS]	10	160	160
	0x0044	Besetztschwelle feucht: Strom			
	0x0045	Besetztschwelle feucht: Zeit ‚Besetzt Test' [mS]			
0x1001	0x0046	Besetztschwelle feucht: Zeit ‚Frei Test' [mS]			
....					
0x1008	0x0048	Besetztschwelle Regen: Strom			
	0x0049	Besetztschwelle Regen: Zeit ‚Besetzt Test' [mS]			
	0x004A	Besetztschwelle Regen: Zeit Zeit ‚Frei Test' [mS]			
	0x0050	Verzögerung Frei ➔ Besetzt [mS]	0		0
	0x0051	Verzögerung Besetzt ➔ Frei [mS]	320	8000	1000
	0x0052	Verzögerung Zugnummern ‚Leer' Meldung [mS]	1000	8000	1000

PIN CONTROL

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

KONFIGURATIONSPARAMETER FÜR ROCO 10808

Die folgende Liste enthält die Konfigurationsparameter für das Roco 10808 Rückmelde-Modul.
Die angegeben Werte dienen als Parameter für die Config Modul Value Datagramme.

EINSTELLUNGEN FÜR GES. MODUL

Object	Item	Beschreibung	Min.	Max.
0x1000	0x0000	Roco 10808 Modulnummer	1	255
	0x0001	Hardware Limit: Minimal Strom [mA]	Read Only, typ. 1mA	
	0x0002	Hardware Limit: Maximal ‚P/N' Strom [mA]	Read Only, typ. 4.000mA	

EINSTELLUNGEN FÜR SEKTIONEN

Object	Item	Beschreibung	Min.	Max.	Def.
	0x0020	Globale Abschnittnummer (1 … 8)	1	255	
	0x0022	Abschnitt Art / Section Type: Bit 0: Besetzmeldung Bit 3: RailCom			
	0x0038	Stromlimit Kurzschluss im mA		4000	2500
0x1001					
….	0x0040	Besetztschwelle normal: Strom [mA]			
0x1008	0x0041	Besetztschwelle normal: Zeit ‚Besetzt Test' [mS]	10	160	160
	0x0042	Besetztschwelle normal: Zeit ‚Frei Test' [mS]	10	160	160
	0x0050	Verzögerung Frei ➔ Besetzt [mS]	0	0	0
	0x0051	Verzögerung Besetzt ➔ Frei [mS]	320	8000	1000
	0x0052	Verzögerung Zugnummern ‚Leer' Meldung [mS]	1000	8000	1000

GLOSSAR

Begriff	Erklärung
UID	Weltweit eindeutig 32 Bit Nummer (Unique Identifier). Diese wird typischerweise während des Anmeldeprozesses verwendet.
NID	Network ID, 16 Bit Nummer welche im laufenden Betrieb zur Adressierung der Module, Fahrzeuge, Decoder,… 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, Decoder, 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 PORT für DCC ‚Basic' Decoder	03.03.2014	03.03.2014
0x01	0x02	Accessory PORT 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);
}
```

IMPLEMENTIERUNGS HINWEISE UND ABLÄUFE FÜR PC SOFTWARE:

PC SOFTWARE, VERBINDUNGSAUFBAU, VCOM:

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' Wartezeit max. 1Sec. (Typisch 10mS). Wenn in dieser Zeit keine Reaktion vom MX10 kommt ➔ Fehler			
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/Uhrzeit'	Cmd	‚Modul Info Datum/Uhrzeit'	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

Die typische Reaktionszeit auf ‚interne' Befehle beträgt ca. 50mS. Bei hoher System Auslastung (Insbesondere bei vielen MX8/MX9 Modulen) kann die Reaktionszeit auf ca. 500mS ansteigen.

Bei ‚Schienenbefehlen' und Abfrage Befehle an die MX8/MX9 kann es zu deutlich längeren Reaktionszeiten kommen. Diese sollten jedoch in keinem Falle über 1000mS sein. Eine PC Software sollte Reaktionszeiten ab 1000mS als Fehler Zustand des MX10 betrachten / behandeln.

PC SOFTWARE, VERBINDUNGSAUFBAU, LAN/UDP:

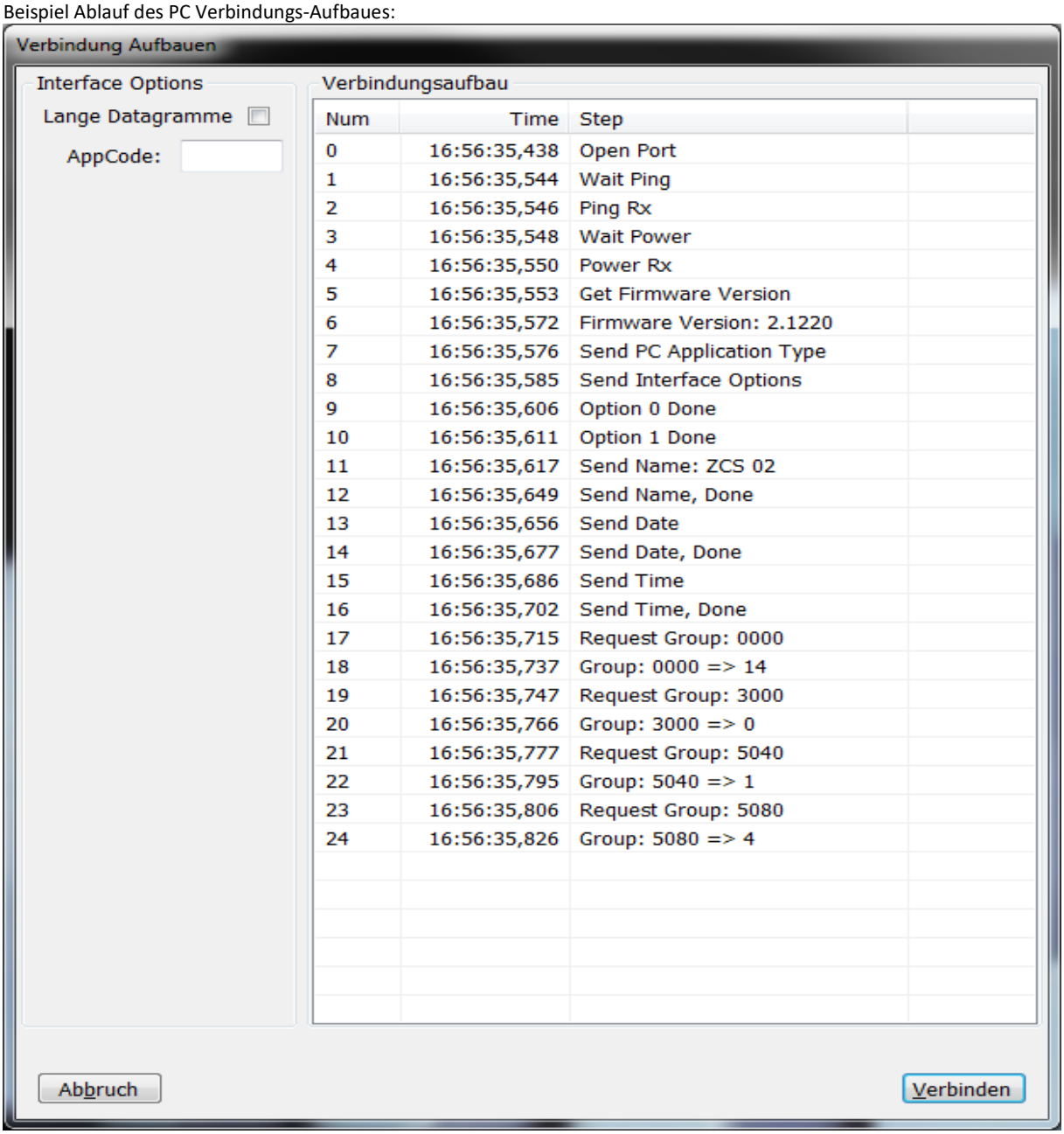
Folgender Ablauf sollte von einer PC Software zum Aufbau der MX10 Verbindung eingehalten werden:

PC		**MX10**	
Datagramm	**Mode**	**Datagramm**	**Tx/Rx**
Open Command	Tx	Ping	
Warten auf ‚Ping' Wartezeit max. 1Sec. (Typisch 10mS). Wenn in dieser Zeit keine Reaktion vom MX10 kommt ➔ Fehler			
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/Uhrzeit'	Cmd	‚Modul Info Datum/Uhrzeit'	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

Die typische Reaktionszeit auf ‚interne' Befehle beträgt ca. 50mS. Bei hoher System Auslastung (Insbesondere bei vielen MX8/MX9 Modulen) kann die Reaktionszeit auf ca. 500mS ansteigen.

Bei ‚Schienenbefehlen' und Abfrage Befehle an die MX8/MX9 kann es zu deutlich längeren Reaktionszeiten kommen. Diese sollten jedoch in keinem Falle über 1000mS sein. Eine PC Software sollte Reaktionszeiten ab 1000mS als Fehler Zustand des MX10 betrachten / behandeln.

Beispiel Ablauf des PC Verbindungs-Aufbaues:



PC SOFTWARE, FAHRZEUG STEUERN:

Damit eine PC Software ein Fahrzeug steuern kann, stehen 3 Ansätze zur Verfügung. Um die Auswirkungen dieser Ansätze besser zu verstehen, muss man folgende funktionelle Eigenschaften des ZIMO Systems verstehen (MX10 + MX32):

MX10 PRIORITÄTSLOGIK:

Da die Schienenbandbreite ja deutlich geringer ist als jene des PC Interfaces und auch des CAN Busses müssen alle Schienen relevanten Befehle mit sinnvollen Prioritäten gepuffert werden. Im MX10 ist daher folgende Prioritätslogik implementiert.
HINWEIS:
Die folgende Beschreibung ist nur ein grundsätzlicher Überblick, im Detail gibt es natürlich technische Feinheiten, welche hier nicht beschrieben werden.

Priorität	Bezeichnung	Zweck/Aufgabe, Beschreibung
0	Änderung Ca. 50% der Bandbreite	Sobald ein Befehl die derzeitigen Sendedaten (DCC Befehle) verändert kommt das jeweilige Fahrzeug in diese Prioritätsstufe. Es verbleibt in dieser Stufe bis die Änderung zumindest 5x oder 1 Sec. gesendet wurde. (Längerer Wert)
1	Aktiv, Vordergrund Ca. 25% der Bandbreite	Nach einer Änderung ‚fallen' Fahrzeuge auf diese Priorität. Sie verbleiben für mind. 5 Sec. in dieser Stufe. Jedes Mal wenn das MX10 einen ‚Aktiv' Befehl erhält, wird diese Zeit für weitere 5 Sec. verlängert. Ebenso werden durch den ‚Aktiv' Befehl, Fahrzeuge mit niedriger Priorität auf diese angehoben.
2	Passiv, Rückholspeicher Ca. 15% der Bandbreite	Nach Ablauf der 5 Sec. ‚Vordergrund' Priorität kommen Fahrzeuge in die ‚Passiv' Priorität und verbleiben in dieser für die nächsten 25 Sec. Durch den Fahrzeug Stack Befehl (Grp=0x02, 0x11) können Fahrzeug in diese Priorität verschoben werden. ACHTUNG!! Der Stack Befehl wirkt in beide Richtungen der Prioritätslogik. Es können also ‚Aktiv' Fahrzeuge vor Ablauf der Zeit in diese Priorität verschoben werden, natürlich aber auch ‚Standby' Fahrzeuge in diese Priorität angehoben werden. Die Änderungspriorität kann jedoch NICHT beeinflusst werden.
3	Standby Ca. 10% der Bandbreite	Alle anderen Adressen und auch System interne Befehle (z.B.: Clock, UID, Service Request) werden in dieser Prioritätsstufe abgearbeitet. Ab ca. 10 aktiven Fahrzeugen kann es in dieser Stufe, je nach Anzahl, durchaus zu erheblichen Latenzzeiten kommen.

FAHRZEUG STEUERN, ‚SIMPLE DRIVE':

Die ‚Simple Drive' Methode ist wie der Name schon sagt, die einfachste Art Fahrzeuge per PC zu steuern. Bei dieser Methode sendet der PC schlicht Fahr- und Funktionsbefehle ohne weitere Rücksichtnahme auf das System.
In diesem Falle entscheidet das MX10 anhand der Befehlsdaten in welcher Priorität die Befehle an die Schiene zu senden sind und ‚exekutiert' den Befehl.
System intern (also MX32) werden diese Befehle gespiegelt und kurzfristig angezeigt. Da die PC Software auf diese Art und Weise jedoch das Fahrzeug NICHT übernimmt, gibt es auch keinen Übernahme Dialog oder sonstige Zusammenarbeit zwischen System (MX32 + MX10) und der PC Software.

FAHRZEUG STEUERN, ‚FAHRPULT STYLE':

Bei dieser Methode verhält sich eine PC Software wie ein ‚reales' Fahrpult.
In diesem Falle muss sich durch den ‚Aktiv' Befehl Fahrzeuge übernehmen, die Übergabe Logik implementieren und auch etwa alle 500mS die Fahrzeug aktiv Meldung senden.
Natürlich kann eine PC Software dies für mehrere Fahrzeug gleichzeitig machen, bzw. einen Teil der Fahrzeuge im ‚Fahrpult Style' steuern und andere Fahrzeuge mit einer der beiden anderen Methoden.

ACHTUNG:
Wenn eine PC Software mehrere Steuermethoden anwendet, so muss sie selber für einen sauberen Übergang zwischen diesen Methoden sorgen.

FAHRZEUG STEUERN, ‚VOLLWERTIGES STELLWERK':

Hinweis:
Die Doku ist noch nachzutragen.

PC SOFTWARE, FAHRZEUG GESCHWINDIGKEITSTABELLE:

Abgleich Geschwindigkeitstabelle ZIMO App – MX10

Um diesen Abgleich durchzuführen sind folgende Datagramme zu nutzen:

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB5	DB6	DB7	DB8
0x07	0x18	0b00		6	SrcNID		Fahrzeug-NID		1	0		

Als ‚ScrNId' ist die NId des jeweiligen MX10 einzusetzen, als ‚Fahrzeug-NID' ist die Adresse des gewünschten Fahrzeuges einzusetzen. Datenbyte 5 ist immer ‚1', Datenbyte 6 immer ‚0'

Aufgrund dieser Abfrage sendet das MX10 folgende Antwort:

Grp	Cmd	M	ID	DLC	DB1	DB2	DB3	DB4	DB 5…
0x07	0x18	0b01		8	Fahrzeug-NID		1	5	

In den Datagramm Databytes 5 … befinden sich folgende Daten
Diese enthält so viele Speed Pärchen wie unter ‚Items' angegeben.

DB	Len	Inhalt
1 .. 2	2	Tacho Design
3 .. 4	2	Tacho Farbe
5 .. 6	2	Fahrstufe ‚0' ➔ Wert immer ‚0'
7 .. 8	2	Geschwindigkeit ➔ immer ‚0'
9 .. 10	2	Fahrstufe ‚a'
11 .. 12	2	Geschwindigkeit passend zu Fahrstufe ‚a'
13 .. 14	2	Fahrstufe ‚b'
15 .. 16	2	Geschwindigkeit passend zu Fahrstufe ‚b'
17 .. 18	2	Fahrstufe ‚c'
19 .. 20	2	Geschwindigkeit passend zu Fahrstufe ‚c'
21 .. 22	2	Fahrstufe ‚126'
23 .. 24	2	Geschwindigkeit passend zu Fahrstufe ‚126'

Fahrstufe/Geschwindigkeit ‚0' (Datenbytes 5 … 8) sind immer ‚0'
Die Pärchen ‚a', ‚b', ‚c' sind frei wählbar, sofern gilt: 0 < Fahrstufe ‚a' < Fahrstufe ‚b' < Fahrstufe ‚c' Fahrstufe ‚126'
Das letzte Pärchen enthält die Vmax, daher ist die Fahrstufe immer ‚126', die jeweils passende Geschwindigkeit ist wieder frei festlegbar.

FEHLERBEHANDLUNG FÜR MX8 MODULE

Leider verursachen die MX8 Module Kommunikationsfehler.
Auf diese MUSS eine PC Anwendung entsprechend reagieren.

Sobald das MX10 einen Kommunikationsfehler erkennt, meldet es diesen an den PC (Grp=0x00, Cmd=0x10).
Sobald der PC dieses ACK empfängt ist folgende Vorgangsweise einzuhalten:

1. Bis auf weiteres dürfen KEINE weiteren Befehle gesendet werden.
2. Die Anwendung MUSS davon ausgehen das alle MX8, zumindest aber das auslösende MX8 einen illegalen Zustand hat.
3. Die Anwendung MUSS auf das erste Accessory EVT (PORT oder Port) vom auslösenden MX8 warten.
4. Die Anwendung MUSS danach sämtliche MX8 entsprechend ansteuern, also die jeweils gewünschten Stellungen erneut an die MX8 senden.

TYPE DEFINITIONS FÜR PC IMPLEMENTIERUNG:

Um die Protokoll Implementierung für PC Programme zu vereinfachen, sind hier die ZCan Datagramme als ‚C' Typedefs zu finden.

[0X00.0X00], TYPE DEFINITION FÜR SYSTEM POWER

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16NId          : 16;   //    00..15: System NId
        Int64U  f08Pin          :  8;   //    16..23: System Pin
        Int64U  f08Mode         :  8;   //    24..31: System Mode
        Int64U  f32dummy        : 32;   //    32..63: Unused
    };
} __tZ20SysMode;
```

[0X01.0X00], TYPE DEFINITION FÜR ACCESSORY STATE

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16NId          : 16;   //    00..15: Accessory NId
        Int64U  f16State        : 16;   //    16..31: Status Info
        Int64U  f16Data1        : 16;   //    32..47: Zusatz Daten 1
        Int64U  f16Data2        : 16;   //    48..63: Zusatz Daten 2
    };
} __tZ20AccState;
```

[0X01.0X02], TYPE DEFINITION FÜR ACCESSORY PORT

```
typedef union
{
    Int64U      u64Data;
    Struct
    {
        Int64U  f16NId          : 16;   //      00..15: Accessory NId
        Int64U  f16Type         : 16;   //      16..31: Port Type
        Int64U  f32Port         : 32;   //      32..63: Port Data
    };
    struct
    {
        Int64U  f16Mx9Id        : 16;   //      00..15: Mx9 NId
        Int64U  f16Mx9IO        : 16;   //      16..31: Port Type
        Int64U  f01ALA01        :  1;   //      32:     ALA Ausgang 01
        Int64U  f01ALA02        :  1;   //      33:     ALA Ausgang 02
        Int64U  f01ALA03        :  1;   //      34:     ALA Ausgang 03
        Int64U  f01ALA04        :  1;   //      35:     ALA Ausgang 04
        Int64U  f01ALA05        :  1;   //      36:     ALA Ausgang 05
        Int64U  f01ALA06        :  1;   //      37:     ALA Ausgang 06
        Int64U  f01ALA07        :  1;   //      38:     ALA Ausgang 07
        Int64U  f01ALA08        :  1;   //      39:     ALA Ausgang 08
        Int64U  f01ALA09        :  1;   //      40:     ALA Ausgang 09
        Int64U  f01ALA10        :  1;   //      41:     ALA Ausgang 10
        Int64U  f01ALA11        :  1;   //      42:     ALA Ausgang 11
        Int64U  f01ALA12        :  1;   //      43:     ALA Ausgang 12
        Int64U  f01ALA13        :  1;   //      44:     ALA Ausgang 13
        Int64U  f01ALA14        :  1;   //      45:     ALA Ausgang 14
        Int64U  f01ALA15        :  1;   //      46:     ALA Ausgang 15
        Int64U  f01ALA16        :  1;   //      47:     ALA Ausgang 16
        Int64U  f01ALA17        :  1;   //      48:     ALA Ausgang 17
        Int64U  f01ALA18        :  1;   //      49:     ALA Ausgang 18
        Int64U  f01ALA19        :  1;   //      50:     ALA Ausgang 19
        Int64U  f01ALA20        :  1;   //      51:     ALA Ausgang 20
        Int64U  f01ALA21        :  1;   //      52:     ALA Ausgang 21
        Int64U  f01ALA22        :  1;   //      53:     ALA Ausgang 22
        Int64U  f01ALA23        :  1;   //      54:     ALA Ausgang 23
        Int64U  f01ALA24        :  1;   //      55:     ALA Ausgang 24
        Int64U  f01ALA25        :  1;   //      56:     ALA Ausgang 25
        Int64U  f01ALA26        :  1;   //      57:     ALA Ausgang 26
        Int64U  f01ALA27        :  1;   //      58:     ALA Ausgang 27
        Int64U  f01ALA28        :  1;   //      59:     ALA Ausgang 28
        Int64U  f01ALA29        :  1;   //      60:     ALA Ausgang 29
        Int64U  f01ALA30        :  1;   //      61:     ALA Ausgang 30
        Int64U  f01ALA31        :  1;   //      62:     ALA Ausgang 31
    };
} __tZ20AccPort;
```

[0X01.0X04], TYPE DEFINITION FÜR ACCESSORY PIN4

```
typedef union
{
    Int64U      u64Data;
    Struct
    {
        Int64U  f16NId          : 16;   //      00..15: Accessory NId
        Int64U  f08Pin          :  8;   //      16..23: Accessory Pin
        Int64U  f08Val          :  8;   //      24..31: Accessory Pin Value
        Int64U  f32unused       : 32;   //      32..63: Nicht verwendet
    };
} __tZ20AccPin4;
```

[0X01.0X05], TYPE DEFINITION FÜR ACCESSORY DATA

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16NId          : 16;   //      00..15: Accessory NId
        Int64U  f08Pin          :  8;   //      16..23: Accessory Pin
        Int64U  f08Num          :  8;   //      24..31: Accessory Pin Item
        Int64U  f32Data         : 32;   //      32..63: Pin Data
    };
    struct
    {
        Int64U  f16dummy1       : 32;   //      00..15: Accessory NId
        Int64U  f16Loco1        : 16;   //      32..47: Pin Fahrzeug 1
        Int64U  f16Loco2        : 16;   //      48..63: Pin Fahrzeug 2
    };
} __tZ20AccData;
```

[0X01.0X06], TYPE DEFINITION FÜR ACCESSORY PIN6

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16NId          : 16;   //      00..15: Accessory NId
        Int64U  f08Pin          :  8;   //      16..23: Accessory Pin
        Int64U  f08Typ          :  8;   //      24..31: Accessory Pin Type
        Int64U  f16Data         : 16;   //      32..47: Pin Data
        Int64U  f16dummy        : 16;   //      48..63: unused
    };
} __tZ20AccPin6;
```

[0X02.0X00], TYPE DEFINITION FÜR FAHRZEUG STATE

```
#pragma pack(1)
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U      f16NId          : 16;   //      00..15: Loco NId
        Int64U      f08Train        :  8;   //      16..23: Zug NId
        Int64U      f01BiDi         :  1;   //      24:     BiDi
        Int64U      f01ZACK         :  1;   //      25:     ZACK
        Int64U      f01DirCmd       :  1;   //      26:     Vorgabe  Richtung
        Int64U      f01DirAck       :  1;   //      27:     Schienen Richtung
        Int64U      f02EastWest     :  2;   //      28..29: Ost/West
        Int64U      f01SpeedZ       :  1;   //      30:     Geschwindigkeit '0'
        Int64U      f01EStop        :  1;   //      31:     Emergency Stop
        Int64U      f01CabEW        :  1;   //      32:     Fahrpult E/W
        Int64U      f07Flags        :  7;   //      33..39: Weitere Flags
        Int64U      f08CtrlSec      :  8;   //      40..47: Letzter Steuer Tick
        Int64U      f16CtrlNId      : 16;   //      48..63: Steuer Gerät
    };
} __tZ20LocoState;
#pragma pack()
```

[0X02.0X05], TYPE DEFINITION FÜR FAHRZEUG SPECIAL FUNCTION

```
#pragma pack(1)
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16NId          : 16;   //    00..15: Loco NId
        Int64U  f16SxN          : 16;   //    16..31: Special Function Number
        Int64U  f16Val          : 16;   //    32..47: Special Function Value
        Int64U  f16dummy1       : 16;   //    48..63: Unused
    };
    struct
    {
        Int64U  f32dummy        : 32;       //    00..31: Loco NId, SxNum
        Int64U  f15Val          : 15;       //    32..46: Special Function Value
        Int64U  f01TEW          :  1;       //    47:     Track East/West
        Int64U  f16dummy2       : 16;       //    48..63: Unused
    };
} __tZ20LocoSxNum;
#pragma pack()
```

[0X05.0X01], TYPE DEFINITION FÜR ZUG FAHRZEUG SUCHE

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16TrainNId     : 16;   //    00..15: Train NId
        Int64U  f16LocoNId      : 16;   //    16..31: Loco  NId
        Int64U  f16OwnerNId     : 16;   //    32..47: 'Eigentümer' NId
        Int64U  f08Flag         :  8;   //    48..55: Flags
        Int64U  f08Idx          :  8;   //    56..63: Index
    };
} __tZ20TrainPartList;
```

[0X05.0X02], TYPE DEFINITION FÜR FAHRZEUG ZUG SUCHE

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16LocoNId      : 16;   //    00..15: Loco  NId
        Int64U  f16TrainNId     : 16;   //    16..32: Train NId
        Int64U  f16OwnerNId     : 16;   //    32..47: 'Eigentümer' NId
        Int64U  f08Flag         :  8;   //    48..55: Flags
        Int64U  f08Idx          :  8;   //    56..63: Index
    };
} __tZ20TrainPartFind;
```

[0X05.0X03], TYPE DEFINITION FÜR ZUG ‘EIGENTÜMER’

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16TrainNId     : 16;   //    00..15: Train NId
        Int64U  f16OwnerNId     : 16;   //    16..31: New Owner NId
        Int64U  f16TrainNum     : 16;   //    32..47: New Train Number
        Int64U  f16Flag         : 16;   //    48..63: unused
    };
} __tZ20TrainOwnerSet;
```

[0X05.0X08], TYPE DEFINITION FÜR ZUG FAHRZEUG ‘EIGENTÜMER’

```
typedef union
{
    Int64U      u64Data;
    struct
    {
        Int64U  f16TrainNId     : 16;   //    00..15: 'Zug' Nummer
        Int64U  f16LocoNId      : 16;   //    16..31: Fahrzeug NId
        Int64U  f16OwnerNId     : 16;   //    32..47: 'Eigentümer' NId
        Int64U  f08OwnerNum     :  8;   //    48..55: 'Eigentümer' Num
        Int64U  f08State        :  8;   //    56..63: Status
    };
} __tZ20TrainPartSet;
```

CHANGE LOG:

Datum	Änderung
2015.03.28	Ergänzung Fehlermeldungen, Insbesondere MX8 Korrektur RCS Command Group **Fehlerbehandlung für MX8 Module**
2015.03.30	Nachtrag für ‚alte' Befehle. Accessory Port hat nun wie schon lange vorgesehen zusätzlich einen ‚Type' und 16 Bit Values
2015.03.31	Beschreibung für ‚Field Actuator' ergänzt
2015.04.01	Data Name eXtended [0x21]: Fehler in der Byte Zählung behoben
2015.04.02	Grp 0x06 Com 0x0B: Cfg Val korrigiert (5 -> 4 Byte)
2015.04.07	Accessory Port Befehl in PIN4 und Pin6 getrennt. Die PIN4 Befehle sind 3 bzw. 4 Byte lang verhalten sich so wie in den ‚alten' Dokumentationen beschrieben. Die erweiterten Port Befehle haben ab sofort eine eigene Command Kennung [0x06]. Haftungsausschluß Roco wurde ergänzt
2015.04.14	Accessory Port Befehle für StEin Module erg.
2015.04.14	Config Befehle (Grp 0x05) für StEin Module erg.
2015.04.15	Grp 0x01 Com 0x06: DLC korrigiert (3 -> 4 Byte), Com korrigiert (0x04 -> 0x06); Grp 0x06 Com 0x0B, M=0b11: Val in BD7 ergänzt Grp 0x07 Com 0x00, M=0b11: Com korrigiert (0x01 -> 0x00) Diverse eindeutige falsche DLC korrigiert
2015.04.16	Ergänzung Beschreibung für MX8 Fehlmeldungen, insbesondere für möglicherweise falsche PORT Meldungen.
2015.04.23	Änderung Reihenfolge bei Abfrage von Namen (Grp=0x07, Req=0x10)
2015.04.28	Korrektur für Loco Info
2015.04.29	Ergänzung Info Type 8, MiWi Channel der Zentrale
2015.04.29	Modul Info in Gruppe Info verschoben (War fälschlicherweise in Gruppe NetWork)
2015.05.04	Grp 0x01 Com 0x04 wieder auf alte Stand zurückgesetzt
2015.05.11	Korrektur für Grp=0x00, Cmd=0x00 Ergänzung der Modi für SSP, Fahrstufe ‚0' bzw. Emergency SSP
2015.05.13	Grp=0x02, Cmd=0x04: Ergänzung Port Nummer um ‚Valid' Bit.
2015.05.13	Grp=0x01, Cmd=0x04 erneut korrigiert; Grp=0x01, Cmd=0x06 ergänzt;
2015.06.16	Modul Config, Insbesondere MX8/MX9 Group 0x05 ab sofort frei. Modul Config nun in Info Gruppe [0x08]
2015.06.17	Grp=0x02, Cmd=0x02, Accessory Mode Beschreibung um MX8/MX9 Modi ergänzt
2015.07.20	Das TSE Mode Ack enthält nun die Serv. Prog. Spannung und Strom
2015.08.24	Grp=0x04, Cmd=0x05 um Signalbilder und Beschreibung ergänzt Grp=0x04, Cmd=0x0C Feld Folge richtiggestellt, Feld Action Werte festgelegt
2015.08.26	Grp Angaben in den Datagramm Tabellen der Info Group richtiggestellt
2015.09.04	Grp=0x04, Cmd=0x05 RCS Look Ahead Signal Definition überarbeitet
2015.11.10	MAN/Rg Funktion aus Speed entfernt und als eigene Funktion definiert
2016.01.11	Fehler Code Tabelle MX10, StEin überarbeitet
2018.11.28	Ergänzung: Modul Eigenschafts Abfrage [0x08.0x0C]
2019.05.17	Ergänzung: Modul Eigenschafts Abfrage [0x08.0x0C], Detail & PC Doku
2019.05.17	Neues Datagramm: Multilimit [0x01.0x09]
2022.11.15	Pin6 Datagramm Beschreibung für StEin Internal eXtension

OBSOLETE, REMOVED DATAGRAMS

UDP ONLY: LOCO GUI EXTENDED [0X17.0X27], OBSOLETE, REPLACED BY 0X17.0X28

Dieser Befehl steht nur am PC Interface zur Verfügung (LAN).
Damit können die Fahrzeug Basis GUI Informationen ans System gesendet werden bzw. abgefragt werden.

Grp	Cmd	M	ID	DLC	DB 1 .. 2	DB 3 .. 4	DB 5 .. 6	DB 9 .. n
0x17	0x27	0b00		6	SrcID	NID	SubID	
0x17	0x27	0b01		12 ..	NID	SubID	Fahrzeug GUI Daten	
0x17	0x27	0b11		12 ..	NID	SubID	Fahrzeug GUI Daten	

In der Abfrage kann angegeben werden welcher GUI Teil/Version abgefragt wird.

Antwort auf Abfrage mit SubId=0x0000:

DB	Size	Inhalt
5 .. 6	2	Fahrzeug Gruppe
7 … 38	32	Fahrzeug Name (32 Zeichen 0x00 abgeschlossen)
39 … 40	2	Fahrzeug Bild Nummer
41 … 42	2	Fahrzeug Tacho Nummer
43 … 44	2	V. Max. Fwd
45 … 46	2	V. Max. Rev
47 … 48	2	V. Rangier
49 … 50	2	Antriebsart
51 … 52	2	Epoche
53 .. 54	2	Landescode
55 .. 181	128	Function Icons 0 …

Antwort auf Abfrage mit SubId=0x0100:

DB	Size	Inhalt
5 .. 6	2	Fahrzeug Gruppe
7 … 38	32	Fahrzeug Name (32 Zeichen 0x00 abgeschlossen)
39 … 40	2	Fahrzeug Bild, NId
41 … 44	4	Fahrzeug Bild, CRC32
45 … 46	2	Fahrzeug Tacho NId
47 … 50	4	Fahrzeug Tacho CRC32
51 … 52	2	V. Max. Fwd
53 … 54	2	V. Max. Rev
55 … 56	2	V. Rangier
57 … 58	2	Antriebsart
59 … 60	2	Epoche
61 … 62	2	Landescode
63 … 191	128	Function Icons 0 …

DER CAN BUS

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 machbare Entfernungen 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 Nachrichtenebene 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 Buspegel. Er erkennt dabei Differenzen zwischen gesendetem und empfangenen Bit. Dadurch können alle globalen Fehler und lokal am Sender auftretende Bitfehler sicher erkannt werden.

5. Bit-Stuffing

Auf der Bit-Ebene 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 Flags" 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. Lokgeschwindigkeit 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 durch irgendeinen Master 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 CAN-Bus 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 CAN-Bus sind in sogenannten „Schichten“ (Layer) definiert.

- 1. Bitübertragungsschicht (Physical Layer). Diese Schicht beschreibt die physikalischen Eigenschaften wie z.B. Stecker, Kabel, Signalpegel und die Darstellung eines Bits auf der Leitung.
- 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 bereitgestellt und mit einer Kennung versehen, die eine inhaltsbezogene Adressierung ermöglicht. Durch die Wahl der Kennung wird jede Nachricht mit einer festgelegten Priorität versehen.