API LoDi-Rektor

Einführung

Im folgenden werden die vom LoDi-Rektor verstandenen Kommandos aufgelistet. Die Kommunikation zum Gerät wird im Abschnitt Allgemeine API erklärt.

Basis-Kommandos

GetVersion

Dieses Kommando liefert die Gerätekennung sowie die FW-Version des Geräts.

Pakettyp	Kommando	Paketnummer
0x20	0x0F	0x00 .. 0xFF

Von Steuersoftware zu senden

Pakettyp	Kommando	Paketnummer	Gerätetyp	Major	Minor	Patch
0x21	0x0F	0x00 .. 0xFF	0x03			

Antwort vom Gerät

Der **Gerätetyp** ist beim LoDi-Rektor immer 0x03.

Die Firmware-Version setzt sich aus den drei Komponenten Major, Minor und Patch zusammen. Sie wird im Format "v<Major>.<Minor>.<Patch>" angezeigt. Beispiel: "v01.03.01"

Die **Major**-Version ändert sich nur, wenn eine komplett neue Hardware mit geänderten Eigenschaften und Funktionsumfang herausgebracht wird.

Die **Minor**-Version ändert sich, wenn Ergänzungen an der API erfolgen. Diese können bei einzelnen Kommandos auch inkompatibel sein.

Die **Patch**-Version ändert sich bei allgemeinen Fehlerbehebungen, die nicht die API betreffen.

CloseConnection

Dieses Kommando sollte vor der Verbindungstrennung gesendet werden.

Pakettyp	Kommando	Paketnummer
0x20	0x0C	0x00 .. 0xFF

Von Steuersoftware zu senden

Pakettyp	Kommando	Paketnummer
0x21	0x0C	0x00 .. 0xFF

Antwort vom Gerät ist ACK

DeviceConfigGet

Dieses Kommando liest die Einstellungen des LoDi-Rektors.

Pakettyp	Kommando	Paketnummer
0x20	0x99	0x00 .. 0xFF

Von Steuersoftware zu senden

Pakettyp	Kommando	Paketnummer	Fehlerreaktion	StartStop	CDE	Acc Offset	Default protocoll
0x21	0x99	0x00 .. 0xFF	0 .. 1	0 .. 1	0 .. 6	0 .. 1	0 .. 3

Antwort vom Gerät

Das Feld **Fehlerreaktion** definiert, was bei einem Fehler am CDE-Anschluss passiert.

- 0: Die Fehlerleitung am CDE-Ausgang wird nicht verändert
- 1: Der Fehler wird an den CDE-Ausgang durchgereicht. Dort angeschlossene Booster schalten ab.

Im Feld **StartStop** wird das Verhalten des beim Start/Stop am LoDi-Rektor festegelegt. Dies betrifft sowohl den Knopf als auch das Kommando über die API-Schnittstelle.

- 0: Die Zustände der Booster ändern sich nicht.
- 1: Die Booster werden mit dem Start/Stop-Signal geschaltet.

Der Eintrag **CDE** legt das Verhalten des CDE-Anschlusses fest.

- 0: Der CDE-Anschluss ist abgeschaltet. Er ist elektrisch ohne Funktion
- 1: Der CDE-Anschluss ist auf Ausgang geschaltet. Der Error-Eingang wird nicht ausgewertet.
- 2: Der CDE-Anschluss ist als Eingang geschaltet. Das Gleis-Signal kommt über den CDE-Anschluss von einer anderen Zentrale.
- 3: Der CDE-Anschluss ist als Ausgang geschaltet. Das Gleis-Signal wird vom LoDi-Rektor generiert. Es können weitere Booster am CDE-Anschluss angeschlossen werden.
- 6: Der CDE-Anschluss ist als Sniffer konfiguriert. Es wird noch DCC-Befehlen gelauscht. Diese werden interpretiert und im vom Rektor generierten Gleissignal weitergegeben. Dieser Modus eignet sich, um eine weitere Zentrale als Handregler anzuschießen.

Das Feld **"Default protocol"** definiert das Standardprotokoll welches auch generiert werden soll, wenn es noch keine Befehle gibt.

1 = DCC

2 = Motorola

3 = M3 / MFX®

SetWatchdog

Wurde der Watchdog-Befehl einmal empfangen, erwartet der LoDi-Rektor spätestens alle 2 Sekunden diesen Befehl erneut. Bleibt er aus, werden die Booster abgeschaltet.

Pakettyp	Kommando	Paketnummer	Status
0x20	0x9A	0x00 .. 0xFF	0 .. 1

Von Steuersoftware zu senden

Das Feld Status schaltet die Watchdos-Überwachung ein oder aus.

- 0: Die Überwachung wird abgeschaltet.
- 1: Die Überwachung wird eingeschaltet.

Pakettyp	Kommando	Paketnummer
0x21	0x9A	0x00 .. 0xFF

Antwort vom Gerät

GetModules

Dieses Kommando liest den gesamten µCon-Bus aus.

Pakettyp	Kommando	Paketnummer
0x20	0xA0	0x00 .. 0xFF

Von Steuersoftware zu senden

Pakettyp	Kommando	Paketnummer	Anzahl	Typ0	..	Typ63
0x21	0xA0	0x00 .. 0xFF	64			

Antwort vom Gerät

Die 64 Typ-Felder sind aufsteigend mit allen µCon-Adressen beginnend mit 0 gefüllt. Die Werte entsprechen den Typen der vorhandenen Geräte.

Aufbau Typ:

- Bit 7: Gerät wartet auf die Vergabe einer neuen LoDi-Bus Adresse.
- Bit 6: Gerät befindet sich im Bootloader. Es kann eine neue Firmware heruntergeladen werden.
- Bit 5..0: Gerätetyp:
- 0: Kein Gerät auf Adresse vorhanden
- 1: µCon-Booster
- 2: Railspeed
- 3: LoDi-Booster
- 4: CDE-Anschluss
- 5: LoDi-TrainSpeed
- 6: LoDi-Booster 10A
- 7: LoDi-Booster 10A+
- 8: LoDi-DC-Car-Booster
- 9: LoDi-OpenCar-Booster
- 10: LoDi-CV-Programmer

GetModulName

Liest den Namen eines µCon-Geräts am Bus. Die Namen werden anhand der µCon-Adresse im LoDi-Rektor gespeichert.

Pakettyp	Kommando	Paketnummer	µCon-Adresse
0x20	0xA3	0x00 .. 0xFF	1 .. 63

Von Steuersoftware zu senden

µCon-Adresse enthält die µCon-Adresse des zu lesenden Geräts.

Pakettyp	Kommando	Paketnummer	Länge	Zeichen 1	..	Zeichen 16
0x21	0xA3	0x00 .. 0xFF	0 .. 16			

Antwort vom Gerät

Im Feld **Länge** wird die Anzahl der Zeichen im Namen übergeben.

Die Felder **Zeichen 1** bis **Zeichen 16** enthalten die Zeichen des Gerätenamens.

GetModuleVersion

Liest die Firmware-Version eines µCon-Geräts am Bus. Unterstützt das Gerät dieses Kommando nicht, wird für MajorVersion und MinorVersion 0 zurückgegeben.

Pakettyp	Kommando	Paketnummer	µCon-Adresse
0x20	0x8F	0x00 .. 0xFF	1 .. 63

Von Steuersoftware zu senden

µCon-Adresse enthält die µCon-Adresse des zu lesenden Geräts.

Pakettyp	Kommando	Paketnummer	µCon-Adresse	MajorVersion	MinorVersion
0x21	0x8F	0x00 .. 0xFF	1 .. 63	0 .. 255	0 .. 255

Antwort vom Gerät

SetModuleName

Setzt den Namen eines µCon-Geräts. Die Namen werden anhand der µCon-Adresse im LoDi-Rektor gespeichert.

Pakettyp	Kommando	Paketnummer	Adresse	Länge	Zeichen 1	..	Zeichen 16
0x20	0xA2	0x00 .. 0xFF	1 .. 63	0 .. 16			

Von der Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des Geräts.

Das Feld **Länge** bestimmt die Länge des zu setzenden Strings und damit auch die Länge des Pakets.

Die Einträge **Zeichen 1** bis **Zeichen 16** enthalten die einzelnen Zeichen des Strings. Der String wird nicht 0-terminiert.

Booster-Kommandos

BoosterOn

Dieses Kommando schaltet einen oder mehrere Booster ein.

Pakettyp	Kommando	Paketnummer	Booster-Adresse	Status A	Status B
0x20	0x90	0x00 .. 0xFF	1..63, 255	0x00 .. 0x01	0x00 .. 0x01

Von Steuersoftware zu senden

Über die **Booster-Adresse** kann entweder ein einzelner Booster (1 bis 63) oder alle Booster (255) angesprochen werden.

Die beiden Booster-Kanäle A und B werden über **Status A** und **Status B** geschaltet:

- Status X = 0: Booster-Kanal aus
- Status X = 1: Booster-Kanal ein

Das Gerät antwortet mit einem **ACK-Paket**, wenn das Kommando ausgeführt wurde oder mit einem **NACK-Paket**, wenn z.B. die angegebene Booster-Nummer nicht vorhanden ist.

BoosterStatus

Dieses Kommando erfragt den Status eines oder aller Booster.

Pakettyp	Kommando	Paketnummer	Booster-Adresse	Neu
0x20	0x91	0,1 .. 0xFF	1..63, 255	0 .. 2

Von Steuersoftware zu senden

Über die **Booster-Adresse** kann entweder ein einzelner Booster (1 bis 63) oder alle Booster (255) angesprochen werden.

Das Feld **Neu** hat die folgenden Bedeutungen:

- 0: alle Booster
- 1: neue Booster

Pakettyp	Kommando	Paketnummer	Anzahl	Booster-Adresse	Status A	Status B
0x21	0x91	0x00 .. 0xFF		1..63	0 .. 2	0 .. 2

Antwort vom Gerät

Das Feld **Anzahl** gibt die Anzahl der gelesenen Booster zurück jeder Booster belegt im Paket jeweils drei Bytes.

Booster-Adresse entspricht der µCon-Adresse des gefundenen Boosters. **Status A** und **Status B** enthalten die Statusinformation der beiden Booster-Kanäle nach dem folgenden Muster:

- Status X = 0: Booster-Kanal aus
- Status X = 1: Booster-Kanal ein
- Status X = 2: Booster-Kanal Kurzschluss

BoosterDiagnostics

Dieses Kommando fragt Informationen zur Spannung, zu der Stromstärke beider Kanäle und zur Temperatur ab.

Pakettyp	Kommando	Paketnummer	Booster-Adresse
0x20	0x92	0x00 .. 0xFF	1..63, 255

Von Steuersoftware zu senden

Über die **Booster-Adresse** kann entweder ein einzelner Booster (1 bis 63) oder alle Booster (255) angesprochen werden.

Pakettyp	Kommando	Paketnummer	Anzahl	Booster-Adresse	Spannung	Strom A	Strom B	Temp.
0x21	0x92	0x00 .. 0xFF		1..63	0..255	0..255	0..255	0..255

Antwort vom Gerät

Das Feld **Anzahl** gibt die Anzahl der gelesenen Booster zurück jeder Booster belegt im Paket jeweils fünf Bytes.

Booster-Adresse entspricht der µCon-Adresse des gefundenen Boosters.

Das Feld **Spannung** enthält die am Booster gemessene Ausgangsspannung. Sie lässt sich nach der folgenden Formel ermitteln: Spannung*17.0/146.0.

Die Felder **Strom A** und **Strom B** enthalten den am jeweiligen Booster-Kanal gemessenen Strom. Für den LoDi-Booster wird der Strom nach der folgenden Formel ermittelt: Strom*2.5A/127.0.

Für den LoDi-Booster 10A erfolgt die Berechnung nach der folgenden Formel: Strom*50mA

Für den LoDi-Booster 20A erfolgt die Berechnung nach der folgenden Formel: Strom*100mA

Die **Temperatur** wird wie folgt ermittelt: Temp-82.0 in Grad Celsius.

GetBoosterConfig

Dieses Kommando liest die Einstellungen eines Boosters.

Pakettyp	Kommando	Paketnummer	Booster-Adresse
0x20	0x93	0x00 .. 0xFF	1 .. 63

Von Steuersoftware zu senden

Die **Booster-Adresse** gibt den zu lesenden Booster (1 bis 63) an.

Pakettyp	Kommando	Paketnummer	Booster-Adresse	Empfindlichkeit	Ausschaltzeit	Startmodus	Railcom	Max. Current
0x21	0x93	0x00 .. 0xFF	1 .. 63	10 .. 50	0 .. 15	0 .. 1	0 .. 1	0..255

Antwort vom Gerät

Die **Booster-Adresse** gibt den Booster an, dessen Einstellungen gelesen wurden.

Die **Empfindlichkeit** stellt die Zeitdauer ein, bis ein Kurzschluss ausgelöst wird. Die folgenden Einstellungen sind möglich:

- 10: 50ms "sehr schnell"
- 20: 100ms "schnell"
- 30: 300ms "normal"
- 40: 500ms "langsam"
- 50: 700ms "sehr langsam"

Die **Ausschaltzeit** gibt an, wie lange der Booster nach einem Kurzschluss ausgeschaltet bleiben soll, bevor er automatisch wieder einschaltet. Dieser Wert wird in Sekunden angegeben.

Mit dem Startmodus wird das Verhalten nach einem Kurzschluss eingestellt.

- 0: Nach einem Kurzschluss wird der Booster nach Ablauf der in Empfindlichkeit eingestellten Zeit wieder eingeschaltet.
- 1: Der Booster bleibt nach einem Kurzschluss ausgeschaltet.

Das Feld **Railcom** gibt an, ob der Booster einen Railcom-Cutout generiert.

- 0: Der Booster generiert keinen Railcom-Cutout.
- 1: Der Booster generiert einen Railcom-Cutout.

Der Eintrag **Max. Current** wird ab Firmware-Version 3.3.2 versendet. Er gibt den erlaubten Maximalstrom pro Kanal an. Der Maximalstrom ergibt sich wie folgt: Max. Current * 50mA. Ist Max Current 0, so greift das Maximum des Boosters.

Trainspeed und Railspeed-Kommandos

GetTrainspeedConfig

Liest die Einstellungen eines Trainspeed-Sensors.

Pakettyp	Kommando	Paketnummer	Adresse
0x20	0x9E	0x00 .. 0xFF	1 .. 63

Von Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des einzustellenden Trainspeed-Sensors.

Pakettyp	Kommando	Paketnummer	Adresse	ScaleHigh	ScaleLow	Sprache	LenAktiv	TolAktiv
0x21	0x9E	0x00 .. 0xFF	1 .. 63			0 .. 1	0 .. 1	0 .. 1

Von Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des einzustellenden Trainspeed und Railspeed-Sensors.

Die beiden Feldern **ScaleHigh** und **ScaleLow** den Maßstab 1:n für die Messung an. Wobei sich n wie folgt berechnet: n = (ScaleHigh*256 + ScaleLow) / 10.

Das Feld **Sprache** legt die Sprache für das Display des Railspeed-Sensors fest.

- 0: Deutsch
- 1: Englisch

Die Einstellung **LenAktiv** aktiviert die Längenmessung.

- 0: Längenmessung inaktiv
- 1: Längenmessung aktiv

Mit **TolAktiv** wird die Toleranzmessung aktiviert.

- 0: Die Toleranzmessung der Geschwindigkeit ist inaktiv
- 1: Die Toleranzmessung der Geschwindigkeit ist aktiv.

TrainspeedSetConfig

Setzt die Einstellungen eines angeschlossenen Trainspeed-Sensors.

Pakettyp	Kommando	Paketnummer	Adresse	ScaleHigh	ScaleLow	Sprache	LenAktiv	TolAktiv
0x20	0x9B	0x00 .. 0xFF	1 .. 63			0 .. 1	0 .. 1	0 .. 1

Von Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des einzustellenden Trainspeed-Sensors.

In den beiden Feldern **ScaleHigh** und **ScaleLow** wird der Maßstab 1:n für die Messung eingestellt. Wobei sich n wie folgt berechnet: n = (ScaleHigh*256 + ScaleLow) / 10. Für den Maßstab H0 (n=87) ist also ScaleHigh auf 3 und ScaleLow auf 102 zu setzen.

Das Feld **Sprache** legt die Sprache für das Display des Railspeed-Sensors fest.

- 0: Deutsch
- 1: Englisch

Die Einstellung **LenAktiv** aktiviert die Längenmessung.

- 0: Längenmessung inaktiv
- 1: Längenmessung aktiv

Mit **TolAktiv** wird die Toleranzmessung aktiviert.

- 0: Die Toleranzmessung der Geschwindigkeit ist inaktiv
- 1: Die Toleranzmessung der Geschwindigkeit ist aktiv.

Pakettyp	Kommando	Paketnummer
0x21	0x9B	0x00 .. 0xFF

Antwort vom Gerät

TrainspeedSetText

Ändert den im Railspeed-Sensor angezeigten Text.

Pakettyp	Kommando	Paketnummer	Adresse	Typ	Länge	Zeichen 1	..	Zeichen 16
0x20	0x9F	0x00 .. 0xFF	1 .. 63	0	0 .. 16			

Von der Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des Trainspeed-Sensors.

Das Feld **Typ** wird weiter nicht verwendet, es sollte 0 mitgesendet werden.

Das Feld **Länge** bestimmt die Länge des anzuzeigenden Strings und damit auch die Länge des Pakets.

Die Einträge **Zeichen 1** bis **Zeichen 16** enthalten die einzelnen Zeichen des Strings. Der String wird nicht 0-terminiert.

TrainspeedSetActive

Aktiviert und deaktiviert die Events von Trainspeed-Sensoren.

Pakettyp	Kommando	Paketnummer	Adresse	Aktiv
0x20	0x9C	0x00 .. 0xFF	1 .. 63, 255	0 .. 1

Von der Steuersoftware zu senden

Das Feld **Adresse** enthält die µCon-Adresse des zu aktivierenden Trainspeed-Sensors. Wird 255 angegeben, werden die Events aller am Bus vorhandenen Trainspeed-Sensoren aktiviert oder deaktiviert.

Der Eintrag **Aktiv** gibt an, ob die Events des Sensor aktiviert oder deaktiviert werden sollen.

- 0: Events deaktivieren
- 1: Events aktivieren

TrainspeedEvent

Dieser Event wird automatisch vom Gerät gesendet, wenn RailsTrainpeed-Events eingeschaltet sind.

Pakettyp	Kommando	Paketnummer	Adresse	Typ	WertHi	WertLo	Status	Valid	SensorL	SensorR
0x22	0x9D	0x00 .. 0xFF	1 .. 63	1, 2, 4			0 .. 1	0 .. 1	0 .. 1	0 .. 1

Event vom Gerät

Das Feld **Adresse** enthält die µCon-Adresse des auslösenden Trainspeed-Sensors.

Typ bestimmt, die Art des Events.

- 1: Geschwindigkeitsmessung
- 2: Längenmessung
- 3: Toleranz der Geschwindigkeit

Die Einträge **WertHi** und **WertLo** geben den Messwert an. Bei Geschwindigkeits- und Toleranzmessung ist die Einheit mm/s, bei Längenmessung mm. Sind WertHi und WertLo 0, so ist die Messung ungültig.

Der Messwert wird dabei wie folgt ermittelt: Wert = WertHi*256 + Wert Low.

Das Feld **Status** zeigt an, ob die Messung beendet ist.

- 0: Messung beendet, bereit für nächste Messung
- 1: Messung läuft noch

Valid gibt an, ob der in WertHi und WertLo angezeigt Wert gültig ist.

- 0: WertLo und WertHi sind ungültig.
- 1: WertLo und WertHi sind gültig.

Die Einträge **SensorL** und **SensorR** zeigen die aktuelle Verdeckung des linken und rechten Sensors an.

- 0: Sensor frei
- 1: Sensor verdeckt

Das Valid Bit, kann schon, bei einer noch nicht vollendeten Messung, einen gültigen Wert übermitteln, nämlich dann, wenn die Lok den zweiten Sensor erreicht. Dieser Wert kann auch verwendet werden, die Lok darf aber erst nach beenden des Status Feld angehalten und die Fahrtrichtung ggf. geändert werden.

Zentrale

Die Zentrale kann das DCC-, Motorola-, M3 / MFX®-Gleissignale erzeugen. Sie ist nur aktiv, wenn der CDE-Anschluss entweder abgeschaltet ist oder auf Ausgang steht.

Track-Commander

Die Track-Commands sind optional für den LoDi-Rektor und können zuerst gesetzt werden.

Alle Befehle werden quittiert (0x21), ohne zusätzlichen Daten zurückzugeben, es sei den, es ist anders definiert.

Type	Cmd	SeqNr
0x21	0xC0-0xCF	0-255

ActiveTrackProtocols

Dieser Befehl stellt die aktiven Gleisprotokolle ein.

Hinweis: Das mit dem Kommando TrackProtocols eingestellte Standardprotokoll wird zusätzlich zu den in diesem Kommando aktivierten Protokollen eingeschaltet.

Type	Cmd	SeqNr	ActiveProtocols
0x20	0xCA	0-255	Bit 7, 0..2

ActiveProtocols ist bitcodiert:

- Bit 0: DCC aktiv
- Bit 1: Motorola aktiv
- Bit 2: mfx aktiv
- Bit 7: Einstellung permanent speichern

Wird Bit 7 gesetzt, so wird die Protokollauswahl permanent gespeichert und steht auch nach dem nächsten Gerätestart wieder zur Verfügung.

Die Einstellung 0x85 aktiviert die Protokollauswahl DCC + mfx und speichert sie permanent.

Wird AktiveProtocols nicht angegeben, so wird nicht geändert.

Das Antwortpaket gibt immer die aktuell aktivierten Protokolle zurück.

TrackProtocols

Dieses Kommando wählt das Gleisprotokoll und startet Gleisgenerierung wenn nichts gewählt war.

Type	Cmd	SeqNr	Default protocol
0x20	0xCE	0-255	Bit 7, 3..0

Das Feld **"Default protocol"** definiert das Standardprotokoll welches auch generiert werden soll wenn es noch keine Befehle gibt.

1 = DCC

2 = Motorola

3 = M3 / MFX®

Wenn das höchste Bit im Feld (Bit7) gesetzt wird, wird der Wert dauerhaft/permanent gespeichert und verwendet auch nach Neustarten des Rektors (zB 0x81 für DCC).

RemoveProtocols

Dieses Kommando stoppt die Generation des Gleisprotokolls. Sollte verwendet werden wenn über CDE Eingang eine Externen Zentrale angeschlossen ist.

Type	Cmd	SeqNr
0x20	0xCF	0-255

Decoder commands

Alle Kommandos starten im selben Feld wie beschrieben und werden nicht wiederholt.

Type	Cmd	SeqNr	Protocol	AddrL	AddrH
0x20	0xC0-0xC6	0-255	Sub bit 7..4 / Main bit 3..0	bit 7..0	bit 15..8

Das Feld **"Protocol"** ist zweigeteilt. Im ersten Teil (Sub) wird das Protokoll, im zweiten (Main) die Details des Protokolls spezifiziert.

DCC, Main = 1;

Sub 1 =14 Stufen

Sub 2 = 27 Stufen (reserviert, aber nicht unterstützt)

Sub 3 = 28 Stufen

Sub 4 = 126 Stufen

Sub 5 = 126 Stufen Extended (neue Gleisbefehle u.a. F29+)

Motorola, Main = 2;

Sub 1 = 14 Stufen (offizielle Nummer der Fahrstufen).

Sub 2 = 27 Stufen wobei 2 Pakete hintereinander gesendet werden (27A) (reserviert, aber nicht unterstützt)

Sub 3 = 27 Stufen in einem Paket (27B) für neuer MFX Lokomotiven.

Sub 4 = 28 Stufen in einem Paket für ältere MFX Lokomotiven.

Sub 6 = Motorola Funktionsdecoder (alt)

M3 / MFX®, Main = 3

Kein Sub notwendig, deshalb 0.

Wenn der Sub-Part einmal gesendet wurde, ist es nicht unbedingt notwendig, diese immer zu senden. Wird eine 0 gesendet, so wird die Anzahl der Fahrstufen nicht geändert. Jeder andere Wert löst eine Änderung aus.

Die Lokadresse wird in die Bytes **AddrL** und AddrH angegeben. Der Aufbau hängt vom verwendeten Protokoll ab.

DCC

Kurze Adressen (1-127) werden in AddrL angegeben. AddrH muss null sein.

Lange Adressen (1-9999) werden in AddrL und AddrH angegben.

Motrola

Bei Motorola können Adressen (1-255) in AddrL gesetzt werden. AddrH muss 0 bleiben,

M3 / MFX®

Bei M3 / MFX® können Adressen (1-16383) in AddrL und AddrH spezifiziert werden.

LocoRelease

Wird verwendet, um eine Adresse aus dem Refresh zu nehmen, wenn die Lok nicht mehr auf dem Gleis steht.

Type	Cmd	SeqNr	Protocol	AddrL	AddrH
0x20	0xC0	0-255	Sub / Main	bit 7..0	bit 15..8

LocoSpeed

Setzt die Fahrstufe und die Richtung.

Type	Cmd	SeqNr	Protocol	AddrL	AddrH	Mask	Speed
0x20	0xC1	0-255	Sub / Main	bit 7..0	bit 15..8	bit 7..6	0-255

Das Feld **Mask** nutzt nur das aktuelle höhere Bit:

Bit 7, 1 = vorwärts, 0 = rückwärts

Bit 6, 1 = Nothalt (Speed muss 0 sein), 0 für normale Geschwindigkeitsregelung.

Das Feld **Speed** spezifiziert die Fahrstufe (0 = stop, 1 erste Stufe bis zum Maximum. Step ist abhängig vom gewählten Protokoll und dessen maximaler Fahrstufe.

Wird das Kommando ohne Parameter gesendet, so werden die vorhandenen Daten auslesen und das Paket als Antwort (mit 0x21) zurücksenden. Auch kann das Paket als Ereignis (0x22) gesendet werden, wenn ein anderer Teilnehmer die Einstellungen ändert.

LocoFunctions

Setzt die Funktionen die mit im Refresh sind.

Type	Cmd	SeqNr	Protocol	AddrL	AddrH	Index	Data
0x20	0xC2	0-255	Sub / Main	bit 7..0	bit 15..8	0-255	0..1

Das Feld **Index** spezifiziert die Funktionen die geändert werden:

Index 0-127 können individuell gesetzt werden (0 = f0, 1 = f1). Data muss 0 oder 1 sein.

Um die Funktionen effektiver zu setzen, können Funktionsgruppen im höheren Bit vom Feld **Index** (bit 7) gesetzt werden.

DCC (f0-f28)

Gruppe 1: **Index** = 0x81, **Data** = 0 0 0 f0 f4 f3 f2 f1
Gruppe 2: **Index** = 0x82, **Data** = 0 0 0 0 f8 f7 f6 f5
Gruppe 3: **Index** = 0x83, **Data** = 0 0 0 0 f12 f11 f10 f9
Gruppe 4: **Index** = 0x84, **Data** = f20 f19 f18 f17 f16 f15 f14 f13
Gruppe 5: **Index** = 0x85, **Data** = f28 f27 f26 f25 f24 f23 f22 f21
Nur für DCC 126 Extended gibt es noch zusätzliche Gruppe:
Gruppe 6: **Index** = 0x86, **Data** = f36 f35 f34 f33 f32 f31 f30 f29
Gruppe 7: **Index** = 0x87, **Data** = f44 f43 f42 f41 f40 f39 f38 f37
Gruppe 8: **Index** = 0x88, **Data** = f52 f51 f50 f49 f48 f47 f46 f45
Gruppe 9: **Index** = 0x89, **Data** = f60 f59 f58 f57 f56 f55 f54 f53
Gruppe 10: **Index** = 0x8a, **Data** = f68 f67 f66 f65 f64 f63 f62 f61
Diese Gruppen sind die gleichen original DCC Pakete die gesendet werden müssen.

Motorola (f0-f4)

Diese Funktionen können individuell gesetzt werden , da jede Funktion ein individuelles Paket an das Gleis sendet. Sollten Sie aber alle Funktionen gleichzeitig setzen wollen, dann machen sie folgendes:

Index = 0x81, **Data** = 0 0 0 f0 f4 f3 f2 f1

Für Motorola Funktionsdecoder ist es effizienter **Index** 0x81 mit f1-f4 zu setzen. Alle Funktionen werden auf einmal gesetzt.

M3 / MFX® (f0-f31)

Gruppe 1: **Index** = 0x81, **Data** = f7 f6 f5 f4 f3 f2 f1 f0
Gruppe 2: **Index** = 0x82, **Data** = f15 f14 f13 f12 f11 f10 f9 f8
Gruppe 3: **Index** = 0x83, **Data** = f23 f22 f21 f20 f19 f18 f17 f16
Gruppe 4: **Index** = 0x84, **Data** = f31 f30 f29 f28 f27 f26 f25 f24

Funktionen f16-f31 können individuell gesetzt werden, da für diese Funktionen doch ein individuelles Paket an das Gleis gesendet wird.

Das Kommando ohne Parameter wird die Daten auslesen und wird das beschrieben Paket als Antwort (mit 0x21) zurücksenden. Auch kann das Paket als Ereignis (0x22) gesendet werden, wenn andere Teilnehmer etwas ändern.

LocoBinary

Setzt die Binärfunktionen (nur in DCC möglich).

Type	Cmd	SeqNr	Protocol	AddrL	AddrH	Data	Extra
0x20	0xC3	0-255	Sub / Main	bit 7..0	bit 15..8	on/off / indexL	indexH

Das Feld **Extra** ist optional und wird nur für Binärfunktionen verwendet, die mit dem Index 128 beginnen.

Das Feld kann für denselben Effekt auch auf Null gesetzt werden.

AccessoryState

Um den State für alle Funktionsdecoder zu setzen.

Type	Cmd	SeqNr	Protocol	AddrL	AddrH	Data
0x20	0xC4	0-255	Sub / Main	bit 7..0	bit 15..8	basic / extended / aspect

Das Addressfeld für die Zubehörartikel spezifiziert die individuelle Adresse, nicht die Decoder Adresse für einen Decoder mit 4 doppelten Ausgängen. Die Adresse fängt wie in DCC mit 0 an.

Es gibt zwei Einstellungen. Es gibt einen Standard- und einen erweiterten Zubehörbefehl. Beim Standardbefehl gibt es zwei Zustände (Aspekt 0 und 1) und wird unterstützt von DCC und Motorola.
Beim erweiterten Befehl für DCC können 128 Aspekte gesetzt werden (Value 0-127).

Wenn für Protokoll Null verwendet wird, dann wird das Standardprotokoll verwendet. Wenn das M3/MFX® gewählt ist wird für schalten Motorola verwendet.

DCC
Abhängig von der Interpretation der Adressen sollten Sie der Adressen normalerweise eine 4 hinzufügen, um dem allgemeinen Standard zu entsprechen (RCN-213). Die Adressen 0-3 sind eine Art Schattenadressen, die nur von einigen Systemen (zB Roco) verwendet werden.

Wenn der höchste Bit im Feld **Data** (Bit7) gesetzt wird, werden Standardbefehle gesendet. In dem Fall gibt der niedrigste Bit des Feldes **Data** (Bit0) den Status vor und muss 0 oder 1 sein. Ob der Ausgang aktiviert ist oder nicht, wird durch den Bit3 indiziert. Normalerweise wird der Befehl zweimal gesendet, um es zu aktivieren und zu deaktivieren, da es normalerweise pulsiert.

Motorola
Unterstützt nur Basis Zubehör mit Bit 0 als Zustand (1 = Grün, 0 = Red) und Bit 3, um anzuzeigen, ob der Ausgang aktiviert ist oder nicht.

LocoCV / AccessoryCV

Um die Konfigurations-Variablen für Fahrzeuge (0xC5) und für Zubehör (0xC6) zu setzen.

Type	Cmd	SeqNr	Protocol	AddL	AddrH	Mask	IndexL	IndexH	Data
0x20	0xC5- 0xC6	0-255	Sub / Main	bit 7..0	bit 15..8	bit 7, 3..0	bit 7..0	bit 15..8	0- 255

Neben dem Protokoll und der Adresse, die bereits für Fahrzeuge und Zubehör beschrieben wurden, gibt es drei zusätzliche Felder.

Das Feld **Mask** enthält die **Aktion** im untersten Nibble (Bit 3..0):
0 = Write Byte, das **Daten** Feld enthält den Byte-Wert
1 = Write Bit, das Feld **Data** hat den Bit-Wert im höheren Bit (Bit 7) und im Index der anderen Bits.
2 = Read Byte, das **Data** Feld wird ignoriert und sollte auf Zero gesetzt werden.

Bei Zubehör entscheidet Bit 7 des Feldes **Mask**, ob Basic (1) oder Extended (0) verwendet wird.

Momentan werden Konfigurationsvariablen nur für DCC mit Indizes von 0-1023 unterstützt.

ProgTrackCv

Dieser Befehl ist nur ausführbar, wenn ein **LoDi-CV-Programmer** (im folgenden Programmer genannt) am LoDi-Bus angeschlossen ist. Ist kein Programmer angeschlossen, so wird mit **NACK** geantwortet. Da Operationen auf dem Programmiergleis länger dauern, kann es sein, dass bereits ein Befehl läuft. In diesem Fall wird **BUSY** zurückgemeldet. Wurde der Befehl vom Programmer angenommen, so wird **ACK** zurückgegeben.

Bei Leseoperationen erfolgt die Antwort über den ProgResponseEvent.
Schreiboperationen erzeugen keinen Event.

Type	Cmd	SeqNr	Protocol	CV_H	CV_L	Wert
0x20	0xC8	0-255	1	Operation, bit 8..9	bit 7..0	0..255

Das zu bearbeitende CV-Register wird in den Feldern CV_L und CV_H angegeben. CV_L nimmt dabei die unteren 8 Bit auf und CV_H die oberen Bits.

Alle CV Register (1..1024) werden direkt angesprochen. Von der CV-Adresse ist 1 abzuziehen. Daraus ergibt sich die Adressierung von 0 bis 1023, die den CV-Adressen 1 bis 1024 entspricht.

Bei Schreiboperationen ist Bit 7 in CV_H gesetzt.

Das Feld **Wert** ist optional und wird nur bei Schreibbefehlen benötigt.

ProgTrackCvEvent

Dieser Event dient als Rückmeldung auf CV-Befehle, die an eine Lok oder einen Zubehörartikel auf dem Programmiergleis gesendet wurden.

Type	Cmd	SeqNr	Protocol	CV_H	CV_L	Wert
0x22	0xC8	0-255	1	bit 8..9	bit 7..0	0..255

Event vom Gerät

CV_H und **CV_L**: siehe ProgTrackCV (nur lesen)

Wert: Der Inhalt des angesprochenen CVs oder Registers.

LocoID

Type	Cmd	SeqNr	Protocol	AddrL	AddrH	ID1	ID2	ID3	ID4
0x20	0xCC	0-255	3	bit 7..0	bit 15..8	bit 7..0	bit 15..8	bit 16..23	bit 31..24

Zum setzte der ID muss die UID des Decoders bekannt sein.
Die UID wird spezifiziert in ID1-4.

Lokstoredigital e.K.

Andreas Hornung

Stäffelsbergstrasse 13

76889 Dörrenbach

info@lokstoredigital.de