Z21 LAN Protokoll Spezifikation

Rechtliches, Haftungsausschluss

Die Firma Modelleisenbahn 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 die Modelleisenbahn GmbH, Plainbachstraße 4, A-5101 Bergheim, Austria, ausdrücklich keinerlei Haftung.

Die Modelleisenbahn GmbH, Plainbachstraße 4, A-5101 Bergheim, Austria, 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 Modelleisenbahn GmbH, Plainbachstraße 4, A-5101 Bergheim, Austria, 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.

Impressum

Herausgeber: Modelleisenbahn GmbH, Plainbachstraße 4, A-5101 Bergheim, Austria

Änderungshistorie

Datum	Dokumentenversion	Änderung
06.02.2013	1.00	Beschreibung der LAN Schnittstelle für Z21 FW Version 1.10, 1.11 und SmartRail FW Version 1.12
20.03.2013	1.01	Z21 FW Version 1.20 LAN_SET_BROADCASTFLAGS: neue Flags LAN_GET_HWINFO: neuer Befehl LAN_SET_TURNOUTMODE: MM-Format LocoNet: Gateway Funktionalität SmartRail FW Version 1.13 LAN_GET_HWINFO: neuer Befehl
29.10.2013	1.02	**Z21 FW Version 1.22:** **Decoder CV Lesen und Schreiben** POM Lesen und Accessory Decoder: neue Befehle **LocoNet Dispatch und Gleisbesetztmelder** *LAN_LOCONET_DISPATCH_ADDR*: neu Antwort LAN_SET_BROADCASTFLAGS: neues Flag LAN_LOCONET_DETECTOR: neuer Befehl
12.02.2014	1.03	**Z21 FW Version 1.23** Korrektur lange Fahrzeugadresse in Kapitel 4 Fahren LAN_X_MM_WRITE_BYTE LAN_LOCONET_DETECTOR: Erweiterung für LISSY
25.03.2014	1.04	**Z21 FW Version 1.24** LAN_SET_BROADCASTFLAGS: Flag 0x00010000 Kapitel 5 Schalten: Erklärung Weichenadressierung LAN_X_GET_TURNOUT_INFO: Erweiterung Queue-Bit LAN_X_DCC_WRITE_REGISTER
21.01.2015	1.05	**Z21 FW Version 1.25 und 1.26** Kapitel 4 Fahren: Erklärungen Fahrstufen und Format LAN_X_DCC_READ_REGISTER LAN_X_DCC_WRITE_REGISTER LAN_LOCONET_Z21_TX Binary State Control Instruction
05.04.2016	1.06	**Z21 FW Version 1.28** Kapitel 2 System Status Versionen: z21start LAN_GET_HW_INFO LAN_GET_CODE
19.04.2017	1.07	**Z21 FW Version 1.29 und 1.30** Kapitel 8 RailCom Kapitel 10 CAN: Belegtmelder
15.01.2018	1.08	Kapitel 9 LocoNet: Lissy Beispiele
23.05.2019	1.09	Kapitel 4 Fahren: Codierung der Geschwindigkeitsstufen Kapitel 7 R-BUS: 10808 und 10819 hinzugefügt Kapitel 9.3.1: Korrektur Binary State Control Instruction
28.01.2021	1.10	**Z21 FW Version 1.40** Kapitel 2 LAN_GET_HWINFO: weitere HW-Typen Kapitel 5 Schalten: Erweiterte Zubehördecoder DCCext Kapitel 11 zLink
11.08.2021	1.11	**Z21 FW Version 1.41** Kapitel 10 CAN: Booster
28.02.2022	1.12	**Z21 FW Version 1.42** Kapitel 2.18 SystemState: cseRCN213, Capabilities Kapitel 4: DCC Funktionen ≥ F29, Binary States Kapitel 6: Tippfehler POM Read „111001MM“ 0xE4 ausgebessert Kapitel 10.2 und 11.2: Booster Management
20.06.2023	1.13	**Z21 FW Version 1.43** Kapitel 4 Fahren: Motorola-Bit in LAN_X_LOCO_INFO Kapitel 4 Fahren: neue Befehle für Purge und E-STOP Kapitel 12 Modellzeit

Inhaltsverzeichnis

1 Grundlagen

1.1 Kommunikation

Die Kommunikation mit der Z21 erfolgt per UDP über die Ports 21105 oder 21106.
Steuerungsanwendungen am Client (PC, App, ...) sollten in erster Linie den Port 21105 verwenden.

Die Kommunikation erfolgt immer asynchron, d.h. zwischen einer Anforderung und der entsprechenden Antwort können z.B. Broadcast-Meldungen auftreten.

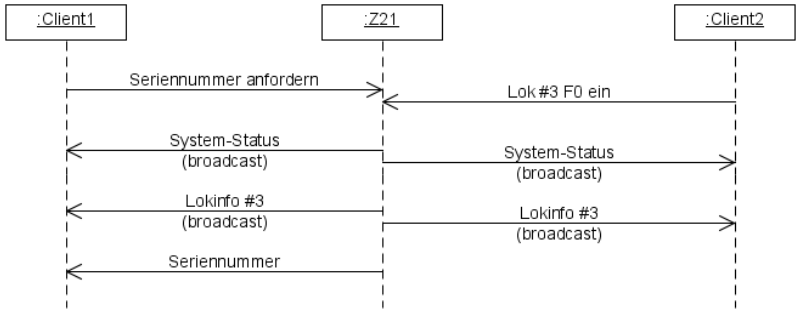

Abbildung 1 Beispiel Sequenz Kommunikation

Es wird erwartet, dass jeder Client einmal pro Minute mit der Z21 kommuniziert, da er sonst aus der Liste der aktiven Teilnehmer entfernt wird. Wenn möglich sollte sich ein Client beim Beenden mit dem Befehl LAN_LOGOFF bei der Zentrale abmelden.

1.2 Z21 Datensatz

1.2.1 Aufbau

Ein Z21-Datensatz, d.h. eine Anforderung oder Antwort, ist folgendermaßen aufgebaut:

DataLen (2 Byte)	Header (2 Byte)	Data (n Bytes)

- **DataLen** (little endian):
  Gesamtlänge über den ganzen Datensatz inklusive DataLen, Header und Data, d.h. DataLen = 2+2+n.
- **Header** (little endian):
  Beschreibt das Kommando bzw. die Protokollgruppe.
- **Data**:
  Aufbau und Anzahl hängen von Kommando ab. Genaue Beschreibung siehe jeweiliges Kommando.

Falls nicht anders angegeben, ist die Byte-Reihenfolge Little-Endian, d.h. zuerst das low byte, danach das high byte.

1.2.2 X-BUS Protokoll Tunnelung

Mit dem Z21-LAN-Header **0x40** (***LAN_X_xxx***) werden Anforderungen und Antworten übertragen, welche an das X-BUS-Protokoll *angelehnt* sind. Gemeint ist dabei nur das Protokoll, denn diese Befehle haben nichts mit dem physikalischen X-BUS der Z21 zu tun, sondern sind ausschließlich an die LAN-Clients bzw. die Z21 gerichtet.

Der eigentliche X-BUS-Befehl liegt dann im Feld **Data** innerhalb des Z21-Datensatzes. Das letzte Byte ist eine Prüfsumme und wird als XOR über den X-BUS-Befehl berechnet. Beispiel:

DataLen		**Header**		**Data**			
				X-Header	**DB0**	**DB1**	**XOR-Byte**
0x08	0x00	**0x40**	0x00	**h**	**x**	**y**	h XOR x XOR y

1.2.3 LocoNet Tunnelung

Ab Z21 FW Version 1.20.

Mit dem Z21-LAN-Header **0xA0 und 0xA1** (***LAN_LOCONET_Z21_RX, LAN_LOCONET_Z21_TX***) werden Meldungen, die von der Z21 am LocoNet-Bus empfangen bzw. gesendet werden, an den LAN-Client weitergeleitet. Der LAN-Client muss dazu die LocoNet-Meldungen mittels ***2.16*** LAN_SET_BROADCASTFLAGS abonniert haben.

Über den Z21-LAN-Header **0xA2 (LAN_LOCONET_FROM_LAN**) kann der LAN-Client Meldungen auf den LocoNet-Bus schreiben.

Damit kann die Z21 als **Ethernet/LocoNet Gateway** verwendet werden, wobei die Z21 gleichzeitig der LocoNet-Master ist, welcher die Refresh-Slots verwaltet und die DCC-Pakete generiert.

Die eigentliche LocoNet-Meldung liegt jeweils im Feld **Data** innerhalb des Z21-Datensatzes.

Beispiel LocoNet-Meldung OPC_MOVE_SLOTS <0><0> („DISPATCH_GET“) wurde von Z21 empfangen:

DataLen		**Header**		**Data**			
				OPC	**ARG1**	**ARG2**	**CKSUM**
0x08	0x00	**0xA0**	0x00	**0xBA**	**0x00**	**0x00**	**0x45**

Mehr zum Thema LocoNet-Gateway finden Sie im Abschnitt ***9*** LocoNet.

1.3 Kombinieren von Datensätzen in einem UDP-Paket

In den Nutzdaten eines UDP-Paket können auch mehrere, von einander unabhängige Z21-Datensätze gemeinsam an einen Empfänger gesendet werden. Jeder Empfänger muss diese kombinierten UDP-Pakete interpretieren können.

Beispiel

Folgendes kombinierte UDP Paket...

UDP Paket				
IP Header	**UDP Header**	**UDP Nutzdaten**		
		Z21 Datensatz 1	**Z21 Datensatz 2**	**Z21 Datensatz 3**
		LAN_X_GET_TOURNOUT_INFO #4	LAN_X_GET_TOURNOUT_INFO #5	LAN_RMBUS_GETDATA #0

... ist gleichwertig mit diesen drei hintereinander gesendeten UDP-Paketen:

UDP Paket 1		
IP Header	**UDP Header**	**UDP Nutzdaten**
		Z21 Datensatz
		LAN_X_GET_TOURNOUT_INFO #4

UDP Paket 2		
IP Header	**UDP Header**	**UDP Nutzdaten**
		Z21 Datensatz
		LAN_X_GET_TOURNOUT_INFO #5

UDP Paket 3		
IP Header	**UDP Header**	**UDP Nutzdaten**
		Z21 Datensatz
		LAN_RMBUS_GETDATA #0

Das UDP Paket muss in eine Ethernet MTU passen, d.h. es stehen abzüglich IPv4 Header und UDP-Header maximal 1500-20-8 = 1472 Bytes Nutzdaten übrig.

2 System, Status, Versionen

2.1 LAN_GET_SERIAL_NUMBER

Auslesen der Seriennummer der Z21.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x04	0x00	**0x10**	0x00	-

Antwort von Z21:

DataLen		**Header**		**Data**
0x08	0x00	**0x10**	0x00	Seriennummer 32 Bit (little endian)

2.2 LAN_LOGOFF

Abmelden des Clients von der Z21.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x04	0x00	**0x30**	0x00	-

Antwort von Z21:
keine

Verwenden Sie beim Abmelden die gleiche Portnummer wie beim Anmelden.

Anmerkung: das Anmelden erfolgt implizit mit dem ersten Befehl des Clients (z.B. *LAN_SYSTEM_STATE_GETDATA*, ...).

2.3 LAN_X_GET_VERSION

Mit folgendem Kommando kann die X-Bus Version der Z21 ausgelesen werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x21**	**0x21**	0x00

Antwort von Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	0x63	0x21	**XBUS_VER**	**CMDST_ID**	0x60

XBUS_VER X-Bus Protokoll Version (0x30 = V3.0, 0x36 = V3.6, 0x40 = V4.0, …)
CMDST_ID Command station ID (0x12 = Z21 Gerätefamilie)

2.4 LAN_X_GET_STATUS

Mit diesem Kommando kann der Zentralenstatus angefordert werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x21**	**0x24**	0x05

Antwort von Z21:
siehe *2.12* LAN_X_STATUS_CHANGED

Dieser Zentralenstatus ist identisch mit dem CentralState, welcher im SystemStatus geliefert wird, siehe *2.18* LAN_SYSTEMSTATE_DATACHANGED.

2.5 LAN_X_SET_TRACK_POWER_OFF

Mit diesem Kommando wird die Gleisspannung abgeschaltet.

Anforderung an Z21:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x21**	**0x80**	0xa1

Antwort von Z21:
siehe *2.7* LAN_X_BC_TRACK_POWER_OFF

2.6 LAN_X_SET_TRACK_POWER_ON

Mit diesem Kommando wird die Gleisspannung eingeschaltet, bzw. der Notstop oder Programmiermodus beendet.

Anforderung an Z21:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x21**	**0x81**	0xa0

Antwort von Z21:
siehe *2.8* LAN_X_BC_TRACK_POWER_ON

2.7 LAN_X_BC_TRACK_POWER_OFF

Folgendes Paket wird von der Z21 an die registrierten Clients versendet, wenn
- ein Client den Befehl *2.5* LAN_X_SET_TRACK_POWER_OFF abgeschickt hat
- durch ein anderes Eingabegerät (multiMaus) die Gleisspannung abgeschaltet worden ist.
- der betreffende Client den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x00**	0x61

2.8 LAN_X_BC_TRACK_POWER_ON

Folgendes Paket wird von der Z21 an die registrierten Clients versendet, wenn
- ein Client den Befehl *2.6* LAN_X_SET_TRACK_POWER_ON abgeschickt hat.
- durch ein anderes Eingabegerät (multiMaus) die Gleisspannung eingeschaltet worden ist.
- der betreffende Client den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x01**	0x60

2.9 LAN_X_BC_PROGRAMMING_MODE

Folgendes Paket wird von der Z21 an die registrierten Clients versendet, wenn die Z21 durch ***6.1*** LAN_X_CV_READ oder ***6.2*** LAN_X_CV_WRITE in den CV-Programmiermodus versetzt worden ist und der betreffende Client den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x02**	0x63

2.10 LAN_X_BC_TRACK_SHORT_CIRCUIT

Folgendes Paket wird von der Z21 an die registrierten Clients versendet, wenn ein Kurzschluss aufgetreten ist und der betreffende Client den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x08**	0x69

2.11 LAN_X_UNKNOWN_COMMAND

Folgendes Paket wird von der Z21 an den Client als Antwort auf eine ungültige Anforderung versendet.

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x82**	E3

2.12 LAN_X_STATUS_CHANGED

Folgendes Paket wird von der Z21 an den Client versendet, wenn der Client den Status explizit mit *2.4* LAN_X_GET_STATUS angefordert hat.

Z21 an Client:

DataLen		**Header**		**Data**			
				X-Header	**DB0**	**DB1**	**XOR-Byte**
0x08	0x00	**0x40**	0x00	**0x62**	**0x22**	**Status**	XOR-Byte

DB1 … Zentralenstatus

Bitmasken für Zentralenstatus:

```
#define csEmergencyStop          0x01   // Der Nothalt ist eingeschaltet
#define csTrackVoltageOff        0x02   // Die Gleisspannung ist abgeschaltet
#define csShortCircuit           0x04   // Kurzschluss
#define csProgrammingModeActive  0x20   // Der Programmiermodus ist aktiv
```

Dieser Zentralenstatus ist identisch mit dem SystemState.CentralState, siehe *2.18* LAN_SYSTEMSTATE_DATACHANGED.

2.13 LAN_X_SET_STOP

Mit diesem Kommando wird der Notstop aktiviert, d.h. die Loks werden angehalten aber die Gleisspannung bleibt eingeschaltet.

Anforderung an Z21:

DataLen		**Header**		**Data**	
				X-Header	**XOR-Byte**
0x06	0x00	**0x40**	0x00	**0x80**	0x80

Antwort von Z21:
siehe *2.14* LAN_X_BC_STOPPED

2.14 LAN_X_BC_STOPPED

Folgendes Paket wird von der Z21 an die registrierten Clients versendet, wenn

- ein Client den Befehl *2.13* LAN_X_SET_STOP abgeschickt hat.
- durch ein anderes Eingabegerät (multiMaus) der Notstop ausgelöst worden ist.
- der betreffende Client den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x81**	0x00	0x81

2.15 LAN_X_GET_FIRMWARE_VERSION

Mit diesem Kommando kann die Firmware-Version der Z21 ausgelesen werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0xF1**	0x0A	0xFB

Antwort von Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	0xF3	0x0A	**V_MSB**	**V_LSB**	XOR-Byte

DB1 … Höherwertiges Byte der Firmware Version
DB2 … Niederwertiges Byte der Firmware Version

Die Version wird im BCD-Format angegeben.
Beispiel:
`0x09 0x00 0x40 0x00 0xf3 0x0a` **`0x01 0x23`** `0xdb` bedeutet: „Firmware Version **1.23**“

2.16 LAN_SET_BROADCASTFLAGS

Setzen der Broadcast-Flags in der Z21. Diese Flags werden pro Client (d.h. pro IP + Portnummer) eingestellt und müssen beim nächsten Anmelden wieder neu gesetzt werden.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x08	0x00	**0x50**	0x00	Broadcast-Flags 32 Bit (little endian)

Broadcast-Flags ist eine OR-Verknüpfung der folgenden Werte:

0x00000001 Automatisch generierte Broadcasts und Meldungen, die das Fahren und Schalten betreffen, werden an den registrierten Client zugestellt.
Folgende Meldungen werden hier abonniert:
2.7 LAN_X_BC_TRACK_POWER_OFF
2.8 LAN_X_BC_TRACK_POWER_ON
2.9 LAN_X_BC_PROGRAMMING_MODE
2.10 LAN_X_BC_TRACK_SHORT_CIRCUIT
2.14 LAN_X_BC_STOPPED
4.4 LAN_X_LOCO_INFO (die betreffende Lok-Adresse muss ebenfalls abonniert sein)
5.3 LAN_X_TURNOUT_INFO

0x00000002 Änderungen der Rückmelder am R-Bus werden automatisch gesendet.
Broadcast Meldung der Z21 siehe ***7.1*** LAN_RMBUS_DATACHANGED

0x00000004 Änderungen bei RailCom-Daten der abonnierten Loks werden automatisch gesendet.
Broadcast Meldung der Z21 siehe **8.1** LAN_RAILCOM_DATACHANGED

0x00000100 Änderungen des Z21-Systemzustands werden automatisch gesendet.
Broadcast Meldung der Z21 siehe ***2.18*** LAN_SYSTEMSTATE_DATACHANGED

Ab Z21 FW Version 1.20:

0x00010000 Ergänzt Flag 0x00000001; Client bekommt nun LAN_X_LOCO_INFO, ohne vorher die entsprechenden Lok-Adressen abonnieren zu müssen, d.h. für alle gesteuerten Loks!
Dieses Flag darf aufgrund des hohen Netzwerkverkehrs nur von vollwertigen PC-Steuerungen verwendet werden und ist keinesfalls für mobile Handregler gedacht.
Ab FW V1.20 bis V1.23: LAN_X_LOCO_INFO wird für **alle** Loks versendet.
Ab **FW V1.24**: LAN_X_LOCO_INFO wird für **alle geänderten** Loks versendet.

0x01000000 Meldungen vom **LocoNet**-Bus an LAN Client weiterleiten ohne Loks und Weichen.

0x02000000 Lok-spezifische **LocoNet**-Meldungen an LAN Client weiterleiten:
OPC_LOCO_SPD, OPC_LOCO_DIRF, OPC_LOCO_SND, OPC_LOCO_F912, OPC_EXP_CMD

0x04000000 Weichen-spezifische **LocoNet**-Meldungen an LAN Client weiterleiten:
OPC_SW_REQ, OPC_SW_REP, OPC_SW_ACK, OPC_SW_STATE

Siehe auch Kapitel ***9*** LocoNet.

Ab Z21 FW Version 1.22:

0x08000000 Status-Meldungen von Gleisbesetztmeldern am LocoNet-Bus an LAN Client senden.
Siehe **9.5** LAN_LOCONET_DETECTOR

Ab Z21 FW Version 1.29:

0x00040000 Änderungen bei RailCom-Daten werden automatisch gesendet.
Client bekommt LAN_RAILCOM_DATACHANGED, auch ohne vorher die entsprechenden Lok-Adressen abonnieren zu müssen, d.h. für alle gesteuerten Loks!
Dieses Flag darf aufgrund des hohen Netzwerkverkehrs nur von vollwertigen PC-Steuerungen verwendet werden und ist keinesfalls für mobile Handregler gedacht.
Broadcast Meldung der Z21 siehe **8.1** LAN_RAILCOM_DATACHANGED

Ab Z21 FW Version 1.30:
0x00080000 Status-Meldungen von Gleisbesetztmeldern am CAN-Bus an LAN Client senden.
Siehe **10.1** LAN_CAN_DETECTOR

Ab Z21 FW Version 1.41:
0x00020000 CAN-Bus Booster Status-Meldungen an LAN Client weiterleiten.
Siehe **10.2.3** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD

Ab Z21 FW Version 1.43:
0x00000010 Fastclock Modellzeit Meldungen an LAN Client senden.
Siehe **12.2** LAN_FAST_CLOCK_DATA

Antwort von Z21:
keine

Berücksichtigen Sie bei den Einstellungen zu den Broadcast-Flags auch die Auswirkungen auf die Netzwerkauslastung. Dies gilt vor allem für die Broadcast-Flags 0x00010000, 0x00040000, 0x02000000 und 0x04000000! Die IP-Pakete dürfen vom Router bei Überlast gelöscht werden und UDP bietet keine hierfür keine Erkennungsmechanismen! Beispielsweise bei Flag 0x00000100 (Systemzustand) ist es überlegenswert, ob nicht 0x00000001 mit den entsprechenden *LAN_X_BC_xxx*-Broadcast-Meldungen eine sinnvollere Alternative darstellt. Denn nicht jede Anwendung muss jederzeit bis ins Detail über die aktuellsten Spannungs-, Strom- und Temperaturwerte der Zentrale informiert sein.

2.17 LAN_GET_BROADCASTFLAGS

Auslesen der Broadcast-Flags in der Z21.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x04	0x00	**0x51**	0x00	-

Antwort von Z21:

DataLen		**Header**		**Data**
0x08	0x00	**0x51**	0x00	Broadcast-Flags 32 Bit (little endian)

Broadcast-Flags siehe oben.

2.18 LAN_SYSTEMSTATE_DATACHANGED

Änderung des Systemzustandes von der Z21 an den Client melden.

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000100
- den Systemzustand explizit angefordert hat, siehe unten *2.19* LAN_SYSTEMSTATE_GETDATA.

Z21 an Client:

DataLen		**Header**		**Data**
0x14	0x00	**0x84**	0x00	**SystemState** (16 Bytes)

SystemState ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	**Typ**	**Name**		
0	INT16	MainCurrent	mA	Strom am Hauptgleis
2	INT16	ProgCurrent	mA	Strom am Programmiergleis
4	INT16	FilteredMainCurrent	mA	geglätteter Strom am Hauptgleis
6	INT16	Temperature	°C	interne Temperatur in der Zentrale
8	UINT16	SupplyVoltage	mV	Versorgungsspannung
10	UINT16	VCCVoltage	mV	interne Spannung, identisch mit Gleisspannung
12	UINT8	CentralState	bitmask	siehe unten
13	UINT8	CentralStateEx	bitmask	siehe unten
14	UINT8	reserved		
15	UINT8	Capabilities	bitmask	siehe unten, ab Z21 Version V1.42

Bitmasken für CentralState:

```
#define csEmergencyStop          0x01   // Der Nothalt ist eingeschaltet
#define csTrackVoltageOff        0x02   // Die Gleisspannung ist abgeschaltet
#define csShortCircuit           0x04   // Kurzschluss
#define csProgrammingModeActive  0x20   // Der Programmiermodus ist aktiv
```

Bitmasken für CentralStateEx:

```
#define cseHighTemperature       0x01   // zu hohe Temperatur
#define csePowerLost             0x02   // zu geringe Eingangsspannung
#define cseShortCircuitExternal  0x04   // am externen Booster-Ausgang
#define cseShortCircuitInternal  0x08   // am Hauptgleis oder Programmiergleis
```

Ab Z21 FW Version 1.42:

```
#define cseRCN213                0x20   // Weichenadressierung gem. RCN213
```

Ab Z21 FW Version 1.42:

Bitmasken für Capabilities:

```
#define capDCC                   0x01   // beherrscht DCC
#define capMM                    0x02   // beherrscht MM
//#define capReserved            0x04   // reserviert für zukünftige Erweiterungen
#define capRailCom               0x08   // RailCom ist aktiviert
#define capLocoCmds              0x10   // akzeptiert LAN-Befehle für Lokdecoder
#define capAccessoryCmds         0x20   // akzeptiert LAN-Befehle für Zubehördecoder
#define capDetectorCmds          0x40   // akzeptiert LAN-Befehle für Belegtmelder
#define capNeedsUnlockCode       0x80   // benötigt Freischaltcode (z21start)
```

SystemState.Capabilities verschafft dem LAN-Client einen Überblick über Feature-Umfang des Geräts. Falls SystemState.Capabilities == 0 ist, dann kann man davon ausgehen, dass es sich um eine ältere Firmwareversion handelt. Bei älteren Firmware-Versionen sollte SystemState.Capabilities nicht ausgewertet werden!

2.19 LAN_SYSTEMSTATE_GETDATA

Anfordern des aktuellen Systemzustandes.

Anforderung an Z21:

DataLen		Header		Data
0x04	0x00	**0x85**	0x00	-

Antwort von Z21:
Siehe oben *2.18* LAN_SYSTEMSTATE_DATACHANGED

2.20 LAN_GET_HWINFO

Ab Z21 FW Version 1.20 und SmartRail FW Version V1.13.

Mit diesem Kommando kann der Hardware-Typ und die Firmware-Version der Z21 ausgelesen werden.

Anforderung an Z21:

DataLen		Header		Data
0x04	0x00	**0x1A**	0x00	-

Antwort von Z21:

DataLen		Header		Data	
0x0C	0x00	**0x1A**	0x00	HwType 32 Bit (little endian)	FW Version 32 Bit (little endian)

HwType:

```
#define D_HWT_Z21_OLD      0x00000200   // „schwarze Z21” (Hardware-Variante ab 2012)
#define D_HWT_Z21_NEW      0x00000201   // „schwarze Z21”(Hardware-Variante ab 2013)
#define D_HWT_SMARTRAIL    0x00000202   // SmartRail (ab 2012)
#define D_HWT_z21_SMALL    0x00000203   // „weiße z21” Starterset-Variante (ab 2013)
#define D_HWT_z21_START    0x00000204   // „z21 start” Starterset-Variante (ab 2016)

#define D_HWT_SINGLE_BOOSTER     0x00000205   // 10806 „Z21 Single Booster” (zLink)
#define D_HWT_DUAL_BOOSTER       0x00000206   // 10807 „Z21 Dual Booster” (zLink)

#define D_HWT_Z21_XL       0x00000211   // 10870 „Z21 XL Series”  (ab 2020)
#define D_HWT_XL_BOOSTER   0x00000212   // 10869 „Z21 XL Booster” (ab 2021, zLink)

#define D_HWT_Z21_SWITCH_DECODER 0x00000301   // 10836 „Z21 SwitchDecoder” (zLink)
#define D_HWT_Z21_SIGNAL_DECODER 0x00000302   // 10836 „Z21 SignalDecoder” (zLink)
```

Die **FW Version** wird im BCD-Format angegeben.

Beispiel:
0x0C 0x00 0x1A 0x00 0x00 0x02 0x00 0x00 0x20 0x01 0x00 0x00
bedeutet: „Hardware Typ **0x200,** Firmware Version **1.20**“

Um die Version einer älteren Firmware auszulesen, verwenden Sie alternativ den Befehl ***2.15*** LAN_X_GET_FIRMWARE_VERSION. Für ältere Firmwareversionen gilt dabei:

- V1.10 ... Z21 (Hardware-Variante ab 2012)
- V1.11 ... Z21 (Hardware-Variante ab 2012)
- V1.12 ... SmartRail (ab 2012)

2.21 LAN_GET_CODE

Mit diesem Kommando kann der SW Feature-Umfang der Z21 geprüft und ausgelesen werden.

Dieses Kommando ist besonders bei der Hardwarevariante „z21 start“ von Interesse, um überprüfen zu können, ob das Fahren und Schalten per LAN gesperrt oder erlaubt ist.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x04	0x00	**0x18**	0x00	-

Antwort von Z21:

DataLen		**Header**		**Data**
0x05	0x00	**0x18**	0x00	Code (8 Bit)

Code:

```
#define Z21_NO_LOCK         0x00  // keine Features gesperrt
#define z21_START_LOCKED    0x01  // „z21 start”: Fahren und Schalten per LAN gesperrt
#define z21_START_UNLOCKED 0x02  // „z21 start”: alle Feature-Sperren aufgehoben
```

3 Einstellungen

Die folgenden hier beschriebenen Einstellungen werden in der Z21 persistent abgespeichert.
Diese Einstellungen können vom Anwender auf die Werkseinstellung zurückgesetzt werden, indem die STOP-Taste an der Z21 gedrückt bleibt wird bis die LEDs violett blinken.

3.1 LAN_GET_LOCOMODE

Lesen des Ausgabeformats für eine gegebene Lok-Adresse.

In der Z21 kann das Ausgabeformat (DCC, MM) pro Lok-Adresse persistent gespeichert werden. Es können maximal 256 verschiedene Lok-Adressen abgelegt werden. Jede Adresse >= 256 ist automatisch DCC.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x06	0x00	**0x60**	0x00	Lok-Adresse 16 bit (**big endian**)

Antwort von Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0x60**	0x00	Lok-Adresse 16 Bit (**big endian**)	Modus 8 bit

Lok-Adresse 2 Byte, **big endian** d.h. zuerst high byte, gefolgt von low byte.

Modus 0 ... DCC Format
1 ... MM Format

3.2 LAN_SET_LOCOMODE

Setzen des Ausgabeformats für eine gegebene Lok-Adresse. Das Format wird persistent in der Z21 gespeichert.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0x61**	0x00	Lok-Adresse 16 Bit (**big endian**)	Modus 8 bit

Antwort von Z21:
keine

Bedeutung der Werte siehe oben.

Anmerkung: jede Lok-Adresse >= 256 ist und bleibt automatisch „Format DCC“.

Anmerkung: die Fahrstufen (14, 28, 128) werden ebenfalls in der Zentrale persistent abgespeichert. Dies geschieht automatisch beim Fahrbefehl, siehe ***4.2*** LAN_X_SET_LOCO_DRIVE.

3.3 LAN_GET_TURNOUTMODE

Lesen der Einstellungen für eine gegebene Funktionsdecoder-Adresse („Funktionsdecoder“ im Sinne von „Accessory Decoder“ RP-9.2.1).

In der Z21 kann das Ausgabeformat (DCC, MM) pro Funktionsdecoder-Adresse persistent gespeichert werden. Es können maximal 256 verschiedene Funktionsdecoder -Adressen gespeichert werden. Jede Adresse >= 256 ist automatisch DCC.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x06	0x00	**0x70**	0x00	Funktionsdecoder-Adresse16 bit (**big endian**)

Antwort von Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0x70**	0x00	Funktionsdecoder-Adresse 16 Bit (**big endian**)	Modus 8 bit

Funktionsdecoder-Adresse 2 Byte, **big endian** d.h. zuerst high byte, gefolgt von low byte.

Modus 0 ... DCC Format
1 ... MM Format

An der LAN-Schnittstelle und in der Z21 werden die Funktionsdecoder-Adressen ab 0 adressiert, in der Visualisierung in den Apps oder auf der multiMaus jedoch ab 1. Dies ist lediglich ist eine Entscheidung der Visualisierung. Beispiel: multiMaus Weichenadresse #3, entspricht am LAN und in der Z21 der Adresse 2.

3.4 LAN_SET_TURNOUTMODE

Setzen des Ausgabeformats für eine gegebene Funktionsdecoder -Adresse. Das Format wird persistent in der Z21 gespeichert.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0x71**	0x00	Funktionsdecoder-Adresse 16 Bit (**big endian**)	Modus 8 bit

Antwort von Z21:
keine

Bedeutung der Werte siehe oben.

MM-Funktionsdecoder werden von Z21 Firmware ab Firmware Version 1.20 unterstützt.
MM-Funktionsdecoder werden von SmartRail nicht unterstützt.

Anmerkung: jede Funktionsdecoder-Adresse >= 256 ist und bleibt automatisch „Format DCC“.

4 Fahren

In diesem Kapitel werden Meldungen behandelt, die für den Fahrbetrieb mit Lok-Decodern benötigt werden.

Ein Client kann Lok-Infos mit *4.1* LAN_X_GET_LOCO_INFO abonnieren, um über zukünftige Änderungen an dieser Lok-Adresse, welche durch andere Clients oder Handregler verursacht werden, automatisch informiert zu werden. Zusätzlich muss für den Client auch der entsprechende Broadcast aktiviert sein, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001.

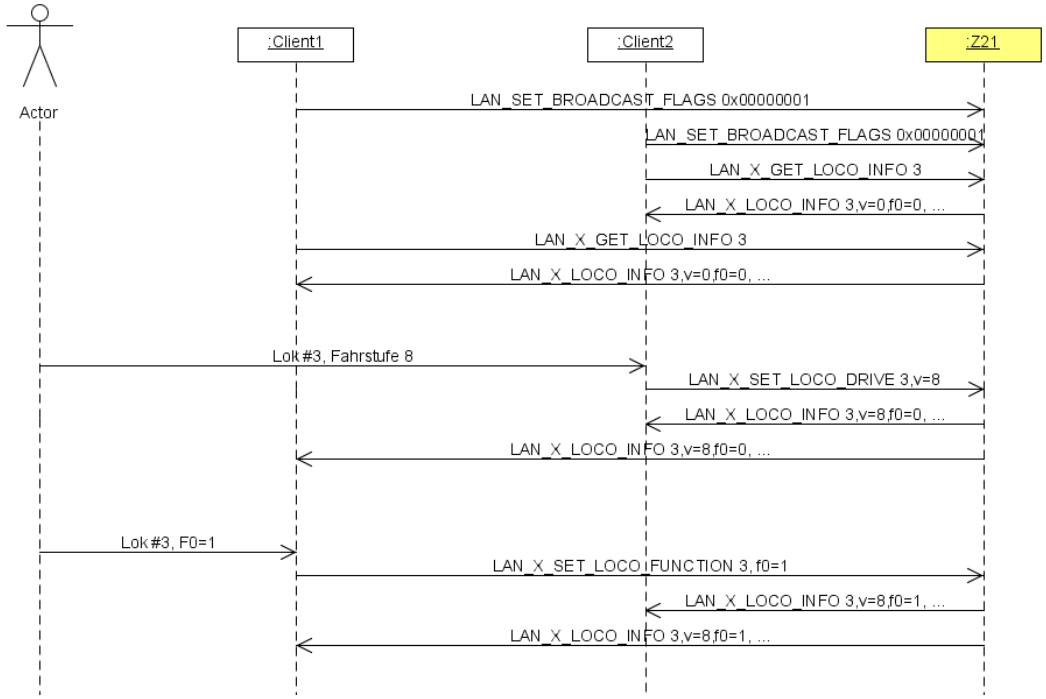

Abbildung 2 Beispiel Sequenz Lok-Steuerung

Um den Netzwerk-Verkehr in sinnvollen Schranken zu halten, können maximal 16 Lok-Adressen pro Client abonniert werden (FIFO). Es spricht zwar nichts dagegen danach weiter zu „pollen“, aber dies sollte nur mit Rücksicht auf die Netzwerkauslastung gemacht werden: die IP-Pakete dürfen vom Router bei Überlast gelöscht werden und UDP bietet keine hierfür keine Erkennungsmechanismen!

4.1 LAN_X_GET_LOCO_INFO

Mit folgendem Kommando kann der Status einer Lok angefordert werden. Gleichzeitig werden damit die Lok-Infos für diese Lok-Adresse vom Client „abonniert“ (nur in Kombination mit LAN_SET_BROADCASTFLAGS, Flag 0x00000001).

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0xE3**	**0xF0**	**Adr_MSB**	**Adr_LSB**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

Antwort von Z21:
siehe *4.4* LAN_X_LOCO_INFO

4.2 LAN_X_SET_LOCO_DRIVE

Mit folgendem Kommando kann die Fahrstufe eines Lok-Decoders verändert werden.

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0xE4**	**0x1S**	**Adr_MSB**	**Adr_LSB**	**RVVVVVVV**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

0x1**S** Anzahl der Fahrstufen, abhängig vom eingestellten Schienenformat
- S=0: DCC 14 Fahrstufen bzw. MMI mit 14 Fahrstufen und F0
- S=2: DCC 28 Fahrstufen bzw. MMII mit 14 realen Fahrstufen und F0-F4
- S=3: DCC 128 Fahrstufen (alias „126 Fahrstufen“ ohne die Stops), bzw. MMII mit 28 realen Fahrstufen (Licht-Trit) und F0-F4

RVVVVVVV
- R ... Richtung: 1=vorwärts
- V ... Geschwindigkeit: abhängig von den Fahrstufen S. Codierung siehe unten. Sollte für die Lok das Format MM konfiguriert sein, erfolgt die Umrechnung der gegebenen DCC-Fahrstufe in die reale MM-Fahrstufe automatisch in der Z21.

Die Codierung der Geschwindigkeit erfolgt ähnlich wie in NMRA S 9.2 und S 9.2.1. „**Stop**“ bedeutet „normaler Stop“ bzw. „Step 0“. „**E-Stop**“ bedeutet „Nothalt“.

Fahrstufen -Codierung bei „DCC 14“:

R000 VVVV	**Speed**	**R000 VVVV**	**Speed**	**R000 VVVV**	**Speed**	**R000 VVVV**	**Speed**
R000 0000	**Stop**	R000 0100	Step 3	R000 1000	Step 7	R000 1100	Step 11
R000 0001	**E-Stop**	R000 0101	Step 4	R000 1001	Step 8	R000 1101	Step 12
R000 0010	Step 1	R000 0110	Step 5	R000 1010	Step 9	R000 1110	Step 13
R000 0011	Step 2	R000 0111	Step 6	R000 1011	Step 10	R000 1111	Step 14 **max**

Fahrstufen-Codierung bei „DCC 28“ (ähnlich „DCC 14“ mit einem Zwischenschritt im fünften Bit **V_5**):

R00V_5 VVVV	**Speed**	**R00V_5 VVVV**	**Speed**	**R00V_5 VVVV**	**Speed**	**R00V_5 VVVV**	**Speed**
R000 0000	**Stop**	R000 0100	Step 5	R000 1000	Step 13	R000 1100	Step 21
R001 0000	**Stop**[1]	R001 0100	Step 6	R001 1000	Step 14	R001 1100	Step 22
R000 0001	**E-Stop**	R000 0101	Step 7	R000 1001	Step 15	R000 1101	Step 23
R001 0001	**E-Stop**[1]	R001 0101	Step 8	R001 1001	Step 16	R001 1101	Step 24
R000 0010	Step 1	R000 0110	Step 9	R000 1010	Step 17	R000 1110	Step 25
R001 0010	Step 2	R001 0110	Step 10	R001 1010	Step 18	R001 1110	Step 26
R000 0011	Step 3	R000 0111	Step 11	R000 1011	Step 19	R000 1111	Step 27
R001 0011	Step 4	R001 0111	Step 12	R001 1011	Step 20	R001 1111	Step 28 **max**

Fahrstufen-Codierung bei „DCC 128“:

RVVV VVVV	**Speed**
R000 0000	**Stop**
R000 0001	**E-Stop**
R000 0010	Step 1
R000 0011	Step 2
R000 0100	Step 3
R000 0101	Step 4
…	…
R111 1110	Step 125
R111 1111	Step 126 **max**

[1] Verwendung nicht empfohlen

Antwort von Z21:
keine Standardantwort, *4.4* LAN_X_LOCO_INFO an Clients mit Abo.

Anmerkung: eine Änderung der Anzahl der Fahrstufen (14/28/128) wird für die gegebene Lokadresse automatisch in der Zentrale persistent abgespeichert.

4.3 Funktionen für Fahrzeugdecoder

Funktionsbefehle von F0 bis inklusive F12 werden am Gleis - so wie die Fahrstufe und Fahrtrichtung – regelmäßig (prioritätsgesteuert), wiederholt ausgegeben.

Funktionsbefehle ab F13 werden dagegen nach einer Änderung drei Mal am Gleis ausgegeben, und danach aber aus Rücksicht auf die verfügbare Bandbreite am Gleis und im Sinne von RCN-212 bis zur nächsten Änderung nicht mehr regelmäßig wiederholt.

4.3.1 LAN_X_SET_LOCO_FUNCTION

Mit folgendem Kommando kann eine Einzelfunktion eines Lok-Decoders geschaltet werden.

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0xE4**	**0xF8**	**Adr_MSB**	**Adr_LSB**	**TTNN NNNN**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

TT Umschalttyp: 00=aus, 01=ein, 10=umschalten,11=nicht erlaubt
NNNNNN Funktionsindex, 0x00=F0 (Licht), 0x01=F1 usw.

Bei Motorola MMI kann nur F0, bei MMII F0 bis F4 geschaltet werden.
Bei DCC können hier F0 bis F28 geschaltet werden, **ab Z21 FW Version 1.42** der erweiterte Bereich von **F0 bis F31**.

Antwort von Z21:
keine Standardantwort, *4.4* LAN_X_LOCO_INFO an Clients mit Abo.

4.3.2 LAN_X_SET_LOCO_FUNCTION_GROUP

Mit folgendem Kommando kann alternativ zu den Einzelfunktionen eine ganze Funktionsgruppe eines Lok-Decoders geschaltet werden. Dabei werden bis zu 8 Funktionen mit einem einzigen Befehl geschaltet. Ab **Z21 FW Version 1.42** können DCC Funktionen bis F31 geschaltet werden, mit gewissen Einschränkungen sogar bis F68.

Der Client sollte dabei ständig den aktuellen Zustand aller Funktionen der gesteuerten Lok mitverfolgen, damit beim Senden dieses Befehls nicht versehentlich eine Funktion gelöscht wird, welche vorher eventuell von einem anderen LAN-Client oder Handregler gesetzt worden ist. Aus diesem Grund ist dieser Befehl eher für PC-Steuerungen geeignet, weil diese ohnehin den Überblick über alle Fahrzeuge behalten sollten.

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0xE4**	**Group**	**Adr_MSB**	**Adr_LSB**	**Functions**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

Group und **Functions** sind wie folgt aufgebaut:

Nummer	**Group**	**Functions**								**Anmerkungen**
	HEX	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	
1	**0x20**	0	0	0	F0	F4	F3	F2	F1	(A)
2	**0x21**	0	0	0	0	F8	F7	F6	F5	
3	**0x22**	0	0	0	0	F12	F11	F10	F9	
4	**0x23**	F20	F19	F18	F17	F16	F15	F14	F13	(B)
5	**0x28**	F28	F27	F26	F25	F24	F23	F22	F21	(B)
6	**0x29**	F36	F35	F34	F33	F32	**F31**	**F30**	**F29**	**(C)** (D) (E)
7	**0x2A**	F44	F43	F42	F41	F40	F39	F38	F37	(D) (E)
8	**0x2B**	F52	F51	F50	F49	F48	F47	F46	F45	(D) (E)
9	**0x50**	F60	F59	F58	F57	F56	F55	F54	F53	(D) (E)
10	**0x51**	F68	F67	F66	F65	F64	F63	F62	F61	(D) (E)

Anmerkungen:
(A) Beim Motorola MMI kann nur F0, bei MMII F0 bis maximal F4 geschaltet werden.
(B) DCC F13 bis F28 mit diesem Befehl erst **ab Z21 FW V1.24.**
(C) DCC F29 bis F31 **ab Z21 FW V1.42,** inklusive Rückmeldung an die LAN-Clients, siehe unten.
(D) DCC F32 bis F68 **ab Z21 FW V1.42,** es erfolgt allerdings keine Rückmeldung an die LAN-Clients, die DCC Funktionsbefehle werden nur am Gleis ausgegeben.
(E) Wir können nicht gewährleisten, dass die DCC-Funktionsbefehle ab F29 und höher auch tatsächlich von allen aktuell verfügbaren Decodern verstanden werden! Aktuell (2022) kennen tatsächlich nur sehr wenige DCC Decoder-Typen die Funktionsbefehle ab F29 (getestet wurden F29 bis F31 erfolgreich mit „Loksound 5“-Decoder). Andere Hersteller bieten inzwischen zwar auch schon Soundfunktionen auf F29, F30 oder F31 ab Werk an, was dann aber in der Praxis aber oft nicht mit DCC funktioniert, weil ihre Multiprotokoll-Decoder die entsprechenden neuen DCC Befehle noch gar nicht verstehen.

Antwort von Z21:
keine Standardantwort, für die Funktionen **F0 bis F31** erfolgt die Rückmeldung *4.4* LAN_X_LOCO_INFO an Clients mit Abo.

4.3.3 LAN_X_SET_LOCO_BINARY_STATE

Ab Z21 FW Version 1.42 kann mit folgendem Kommando ein DCC „Binary State“ Kommando an einen Lok-Decoder gesendet werden.

Anforderung an Z21:

DataLen		**Header**		**Data**						
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0xE5**	**0x5F**	**AH**	**AL**	**FLLL LLLL**	**HHHH HHHH**	XOR-Byte

Es gilt: Lok-Adresse = (**AH** & 0x3F) << 8 + **AL**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **AH**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

F Das oberste Bit **F** legt fest, ob der Binärzustand eingeschaltet oder ausgeschaltet ist.
LLLLLLL Die niederwertigen **sieben** (!) Bits der Binärzustandsadresse.
HHHHHHHH Die die höherwertigen acht Bits der Binärzustandsadresse.

Es gilt: die 15-Bit Binärzustandsadresse = (**HHHHHHHH** << **7**) + (**LLLLLLL** & 0x7F)

Erlaubt sind die Binärzustandsadressen von **29 bis 32767**.
Es dürfen für allgemeine Schaltfunktionen nur die Binärzustandsadressen ≥ 29 verwendet werden.
Die Binärzustandsadressen von 1 bis 28 sind für besondere Anwendungen reserviert.
Die Binärzustandsadresse 0 ist als Broadcast reserviert.

Binärzustandsadressen < 128 (d.h. falls **HHHHHHHH** == 0) werden **automatisch** gemäß RCN-212 als DCC „Binärzustandssteuerungsbefehl **kurze Form**“ am Gleis ausgegeben, ab ≥ 128 als DCC „Binärzustandssteuerungsbefehl **lange Form**“.

DCC Binärzustandssteuerungsbefehle werden drei Mal am Gleis ausgegeben, und danach gemäß RCN-212 nicht mehr regelmäßig wiederholt.

Es erfolgt keine Antwort an den Aufrufer und auch keine Benachrichtigung an andere Clients.

Antwort von Z21:
keine.

4.4 LAN_X_LOCO_INFO

Diese Meldung wird von der Z21 an die Clients als Antwort auf das Kommando *4.1* LAN_X_GET_LOCO_INFO gesendet. Sie wird aber auch ungefragt an Clients gesendet, wenn

- der Lok-Status durch einen der Clients oder Handregler verändert worden ist
- und der betreffende Client den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001
- und der betreffende Client die Lok-Adresse mit *4.1* LAN_X_GET_LOCO_INFO abonniert hat

Z21 an Client:

DataLen		**Header**		**Data**									
				X-Header	**DB0**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**XOR-Byte**
7 + ***n***	0x00	**0x40**	0x00	0xEF	**Lok-Information**								XOR-Byte

Die aktuelle Paketlänge kann abhängig von den tatsächlich gesendeten Daten variieren mit ***7 ≤ n ≤ 14***.
Ab Z21 FW Version 1.42 ist DataLen ≥ 15 (**n ≥ 8**), zur Übertragung des Status von F29, F30 und F31!

Die Daten für **Lok-Information** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB0	**Adr_MSB**	Die beiden höchsten Bits in Adr_MSB sind zu ignorieren.
DB1	**Adr_LSB**	Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
DB2	000**MBKKK**	**M**=1 … **Ab Z21 FW Version 1.43** Kennung für MM Lok **B**=1 ... die Lok wird von einem anderen X-BUS Handregler gesteuert („*besetzt*“) **KKK** ... Fahrstufeninformation: 0=14, 2=28, 4=128 0: DCC 14 Fahrstufen bzw. MMI mit 14 Fahrstufen und F0 2: DCC 28 Fahrstufen bzw. MMII mit 14 realen Fahrstufen und F0-F4 **4**: DCC 128 Fahrstufen bzw. MMII mit 28 realen Fahrstufen (Licht-Trit) und F0-F4
DB3	**RVVVVVVV**	**R** ... Richtung: 1=vorwärts **V** ... Geschwindigkeit. Codierung abhängig von der Fahrstufeninformation KKK. Siehe auch oben *4.2* LAN_X_SET_LOCO_DRIVE. Sollte für die Lok das Format MM konfiguriert sein, dann ist die Umrechnung der realen MM-Fahrstufe in die vorliegende DCC-Fahrstufe bereits in der Z21 erfolgt.
DB4	0**DSLFGHJ**	**D** ... Doppeltraktion: 1=Lok in Doppeltraktion enthalten. **S** ... Smartsearch **L** ... F0 (Licht) **F** ... F4 **G** ... F3 **H** ... F2 **J** ... F1
DB5	F5-F12	Funktion F5 ist bit0 (LSB)
DB6	F13-F20	Funktion F13 ist bit0 (LSB)
DB7	F21-F28	Funktion F21 ist bit0 (LSB)
DB8	**F29-F31**	**Ab Z21 FW Version 1.42** und falls DataLen ≥ 15; Funktion F29 ist bit0 (LSB)
DB***n***		optional, für zukünftige Erweiterungen

4.5 LAN_X_SET_LOCO_E_STOP

Ab Z21 FW Version 1.43 kann mit folgendem Kommando eine Lok angehalten werden. Bei einer DCC Lok wird dann am Gleis beim DCC Geschwindigkeitsbefehl die Fahrstufe „**E-STOP**“ („Nothalt“ lt. RCN-212) ausgegeben, d.h. der Decoder soll das Fahrzeug so schnell wie möglich anhalten. Bei einer MM Lok wird die Fahrstufe 0 („Stop“) ausgegeben.

Anforderung an Z21:

DataLen		**Header**		**Data**			
				X-Header	**DB0**	**DB2**	**XOR-Byte**
0x08	0x00	**0x40**	0x00	**0x92**	**Adr_MSB**	**Adr_LSB**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

Antwort von Z21:
keine Standardantwort, *4.4* LAN_X_LOCO_INFO an Clients mit Abo.

4.6 LAN_X_PURGE_LOCO

Ab Z21 FW Version 1.43 kann mit folgendem Kommando eine Lok wieder aus der Z21 herausgenommen werden. Damit wird auch die Ausgabe der Fahrbefehle für diese Lok auf dem Gleis beendet. Sie beginnt erst wieder, sobald ein neuer Fahr- oder Funktionsbefehl an dieselbe Lokadresse gesendet wird.

Auf diese Weise ist es z.B. für eine PC-Steuerung möglich, die Anzahl der Loks im System und damit auch den Datendurchsatz am Gleis zu beeinflussen.

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0xE3**	**0x44**	**Adr_MSB**	**Adr_LSB**	XOR-Byte

Es gilt: Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
Bei Lok-Adressen ≥ 128 müssen die beiden höchsten Bits in DB1 auf 1 gesetzt sein:
DB1 = (**0xC0** | **Adr_MSB**). Bei Lokadressen < 128 sind diese beiden höchsten bits ohne Bedeutung.

Es erfolgt keine Antwort an den Aufrufer und auch keine Benachrichtigung an andere Clients.

Antwort von Z21:
keine Standardantwort.

5 Schalten

In diesem Kapitel werden Meldungen behandelt, die zum Schalten von Funktionsdecodern im Sinne von „Accessory Decoder“ RP-9.2.1(d.h. Weichendecoder, ...) benötigt werden.

Die Visualisierung der Weichennummer an der Benutzeroberfläche ist bei vielen DCC-Systemen unterschiedlich gelöst und kann von der tatsächlich am Gleis verwendeten Accessorydecoder-Adresse und Port deutlich abweichen. Gemäß DCC gibt es pro Accessorydecoder-Adresse vier Ports mit je zwei Ausgängen. Pro Port kann eine Weiche angeschlossen werden. Üblicherweise wird zur Visualisierung der Weichennummer eine von folgenden Möglichkeiten verwendet:

1. Nummerierung ab 1 mit DCC-Adresse bei 1 beginnend mit je 4 Ports (ESU, Uhlenbrock, …)
   Weiche #1: DCC-Addr=1 Port=0; Weiche #5: DCC-Addr=2 Port=0; Weiche #6: DCC-Addr=2 Port=1

2. Nummerierung ab 1 mit DCC-Adresse bei 0 beginnend mit je 4 Ports (**Roco**, Lenz)
   Weiche #1: DCC-Addr=0 Port=0; Weiche #5: DCC-Addr=1 Port=0; Weiche #6: DCC-Addr=1 Port=1

3. Virtuelle Weichennummer mit frei konfigurierbarer DCC-Adresse und Port (Twin-Center)

4. Darstellung DCC-Adresse / Port (Zimo)

Keine dieser Visualisierungsmöglichkeiten kann als „falsch“ bezeichnet werden. Für den Anwender ist es allerdings gewöhnungsbedürftig, dass ein und dieselbe Weiche bei einer ESU Zentrale unter Nummer 1 gesteuert wird, während sie auf der Roco multiMaus mit Z21 unter der Nummer 5 geschaltet wird („Verschiebung um 4“).

Um in Ihrer Applikation die Visualisierung Ihrer Wahl implementieren zu können, hilft es zu wissen, wie die Z21 die Input-Parameter für die Schaltbefehle (**FAdr_MSB**, **FAdr_LSB**, **A**, **P**, siehe unten) in den entsprechenden DCC Accessory Befehl umsetzt:

DCC Basic Accessory Decoder Packet Format: {preamble} 0 10AAAAAA 0 1aaaCDDd 0 EEEEEEEE 1

UINT16 *FAdr* = (**FAdr_MSB** << 8) + **FAdr_LSB;**
UINT16 *Dcc_Addr* = *FAdr* >> 2;

aaaAAAAAA = (~*Dcc_Addr* & 0x1C0) | (*Dcc_Addr* & 0x003F); // DCC Adresse
C = **A**; // Ausgang aktivieren oder deaktivieren
DD = *FAdr* & 0x03; // Port
d = **P**; // Weiche nach links oder nach rechts

Beispiel:
FAdr=0 ergibt DCC-Addr=0 Port=0;
FAdr=3 ergibt DCC-Addr=0 Port=3;
FAdr=4 ergibt DCC-Addr=1 Port=0; usw

Bei MM Format gilt dagegen: FAdr beginnt mit 0, d.h. FAdr=0: MM-Addr=1; FAdr=1: MM-Addr=2; …

Ein Client kann Funktions-Infos abonnieren, um über Änderungen an Funktionsdecodern, welche auch durch andere Clients oder Handregler verursacht werden, automatisch informiert zu werden. Dazu muss für den Client der entsprechende Broadcast aktiviert sein, siehe ***2.16*** *LAN_SET_BROADCASTFLAGS,* Flag 0x00000001.

Die tatsächliche Stellung der Weiche hängt übrigens von der Verkabelung und eventuell auch von der Konfiguration in der Applikation des Clients ab. Davon kann die Zentrale nichts wissen, weshalb in der folgenden Beschreibung auf die Bezeichnungen „*gerade*“ und „*abzweigend*“ bewusst verzichtet wird.

5.1 LAN_X_GET_TURNOUT_INFO

Mit folgendem Kommando kann der Status einer Weiche (bzw. Schaltfunktion) angefordert werden.

Anforderung an Z21:

DataLen		**Header**		**Data**			
				X-Header	**DB0**	**DB1**	**XOR-Byte**
0x08	0x00	**0x40**	0x00	**0x43**	**FAdr_MSB**	**FAdr_LSB**	XOR-Byte

Es gilt: Funktions-Adresse = (**FAdr_MSB** << 8) + **FAdr_LSB**

Antwort von Z21:
siehe *5.3* LAN_X_TURNOUT_INFO

5.2 LAN_X_SET_TURNOUT

Mit folgendem Kommando kann eine Weiche geschaltet werden.

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0x53**	**FAdr_MSB**	**FAdr_LSB**	10**Q**0**A**00**P**	XOR-Byte

Es gilt: Funktions-Adresse = (**FAdr_MSB** << 8) + **FAdr_LSB**

1000**A**00**P** **A**=0 ... Weichenausgang deaktivieren
A=1 ... Weichenausgang aktivieren
P=0 ... Ausgang 1 der Weiche wählen
P=1 ... Ausgang 2 der Weiche wählen
Q=0 … Kommando sofort ausführen
Q=1 … **ab Z21 FW V1.24**: Weichenbefehl in der Z21 in die Queue einfügen und zum nächstmöglichen Zeitpunkt am Gleis ausgeben.

Antwort von Z21:
keine Standardantwort , *5.3* LAN_X_TURNOUT_INFO an Clients mit Abo.

Ab Z21 FW V1.24 wurde das Q-Flag („Queue“) eingeführt.

5.2.1 LAN_X_SET_TURNOUT mit Q=0

Wenn **Q=0** ist, dann verhält sich die Z21 kompatibel zu den bisherigen Versionen: der Weichenstellbefehl wird sofort auf das Gleis ausgegeben, indem er in die laufenden Fahrbefehle gemischt wird**. Das Activate (A=1) wird solange ausgegeben, bis vom LAN-Client das entsprechende Deactivate geschickt wird. Es darf zu einem Zeitpunkt nur ein Weichenstellstellbefehl aktiv sein.** Dieses Verhalten entspricht z.B. dem Drücken und Loslassen der multiMaus-Tasten.

Beachten Sie, dass bei Q=0 unbedingt die korrekte Reihenfolge der Schaltbefehle (d.h. Activate gefolgt von Deactivate) eingehalten werden muss. Ansonsten kann es je nach verwendetem Weichendecoder zu undefinierten Endstellungen kommen.

Die korrekte Serialisierung und das Timing der Schaltdauer liegen in der Verantwortung des LAN-Clients!

Falsch:
Weiche #5/A2 aktivieren (4,0x89); Weiche #6/A2 aktivieren (5,0x89);
Weiche #3/A1 aktivieren (2,0x88); Weiche #3/A1 deaktivieren (2,0x80);
Weiche #5/A2 deaktivieren (4,0x81); Weiche #6/A2 deaktivieren (5,0x81);

Richtig:
Weiche #5/A2 aktivieren (4,0x89); 100ms warten; Weiche #5/A2 deaktivieren (4,0x81); 50ms warten;
Weiche #6/A2 aktivieren (5,0x89); 100ms warten; Weiche #6/A2 deaktivieren (5,0x81); 50ms warten;
Weiche #3/A1 aktivieren (2,0x88); 100ms warten; Weiche #3/A1 deaktivieren (2,0x80); 50ms warten;

Beilspiel:
Weiche #7 / A2 aktivieren (6,0x89); 150ms warten; Weiche #7 / A2 deaktivieren (6,0x81)

```
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=1 , "Roco_lenz f=7 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=0 , "Roco_lenz f=7 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=0 , "Roco_lenz f=7 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=0 , "Roco_lenz f=7 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
DCC  preamble=16  ACESSORY raw data AA=1 DD=5 C=0 , "Roco_lenz f=7 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 FG2 (5-8) F=o7oo
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 FG1 (0-4) F=Loooo
```

Abbildung 3 DCC Sniff am Gleis bei Q=0

5.2.2 LAN_X_SET_TURNOUT mit Q=1

Wenn **Q=1** ist, ergibt sich folgendes Verhalten: der Schaltbefehl wird zuerst in der Z21 in einer internen Queue (FIFO) eingereiht. Beim Generieren des Gleissignals wird diese Queue ständig geprüft, ob ein Schaltbefehl zur Ausgabe anliegt. Dieser Schaltbefehl wird dann ggf. aus der Queue herausgenommen und viermal am Gleis ausgegeben. Dies befreit den LAN-Client von der bisher obligatorischen Serialisierung, d.h. die Schaltbefehle dürfen bei Q=1 gemischt an die Z21 gesendet werden (Fahrstraßen!). Der LAN-Client braucht sich nur mehr um das Timing des Deactivate kümmern. Das Deactivate darf je nach DCC-Decoder unter Umständen sogar entfallen. Bei MM sollte aber keinesfalls darauf verzichtet werden, denn z.B. der k83 und ältere Weichenantriebe besitzen keine Endabschaltung.

Beispiel:
Weiche #25 / A2 aktivieren (24, 0xA9); Weiche #5 / A2 aktivieren (4, 0xA9);
150ms warten;
Weiche #25 / A2 deaktivieren (24, 0xA1)

```
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=1 , "Roco_lenz f=25 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=1 , "Roco_lenz f=25 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=1 , "Roco_lenz f=25 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=1 , "Roco_lenz f=25 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=1 C=1 , "Roco_lenz f=5 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=1 C=1 , "Roco_lenz f=5 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=1 C=1 , "Roco_lenz f=5 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=1 DD=1 C=1 , "Roco_lenz f=5 out=A  ACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=0 , "Roco_lenz f=25 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=0 , "Roco_lenz f=25 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=0 , "Roco_lenz f=25 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  ACESSORY raw data AA=6 DD=1 C=0 , "Roco_lenz f=25 out=A  INACTIVE"
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
DCC  preamble=16  LOCO address=3 ss128=0 fwd Speed=Stop
```

Abbildung 4 DCC Sniff am Gleis bei Q=1

Vermischen Sie in Ihrer Applikation keinesfalls Schaltbefehle mit Q=0 und Schaltbefehle mit Q=1.

5.3 LAN_X_TURNOUT_INFO

Diese Meldung wird von der Z21 an die Clients als Antwort auf das Kommando *5.1* LAN_X_GET_TURNOUT_INFO gesendet. Sie wird aber auch ungefragt an Clients gesendet, wenn

- der Funktions-Status durch einen der Clients oder Handregler verändert worden ist
- und der betreffende Client den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0x43**	**FAdr_MSB**	**FAdr_LSB**	000000**ZZ**	XOR-Byte

Es gilt: Funktions-Adresse = (**FAdr_MSB** << 8) + **FAdr_LSB**

000000**ZZ** **ZZ**=00 ... Weiche noch nicht geschaltet
ZZ=01 ... Weiche steht gemäß Schaltbefehl „P=0“, siehe *5.2* LAN_X_SET_TURNOUT
ZZ=10 ... Weiche steht gemäß Schaltbefehl „P=1“, siehe *5.2* LAN_X_SET_TURNOUT
ZZ=11 ... ungültige Kombination

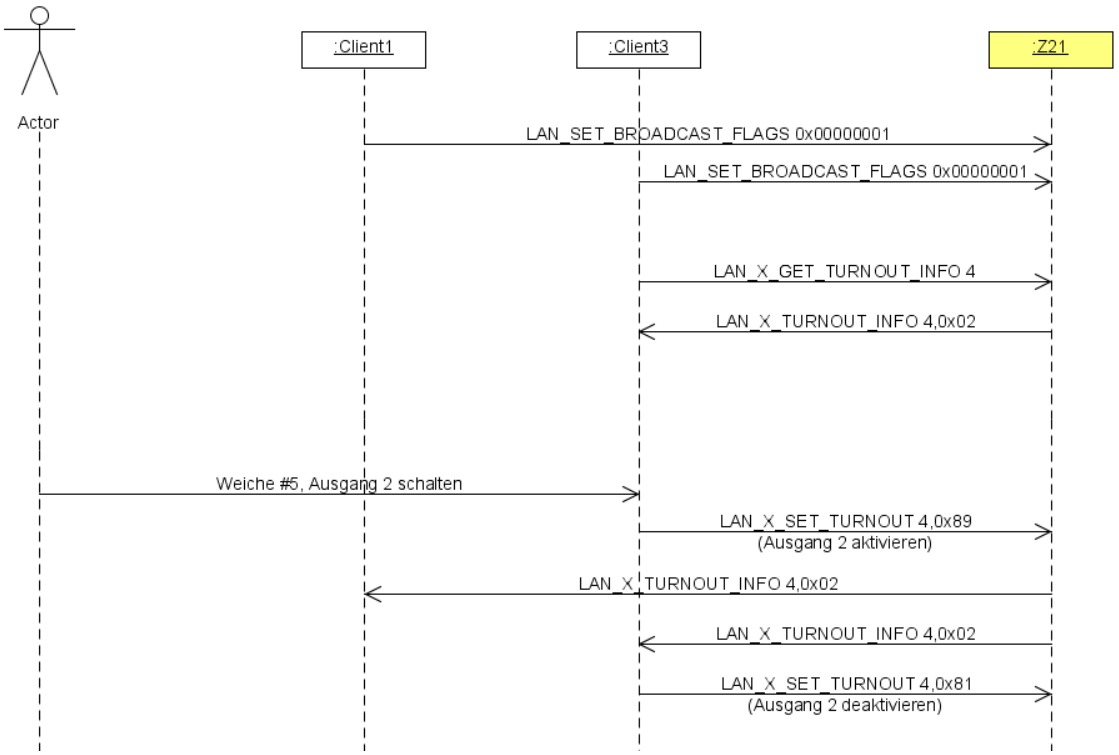

Abbildung 5 Beispiel Sequenz Weiche schalten

5.4 LAN_X_SET_EXT_ACCESSORY

Ab Z21 FW V1.40 kann mit folgendem Kommando ein DCC Befehl im „erweiterten Zubehördecoder Paketformat“ (DCCext) an einen Erweiterten Zubehördecoder gesendet werden. Damit ist es möglich, Schaltzeiten für Weichen oder komplexere Signalbegriffe mit nur einem Kommando zu versenden. Siehe RCN-213 (Abschnitt 2.3).

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0x54**	**Adr_MSB**	**Adr_LSB**	DDDDDDDD	0x00	XOR-Byte

Es gilt: **RawAddress** = (**Adr_MSB** << 8) + **Adr_LSB**

RawAddress Die RawAddress für den ersten erweiterten Zubehördecoder ist gemäß RCN-213 die Adresse 4. Diese Adresse wird in Anwenderdialogen als „Adresse 1“ dargestellt.
Die Adressierung richtet sich strikt nach RCN-213, d.h. es gibt hier *keine* abweichende Adressverschiebung mehr.

DDDDDDDD Über die Bits 0 bis 7 in DB2 werden die 256 möglichen Zustände übertragen.
Der Inhalt wird am Gleis im ***Erweiterten Zubehördecoder Paketformat*** gemäß RCN-213 an den Decoder übertragen.

Hinweis:
Der **10836 Z21 switch DECODER** interpretiert DDDDDDDD wie ein „einfacher Schaltdecoder mit Empfang der Schaltzeit“ als **RZZZZZZZ**. Dabei gilt:

- **ZZZZZZZ** legt die Einschaltzeit mit einer Auflösung von **100 ms** fest.
  - Der Wert 0 bedeutet, dass der Ausgang ausgeschaltet wird.
  - Der Wert 127 bedeutet, dass der Ausgang dauerhaft, d.h. bis zum nächsten Befehl an diese Adresse, eingeschaltet wird.
- Bit 7 **R** wird benutzt, um den Ausgang innerhalb des Paares auszuwählen:
  - R=1 bedeutet „grün“ (gerade).
  - R=0 bedeutet „rot“ (abzweigend).

Der **10837 Z21 signal DECODER** interpretiert DDDDDDDD dagegen als einen von 256 theoretisch möglichen Signalbegriffen. Der tatsächlich verfügbare Wertebereich hängt stark vom im Signal-Decoder eingestellten Signaltyp ab. Mögliche Werte sind zum Beispiel:

- 0 ... absoluter Haltebegriff
- 4 ... Fahrt mit Geschwindigkeitsbegrenzung 40 km/h
- 16 ... freie Fahrt
- 65 (0x41) ... Rangieren erlaubt
- 66 (0x42) ... Dunkelschaltung (z.B. Lichtvorsignale)
- 69 (0x45) ... Ersatzsignal (erlaubt die Vorbeifahrt)

Den konkreten Wert zum gewünschten Signalbegriff für ein gegebenes Signals finden Sie für den Z21 signal DECODER unter https://www.z21.eu/de/produkte/z21-signal-decoder/signaltypen jeweils unter „DCCext“.

Antwort von Z21:
keine Standardantwort, oder *5.6 LAN_X_EXT_ACCESSORY_INFO* an Clients mit Abo.

Beispiel:
0x0A 0x00 0x40 0x00 0x54 0x00 0x04 0x05 0x00 0x55
bedeutet „Sende an Decoder mit RawAddress=4 (diese Adresse wird in Anwenderdialogen als Adresse 1 dargestellt!) den Wert DDDDDDDD=5.“
Ist der Empfänger ein 10836 Z21 switch DECODER, dann wird dadurch der Ausgang 1 „rot“ (Klemme 1A) eingeschaltet und nach 5*100ms automatisch wieder ausgeschaltet.

Mit diesem Kommando ist es auch möglich, den „Notaus-Befehl für Erweiterte Zubehördecoder“ gemäß RCN-213 (Abschnitt 2.4) zu versenden. Das entspricht dem Wert 0 („Halt bei Lichtsignalen“) für die RawAddress=2047:
0x0A 0x00 0x40 0x00 0x54 0x07 0xFF 0x00 0x00 0xAC

5.5 LAN_X_GET_EXT_ACCESSORY_INFO

Ab Z21 FW V1.40 kann mit folgendem Kommando der letzte an einen **Erweiterten Zubehördecoder** übertragene Befehl abgefragt werden.

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0x44**	**Adr_MSB**	**Adr_LSB**	0x00	XOR-Byte

Es gilt: **RawAddress** = (**Adr_MSB** << 8) + **Adr_LSB**

RawAddress Die Adresse des Zubehördecoders gemäß RCN-213.
Siehe Abschnitt *5.4* LAN_X_SET_EXT_ACCESSORY.

DB2 Reserviert für zukünftige Erweiterungen, sollte bis auf weiteres mit 0 initialisiert bleiben.

Antwort von Z21:
siehe *5.6* LAN_X_EXT_ACCESSORY_INFO

5.6 LAN_X_EXT_ACCESSORY_INFO

Diese Meldung wird von der Z21 an die Clients als Antwort auf das Kommando *5.5 LAN_X_GET_EXT_ACCESSORY_INFO* gesendet. Sie wird aber auch ungefragt an Clients gesendet, wenn

- irgendjemand anderer ein Kommando an einen Erweiterten Zubehördecoder sendet
- und der betreffende Client den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000001

Z21 an Client:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0x44**	**Adr_MSB**	**Adr_LSB**	**DDDDDDDD**	**Status**	XOR-Byte

Es gilt: **RawAddress** = (**Adr_MSB** << 8) + **Adr_LSB**

RawAddress Die Adresse des Zubehördecoders gemäß RCN-213.
Siehe Abschnitt *5.4* LAN_X_SET_EXT_ACCESSORY.

DDDDDDDD Bis zu 256 mögliche Zustände, codiert im ***Erweiterten Zubehördecoder Paketformat*** gemäß RCN-213.
Siehe Abschnitt *5.4* LAN_X_SET_EXT_ACCESSORY.

Status 0x00 … Data Valid
0xFF … Data Unknown

6 Decoder CV Lesen und Schreiben

In diesem Kapitel werden Meldungen behandelt, die zum Lesen und Schreiben von Decoder-CVs (Configuration Variable, RP-9.2.2, RP-9.2.3) benötig werden.

Ob der Zugriff am Decoder bit- oder byteweise geschieht, hängt von den Einstellungen in der Z21 ab.

6.1 LAN_X_CV_READ

Mit folgendem Kommando kann eine CV im Direct-Mode ausgelesen werden

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB1**	**DB2**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0x23**	**0x11**	**CVAdr_MSB**	**CVAdr_LSB**	XOR-Byte

Es gilt: CV-Adresse = (**CVAdr_MSB** << 8) + **CVAdr_LSB**, sowie 0=CV1., 1=CV2, 255=CV256, usw.

Antwort von Z21:
2.9 LAN_X_BC_PROGRAMMING_MODE an Clients mit Abo, sowie das Ergebnis
6.3 LAN_X_CV_NACK_SC*, 6.4* LAN_X_CV_NACK oder *6.5* LAN_X_CV_RESULT.

6.2 LAN_X_CV_WRITE

Mit folgendem Kommando kann eine CV im Direct-Mode überschrieben werden.

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0x24**	**0x12**	**CVAdr_MSB**	**CVAdr_LSB**	**Value**	XOR-Byte

Es gilt: CV-Adresse = (**CVAdr_MSB** << 8) + **CVAdr_LSB**, sowie 0=CV1., 1=CV2, 255=CV256, usw.

Antwort von Z21:
2.9 LAN_X_BC_PROGRAMMING_MODE an Clients mit Abo, sowie das Ergebnis
6.3 LAN_X_CV_NACK_SC*, 6.4* LAN_X_CV_NACK oder *6.5* LAN_X_CV_RESULT.

6.3 LAN_X_CV_NACK_SC

Wenn die Programmierung aufgrund eines Kurzschlusses am Gleis fehlerhaft war, wird diese Meldung automatisch an den Client geschickt, der die Programmierung durch *6.1* LAN_X_CV_READ oder *6.2* LAN_X_CV_WRITE veranlasst hat.

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x12**	0x73

6.4 LAN_X_CV_NACK

Wenn das ACK vom Decoder ausbleibt, wird diese Meldung automatisch an den Client geschickt, der die Programmierung durch *6.1* LAN_X_CV_READ oder *6.2* LAN_X_CV_WRITE veranlasst hat.
Bei byteweisen Zugriff kann beim Lesen die Zeit bis *LAN_X_CV_NACK* sehr lange dauern.

Z21 an Client:

DataLen		**Header**		**Data**		
				X-Header	**DB0**	**XOR-Byte**
0x07	0x00	**0x40**	0x00	**0x61**	**0x13**	0x72

6.5 LAN_X_CV_RESULT

Diese Meldung ist gleichzeitig ein „positives ACK“ und wird automatisch an den Client geschickt, der die Programmierung durch *6.1* LAN_X_CV_READ oder *6.2* LAN_X_CV_WRITE veranlasst hat.
Bei byteweisen Zugriff kann beim Lesen die Zeit bis *LAN_X_CV_RESULT* sehr lange dauern.

Z21 an Client:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0x64**	**0x14**	**CVAdr_MSB**	**CVAdr_LSB**	**Value**	XOR-Byte

Es gilt: CV-Adresse = (**CVAdr_MSB** << 8) + **CVAdr_LSB**, sowie 0=CV1., 1=CV2, 255=CV256, usw.

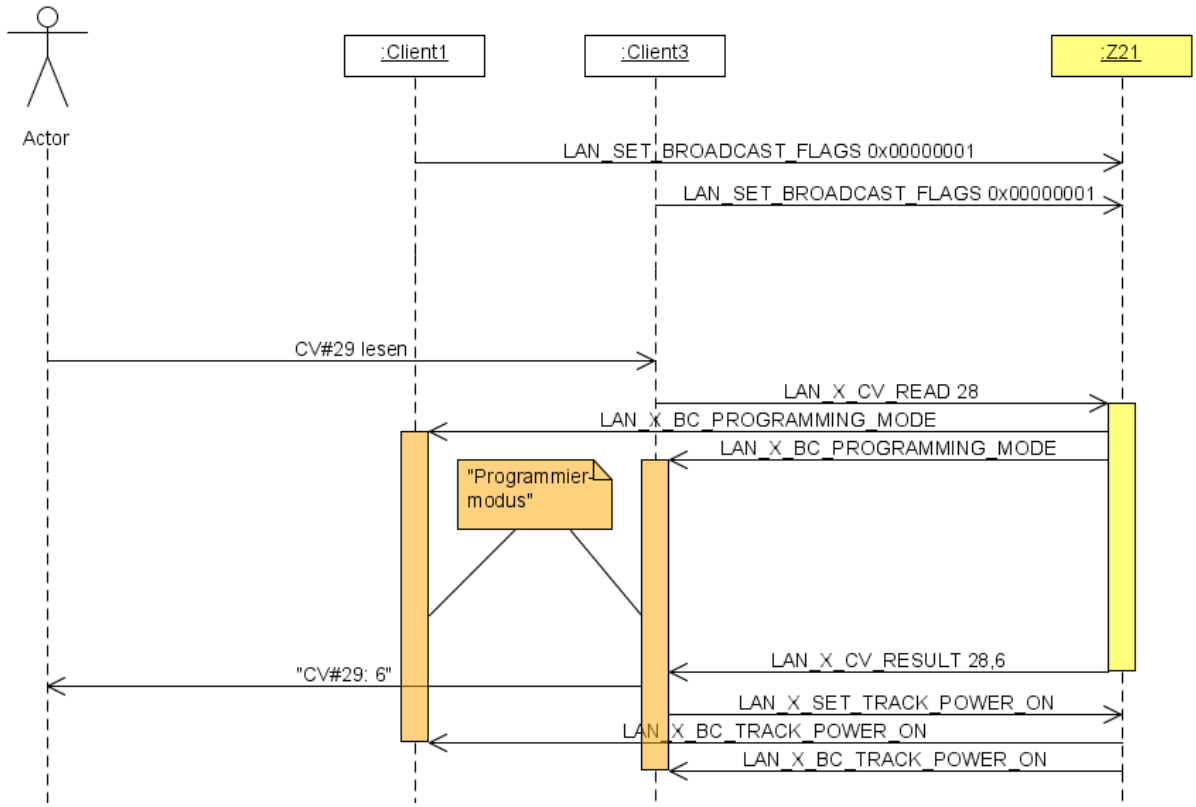

Abbildung 6 Beispiel Sequenz CV Lesen

6.6 LAN_X_CV_POM_WRITE_BYTE

Mit folgendem Kommando kann eine CV eines Lokdecoders (Multi Function Digital Decoders gemäß NMRA S-9.2.1 Abschnitt C; Configuration Variable Access Instruction - Long Form) auf dem Hauptgleis geschrieben werden (POM „Programming on the Main“). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. Es gibt keine Rückmeldung.

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x30**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**Adr_MSB**	
DB2	**Adr_LSB**	Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
DB3	**111011MM**	**Option** ... **0xEC** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	**Value**	neuer CV-Wert

Antwort von Z21:
keine

6.7 LAN_X_CV_POM_WRITE_BIT

Mit folgendem Kommando kann ein Bit einer CV eines Lokdecoders (Multi Function Digital Decoders gemäß NMRA S-9.2.1 Abschnitt C; Configuration Variable Access Instruction - Long Form) auf dem Hauptgleis geschrieben werden (POM). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. Es gibt keine Rückmeldung.

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x30**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**Adr_MSB**	
DB2	**Adr_LSB**	Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
DB3	**111010MM**	**Option** ... **0xE8** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	0000**VPPP**	**PPP** ... Bit-Position in CV **V** ... neuer Bit-Wert

Antwort von Z21:
keine

6.8 LAN_X_CV_POM_READ_BYTE

Ab Z21 FW Version 1.22.

Mit folgendem Kommando kann eine CV eines Lokdecoders (Multi Function Digital Decoders gemäß NMRA S-9.2.1 Abschnitt C; Configuration Variable Access Instruction - Long Form) auf dem Hauptgleis gelesen werden (POM). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. RailCom muss in der Z21 aktiviert sein. Der zu lesende Fahrzeugdecoder muss RailCom beherrschen, CV28 bit 0 und 1 sowie CV29 bit 3 müssen im Lokdecoder auf 1 gesetzt sein (Zimo).

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x30**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**Adr_MSB**	
DB2	**Adr_LSB**	Lok-Adresse = (**Adr_MSB** & 0x3F) << 8 + **Adr_LSB**
DB3	**111001MM**	**Option** ... **0xE4** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	**0**	neuer CV-Wert

Antwort von Z21:
6.4 LAN_X_CV_NACK oder *6.5* LAN_X_CV_RESULT.

6.9 LAN_X_CV_POM_ACCESSORY_WRITE_BYTE

Ab Z21 FW Version 1.22.

Mit folgendem Kommando kann eine CV eines Accessory Decoders (gemäß NMRA S-9.2.1 Abschnitt D, Basic Accessory Decoder Packet address for operations mode programming) auf dem Hauptgleis geschrieben werden (POM). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. Es gibt keine Rückmeldung.

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x31**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**aaaaa**	Decoder_Adresse MSB
DB2	**AAAACDDD**	Es gilt: **aaaaaAAAACDDD** = ((Decoder_Address & 0x1FF) << 4) | CDDD; Falls **CDDD**=0000, dann bezieht sich die CV auf den ganzen Decoder. Falls **C**=1, so ist **DDD** die Nummer des zu programmierenden Ausgangs.
DB3	**111011MM**	**Option** ... **0xEC** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	**Value**	neuer CV-Wert

Antwort von Z21:
keine

6.10 LAN_X_CV_POM_ ACCESSORY_WRITE_BIT

Ab Z21 FW Version 1.22.

Mit folgendem Kommando kann ein Bit einer CV eines Accessory Decoders (gemäß NMRA S-9.2.1 Abschnitt D, Basic Accessory Decoder Packet address for operations mode programming) auf dem Hauptgleis geschrieben werden (POM). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. Es gibt keine Rückmeldung.

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x31**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**aaaaa**	Decoder_Adresse MSB
DB2	**AAAACDDD**	Es gilt: **aaaaaAAAACDDD** = ((Decoder_Address & 0x1FF) << 4) | CDDD; Falls **CDDD**=0000, dann bezieht sich die CV auf den ganzen Decoder. Falls **C**=1, so ist **DDD** die Nummer des zu programmierenden Ausgangs.
DB3	**111010MM**	**Option** ... **0xE8** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	0000**VPPP**	**PPP** ... Bit-Position in CV **V** ... neuer Bit-Wert

Antwort von Z21:
keine

6.11 LAN_X_CV_POM_ ACCESSORY_READ_BYTE

Ab Z21 FW Version 1.22.

Mit folgendem Kommando kann eine CV eines Accessory Decoders (gemäß NMRA S-9.2.1 Abschnitt D, Basic Accessory Decoder Packet address for operations mode programming) auf dem Hauptgleis gelesen werden POM). Das geschieht im normalen Betriebsmodus, d.h. die Gleisspannung muss eingeschaltet sein, der normale Programmiermodus ist nicht aktiviert. RailCom muss in der Z21 aktiviert sein. Der zu lesende Accessory Decoder muss RailCom beherrschen.

Anforderung an Z21:

DataLen		**Header**		**Data**							
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**DB4**	**DB5**	**XOR-Byte**
0x0C	0x00	**0x40**	0x00	**0xE6**	**0x31**	**POM-Parameter**					XOR-Byte

Die Daten für **POM-Parameter** sind folgendermaßen aufgebaut:

Position	**Daten**	**Bedeutung**
DB1	**aaaaa**	Decoder_Adresse MSB
DB2	**AAAACDDD**	Es gilt: **aaaaaAAAACDDD** = ((Decoder_Addresse & 0x1FF) << 4) | CDDD; Falls **CDDD**=0000, dann bezieht sich die CV auf den ganzen Decoder. Falls **C**=1, so ist **DDD** die Nummer des betreffenden Ausgangs.
DB3	**111001MM**	**Option** ... **0xE4** **MM** ... CVAdr_MSB
DB4	**CVAdr_LSB**	CV-Adresse = (**MM** << 8) + **CVAdr_LSB** (0=CV1., 1=CV2, 255=CV256, usw.)
DB5	**0**	neuer CV-Wert

Antwort von Z21:
6.4 LAN_X_CV_NACK oder *6.5* LAN_X_CV_RESULT.

6.12 LAN_X_MM_WRITE_BYTE

Ab Z21 FW Version 1.23.

Mit folgendem Kommando kann ein Register eines Motorola Decoders auf dem Programmiergleis überschrieben werden.

Anforderung an Z21:

DataLen		**Header**		**Data**					
				X-Header	**DB0**	**DB1**	**DB2**	**DB3**	**XOR-Byte**
0x0A	0x00	**0x40**	0x00	**0x24**	**0xFF**	**0**	**RegAdr**	**Value**	XOR-Byte

Es gilt für **RegAdr**: 0=Register1, 1=Register2, …, 78=Register79.
Es gilt 0 ≤ **Value** ≤ 255, aber einige Decoder akzeptieren nur Werte von 0 bis 80.

Antwort von Z21:
2.9 LAN_X_BC_PROGRAMMING_MODE an Clients mit Abo, sowie das Ergebnis
6.3 LAN_X_CV_NACK_SC oder *6.5* LAN_X_CV_RESULT.

Anmerkung: Das Programmieren von Motorola-Decodern war im ursprünglichen Motorola-Format nicht vorgesehen. Daher gibt es zum Programmieren von Motorola-Decodern kein genormtes und verbindliches Programmierverfahren. Für die Programmierung von Motorola Decodern wurde in der Z21 der später eingeführte, sogenannte „6021-Programmiermodus“ implementiert. Dieser erlaubt das Schreiben von Werten, jedoch nicht das auslesen. Ebenso kann der Erfolg der Schreibeoperation nicht überprüft werden (ausgenommen Kurzschlusserkennung). Dieses Programmierverfahren funktioniert für viele Decoder von ESU, Zimo und Märklin, jedoch nicht zwingend für alle MM-Decoder. Beispielsweise können Motorola-Decoder mit DIP-Schaltern nicht programmiert werden. Manche Decoder akzeptieren nur Werte von 0 bis 80, andere Werte von 0 bis 255 (siehe Decoder-Beschreibung).

Da bei der Motorola-Programmierung vom Decoder keinerlei Rückmeldung über den Erfolg der Schreibeoperation kommt, ist hier die Meldung *LAN_X_CV_RESULT* lediglich als *„MM Programmiervorgang beendet“* und **nicht** als *„MM Programmiervorgang erfolgreich“* zu verstehen.

Beispiel:
`0x0A 0x00 0x40 0x00 0x24 0xFF 0x00 0x00 0x05 0xDE`
bedeutet: „Ändere die Lokdecoder-Adresse (**Register1**) auf **5**“

6.13 LAN_X_DCC_READ_REGISTER

Ab Z21 FW Version 1.25.

Mit folgendem Kommando kann ein Register eines DCC Decoders im Registermodus (S-9.2.3 Service Mode Instruction Packets for Physical Register Addressing) auf dem Programmiergleis ausgelesen werden.

Anforderung an Z21:

DataLen		**Header**		**Data**			
				X-Header	**DB0**	**DB1**	**XOR-Byte**
0x08	0x00	**0x40**	0x00	**0x22**	**0x11**	**REG**	XOR-Byte

Es gilt für **REG**: 0x01=Register1, 0x02=Register2, …, 0x08=Register8.
Es gilt 0 ≤ **Value** ≤ 255

Antwort von Z21:
2.9 LAN_X_BC_PROGRAMMING_MODE an Clients mit Abo, sowie das Ergebnis
6.3 LAN_X_CV_NACK_SC oder *6.5* LAN_X_CV_RESULT.

Anmerkung: Das Programmieren im Registermodus wird nur für sehr alte DCC Decoder benötigt. Direct CV ist möglichst zu bevorzugen.

6.14 LAN_X_DCC_WRITE_REGISTER

Ab Z21 FW Version 1.25.

Mit folgendem Kommando kann ein Register eines DCC Decoders im Registermodus (S-9.2.3 Service Mode Instruction Packets for Physical Register Addressing) auf dem Programmiergleis überschrieben werden.

Anforderung an Z21:

DataLen		**Header**		**Data**				
				X-Header	**DB0**	**DB2**	**DB3**	**XOR-Byte**
0x09	0x00	**0x40**	0x00	**0x23**	**0x12**	**REG**	**Value**	XOR-Byte

Es gilt für **REG**: 0x01=Register1, 0x02=Register2, …, 0x08=Register8.
Es gilt 0 ≤ **Value** ≤ 255

Antwort von Z21:
2.9 LAN_X_BC_PROGRAMMING_MODE an Clients mit Abo, sowie das Ergebnis
6.3 LAN_X_CV_NACK_SC oder *6.5* LAN_X_CV_RESULT.

Anmerkung: Das Programmieren im Registermodus wird nur für sehr alte DCC Decoder benötigt. Direct CV ist möglichst vorzuziehen.

7 Rückmelder – R-BUS

Die Rückmeldemodule (Bestellnummer 10787, 10808 und 10819) am R-BUS können mit den folgenden Kommandos ausgelesen und konfiguriert werden.

7.1 LAN_RMBUS_DATACHANGED

Änderung am Rückmeldebus von der Z21 an den Client melden.

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS*,* Flag 0x00000002
- oder den Rückmelder-Status explizit angefordert hat, siehe unten *7.2* LAN_RMBUS_GETDATA.

Z21 an Client:

DataLen		**Header**		**Data**	
0x0F	0x00	**0x80**	0x00	**Gruppenindex** (1 Byte)	**Rückmelder-Status** (10 Byte)

Gruppenindex: 0 ... Rückmeldemodule mit Adressen von 1 bis 10
1 ... Rückmeldemodule mit Adressen von 11 bis 20

Rückmelder-Status: 1 Byte pro Rückmelder, 1 bit pro Eingang.
Die Zuordnung Rückmelder-Adresse und Byteposition ist statisch aufsteigend.

Beispiel:
GruppenIndex = 1 und Rückmelder-Status = 0x01 0x00 0xC5 0x00 0x00 0x00 0x00 0x00 0x00 0x00
bedeutet „Rückmelder 11, Kontakt auf Eingang 1; Rückmelder 13, Kontakt auf Eingang 8,7,3 und 1“

7.2 LAN_RMBUS_GETDATA

Anfordern des aktuellen Rückmelder-Status.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x05	0x00	**0x81**	0x00	**Gruppenindex** (1 Byte)

Gruppenindex: siehe oben

Antwort von Z21:
Siehe oben *7.1* LAN_RMBUS_DATACHANGED

7.3 LAN_RMBUS_PROGRAMMODULE

Ändern der Rückmelder-Adresse.

Anforderung an Z21:

DataLen		Header		Data
0x05	0x00	**0x82**	0x00	**Adresse** (1 Byte)

Adresse: neue Adresse für das zu programmierende Rückmeldemodul.
Unterstützter Wertebereich: 0 und 1 ... 20.

Antwort von Z21:
keine

Der Programmierbefehl wird am R-BUS solange ausgegeben, bis dieser Befehl erneut an die Z21 mit der Adresse=0 gesendet wird.

Während des Programmiervorgangs darf sich kein anderes Rückmeldemodul am R-BUS befinden.

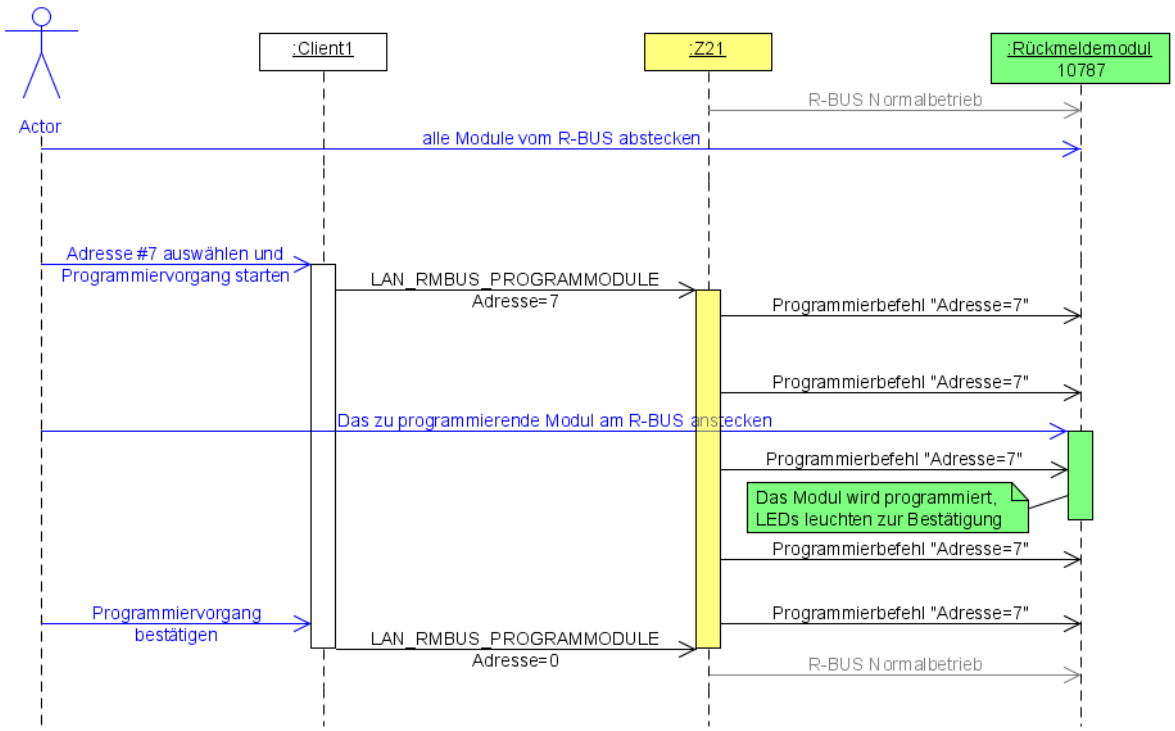

Abbildung 7 Beispiel Sequenz Rückmeldemodul programmieren

8 RailCom

Die Z21 unterstützt RailCom durch:

- Erzeugung der RailCom-Lücke am Gleissignal.
- Globaler Empfänger in der Z21.
- Lokale Empfänger, z.B. in den Belegtmeldern 10808 für die Lokerkennung.
  Zusätzlich können beim 10808 die Daten vom RailCom-Kanal 2 über CAN an die Z21 weitergeleitet und dort ausgewertet werden ab FW V1.29.
- POM-Lesen.
  Siehe auch 6.8 LAN_X_CV_POM_READ_BYTE ab FW V1.22.
- Lokadressen-Erkennung bei Belegtmeldern.
  Siehe 9.5 LAN_LOCONET_DETECTOR ab V1.22 und 10.1 LAN_CAN_DETECTOR ab V1.30.
- Decoder-Geschwindigkeit (siehe unten) ab FW V1.29.
- Decoder-QoS (siehe unten) ab FW V1.29.

Um diese Leistungsmerkmale nutzen zu können, muss der Decoder RailCom-fähig, CV28 und CV29 korrekt konfiguriert und die Option „RailCom“ in den Einstellungen der Z21 aktiviert sein.

Ob und in welcher Form ein Decoder die Geschwindigkeit, QoS und POM unterstützt, hängt von der Decoder-Firmware ab.

8.1 LAN_RAILCOM_DATACHANGED

Diese Meldung wird von der Z21 ab FW Version 1.29 an die Clients als Antwort auf das Kommando 8.2 LAN_RAILCOM_GETDATA gesendet.

Sie wird aber auch ungefragt an Clients gesendet, wenn
- sich die entsprechenden RailCom-Daten tatsächlich verändert haben
- und der betreffende Client den entsprechenden Broadcast aktiviert hat
  (siehe **2.16** LAN_SET_BROADCASTFLAGS, Flag 0x00000004) und der betreffende Client die Lok-Adresse mit 4.1 LAN_X_GET_LOCO_INFO abonniert hat
- oder der betreffende Client den Broadcast 0x00040000 abonniert hat (d.h. RailCom-Daten aller Loks, für PC-Steuerungen).

Z21 an Client:

DataLen		**Header**		**Data**
0x11	0x00	**0x88**	0x00	**RailComDaten**

Die Struktur **RailComDaten** ist wie folgt aufgebaut (die 16-bit und 32-bit Werte sind little endian):

Byte Offset	**Typ**	**Name**	
0	UINT16	LocoAddress	Adresse des erkannten Decoders
2	UINT32	ReceiveCounter	Empfangszähler in Z21
6	UINT16	ErrorCounter	Empfangsfehlerzähler in Z21
8	UINT8	reserved	

9	UINT8	Options	Flags Bitmaske:
			`#define rcoSpeed1  0x01 // CH7 subindex 0`
			`#define rcoSpeed2  0x02 // CH7 subindex 1`
			`#define rcoQoS  0x04 // CH7 subindex 7`

10	UINT8	Speed	Geschwindigkeit 1 oder 2 (falls vom Decoder unterstützt)
11	UINT8	QoS	Quality of Service (falls vom Decoder unterstützt)
12	UINT8	reserved	

Die Struktur kann in Zukunft vergrößert werden, daher ist unbedingt bei der Auswertung DataLen zu berücksichtigen.

8.2 LAN_RAILCOM_GETDATA

RailCom-Daten von Z21 anfordern ab FW V1.29:

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0x89**	0x00	**Typ** 8 bit	**LocoAdress** 16 (bit little endian)

Typ 0x01 = RailCom-Daten für gegebene Lokadresse anfordern

LocoAddress Lokadresse
0=nächste Lok im Ringbuffer anfragen

Antwort von Z21:
Siehe oben *8.2* LAN_RAILCOM_DATACHANGED

9 LocoNet

Ab Z21 FW Version 1.20.

Wie bereits in der Einleitung erwähnt, kann die Z21 als **Ethernet/LocoNet Gateway** verwendet werden, wobei die Z21 gleichzeitig der LocoNet-Master ist, welcher die Refresh-Slots verwaltet und die DCC-Pakete generiert.

Damit der LAN-Client Meldungen vom LocoNet bekommt, muss er die entsprechenden LocoNet-Meldungen mittels ***2.16*** LAN_SET_BROADCASTFLAGS abonniert haben.

Meldungen, welche die Z21 am LocoNet-Bus empfängt, werden mit dem LAN-Header ***LAN_LOCONET_Z21_RX*** an den LAN-Client weitergeleitet.

Meldungen, welche die Z21 selber auf den LocoNet-Bus schreibt, werden ebenfalls mit dem LAN-Header ***LAN_LOCONET_Z21_TX*** an den LAN-Client weitergeleitet.

Mit den Z21-LAN-Befehl ***LAN_LOCONET_FROM_LAN*** kann der LAN-Client selber Meldungen auf den LocoNet-Bus schreiben. Sollte es gleichzeitig noch weitere LAN-Clients mit LocoNet-Abo geben, werden diese ebenfalls mit einer Meldung ***LAN_LOCONET_FROM_LAN*** benachrichtig werden. Nur der eigentliche Absender wird dabei nicht mehr benachrichtig.

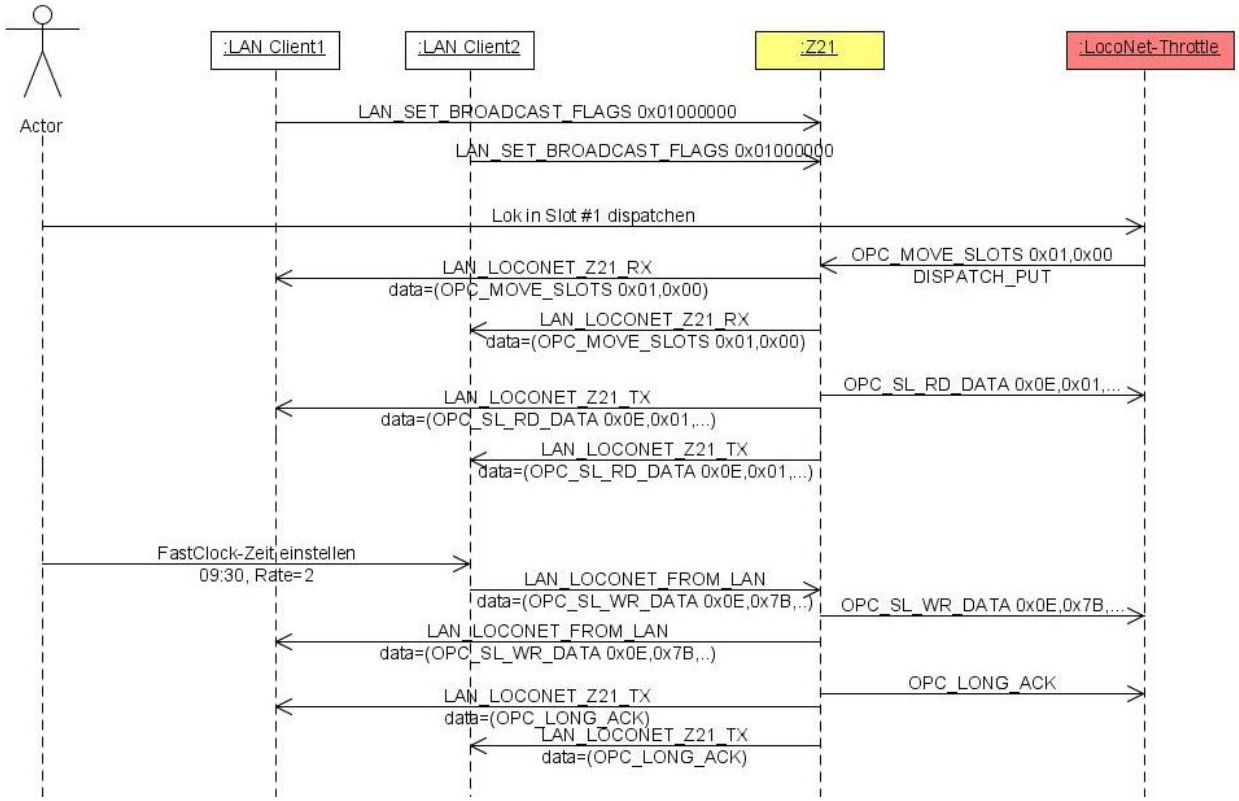

Abbildung 8 Beispiel Sequenz Ethernet/LocoNet Gateway

Dieses Beispiel zeigt, dass selbst bei trivialen Vorgängen am LocoNet-Bus gleichzeitig ein beträchtlicher Netzwerkverkehr am Ethernet bzw. WLAN entstehen kann.

Bitte beachten Sie, dass diese Ethernet/LocoNet Gateway Funktionalität in erster Line für PC-Steuerungen als Hilfsmittel zur Kommunikation mit LocoNet-Rückmelder etc. geschaffen worden ist.

Wägen Sie daher beim Abonnieren der LocoNet-Meldungen genau ab, ob die Broadcast Flags 0x02000000 (Loks) und 0x04000000 (Weichen) auch wirklich für Ihre Applikation unbedingt notwendig sind. Verwenden Sie vor allem zum konventionellen Fahren und Schalten nach wie vor soweit wie möglich die bereits beschriebenen LAN-Befehle aus den Kapiteln ***4*** Fahren, ***5*** Schalten und ***6*** Decoder CV Lesen und Schreiben.
Das eigentliche LocoNet-Protokoll wird in dieser Spezifikation nicht weiter beschrieben. Bitte wenden Sie sich dazu direkt an Digitrax oder ggf. an den Hersteller der jeweiligen LocoNet-Hardware, speziell wenn dieser das LocoNet-Protokoll für Konfiguration etc. eigenmächtig erweitert haben sollte.

9.1 LAN_LOCONET_Z21_RX

Ab Z21 FW Version 1.20.

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flags 0x01000000, 0x02000000 bzw. 0x04000000.
- und von der Z21 eine Meldung am LocoNet-Bus empfangen worden ist.

Z21 an Client:

DataLen		**Header**		**Data**
				LocoNet Meldung inkl. CKSUM
0x04+n	0x00	**0xA0**	0x00	n Bytes

9.2 LAN_LOCONET_Z21_TX

Ab Z21 FW Version 1.20.

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flags 0x01000000, 0x02000000 bzw. 0x04000000.
- und von der Z21 eine Meldung auf den LocoNet-Bus geschrieben worden ist.

Z21 an Client:

DataLen		**Header**		**Data**
				LocoNet Meldung inkl. CKSUM
0x04+n	0x00	**0xA1**	0x00	n Bytes

9.3 LAN_LOCONET_FROM_LAN

Ab Z21 FW Version 1.20.

Mit dieser Meldung kann ein LAN-Client eine Meldung auf den LocoNet-Bus schreiben.

Diese Meldung wird außerdem asynchron von der Z21 an einen Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flags 0x01000000, 0x02000000 bzw. 0x04000000.
- und ein **anderer** LAN-Client über die Z21 eine Meldung auf den LocoNet-Bus geschrieben hat.

LAN-Client an Z21, bzw. Z21 an LAN-Client:

DataLen		**Header**		**Data**
				LocoNet Meldung inkl. CKSUM
0x04+n	0x00	**0xA2**	0x00	n Bytes

9.3.1 DCC Binary State Control Instruction per LocoNet OPC_IMM_PACKET

Ab Z21 FW Version 1.42 wird zum Schalten von Binary States anstelle er wie folgt beschriebenen Methode das neue Kommando *4.3.3 LAN_X_SET_LOCO_BINARY_STATE* empfohlen.
Der nun folgende, inzwischen etwas veraltete Absatztext bleibt zwecks Vollständigkeit trotzdem bestehen:

Ab FW Version V1.25 können mittels LAN_LOCONET_FROM_LAN und dem LocoNet Befehl OPC_IMM_PACKET beliebige DCC Pakete am Gleisausgang generiert werden, darunter auch die Binary State Control Instruction (auch „F29…F32767“ genannt). Das gilt auch für die weiße z21, die zwar keine physikalische LocoNet Schnittstelle aufweist, aber sehr wohl über einen virtuellen LocoNet Stack verfügt.

Zum Aufbau des OPC_IMM_PACKET siehe LocoNet Spec (auch in personal edition zu Lernzwecken).
Zum Aufbau der Binary State Control Instruction siehe NMRA S-9.2.1 Abschnitt Feature Expansion Instruction.

9.4 LAN_LOCONET_DISPATCH_ADDR

Ab Z21 FW Version 1.20.

Eine Lok-Adresse zum LocoNet-Dispatch vorbereiten.

Mit dieser Meldung kann ein LAN-Client eine bestimmte Lok-Adresse für den LocoNet-Dispatch vorbereiten. Dies entspricht einem „DISPATCH_PUT“ und bedeutet, dass bei einem nächsten „DISPATCH_GET“ (ausgelöst durch Handregler) von der Z21 der zu dieser Lok-Adresse gehörende Slot zurück gemeldet wird. Gegebenenfalls wird dafür von der Z21 automatisch ein freier Slot belegt.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x06	0x00	**0xA3**	0x00	Lok-Adresse 16 bit (**little endian**)

Antwort von Z21:
Z21 FW Version < 1.22: keine
Z21 FW Version ≥ 1.22:

Z21 an Client:

DataLen		**Header**		**Data**	
0x07	0x00	**0xA3**	0x00	Lok-Adresse 16 bit (**little endian**)	**Ergebnis** 8 bit

Ergebnis 0 Der „DISPATCH_PUT“ für die gegebene Adresse ist fehlgeschlagen. Das kann passieren wenn z.B. die Z21 als LocoNet Slave betrieben wird und der LocoNet Master die Dispatch-Anforderung abgelehnt hat, weil diese Lok-Adresse bereits einem weiteren Handregler zugeteilt ist.

>0 Der „DISPATCH_PUT“ wurde erfolgreich ausgeführt. Die Lok-Adresse kann nun auf einem Handregler (z.B. FRED) übernommen werden. Der Wert von Result entspricht der aktuellen LocoNet Slot-Nummer für die gegebene Lok-Adresse.

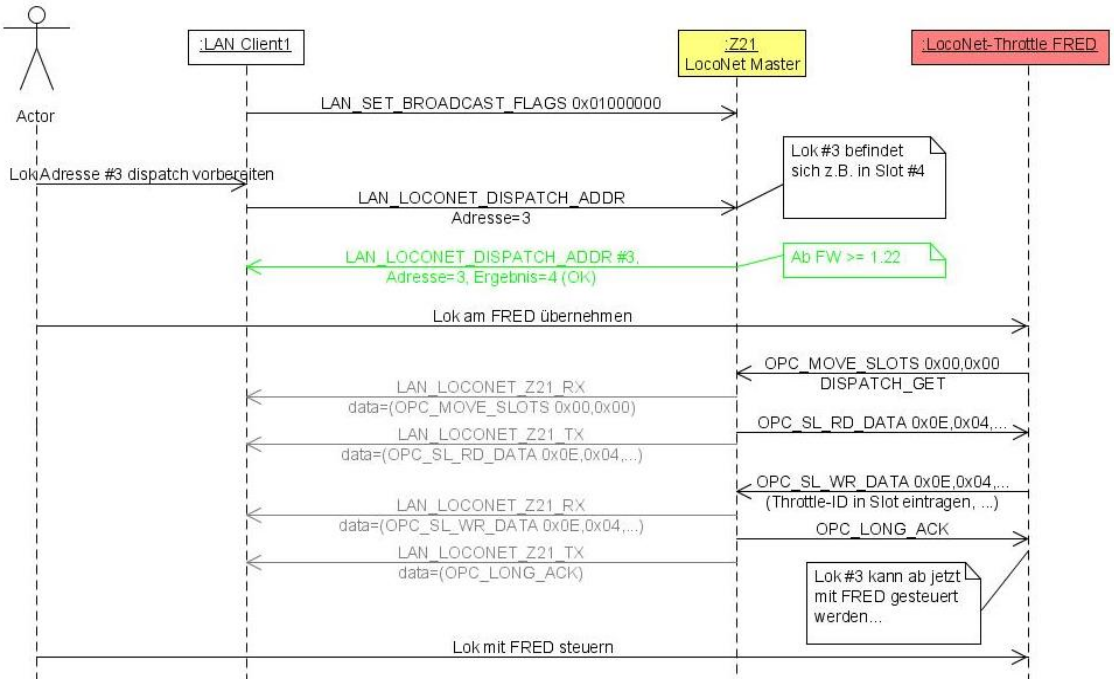

Abbildung 9 Beispiel Sequenz LocoNet Dispatch per LAN-Client

9.5 LAN_LOCONET_DETECTOR

Ab Z21 FW Version 1.22.

Falls eine Applikation im LAN Client einen LocoNet Gleisbesetztmelder unterstützen möchte, gibt es dafür zwei Möglichkeiten. Die erste wäre, mittels ***9.1*** *LAN_LOCONET_Z21_RX* die LocoNet-Pakete zu empfangen und die entsprechenden LocoNet-Meldungen selbständig zu verarbeiten. Das setzt aber eine entsprechend genaue Kenntnis des LocoNet Protokolls voraus.

Deswegen wurde die folgende Alternative geschaffen, mit denen man als LAN Client **sowohl** den Belegtstatus **abfragen** kann, **als auch** über eine Änderung des Belegtstatus **asynchron informiert** werden kann, ohne in die Tiefen des LocoNet-Protokolls einsteigen zu müssen.

Information: bitte beachten Sie folgenden wesentlichen Unterschied zwischen dem Roco Rückmeldemodul 10787 am R-BUS (siehe ***7*** Rückmelder – R-BUS) und LocoNet Gleisbesetztmeldern:

- 10787 basiert auf mechanisch betätigten Schaltkontakten, die pro Achse des darüber fahrenden Zugs geschlossen und wieder geöffnet werden können.

- LocoNet Gleisbesetztmelder basieren üblicherweise auf exakter Strommessung am überwachten Gleisabschnitt bzw. auf fortgeschrittene Technologien (Transponder, Infrarot, RailCom, ..), um den Besetzt-Zustand des Gleises zuverlässig ermitteln zu können. Während des Normalbetriebs wird im Idealfall nur eine Meldung bei der Änderung des Besetztzustands generiert.

Mit folgendem Kommando kann der Status eines oder mehrerer Gleisbesetztmelder abgefragt werden.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0xA4**	0x00	**Typ** 8 bit	Reportadresse 16 bit (**little endian**)

Typ

0x80 Abfrage mittels „Stationary Interrogate Request“ (**SIC**) gemäß Digitrax-Verfahren. Dieses Verfahren ist auch bei den Belegtmeldern von Blücher-Elektronik zu verwenden. Die Reportadresse ist hier 0 (don't care).

0x81 Abfrage mittels sogenannter **Reportadresse** für Uhlenbrock-Besetztmelder. Diese Reportadresse kann vom Anwender z.B. beim UB63320 über LNCV 17 im Besetztmelder konfiguriert werden. Der Default-Wert ist dort 1017.
Die Reportadresse wird beim Typ 0x81 nur zum Abfragen verwendet und ist **nicht** mit der **Rückmelderadresse** zu verwechseln.
Hinweis: Am LocoNet-Bus ist diese Abfrage über Weichenstellbefehle implementiert, deswegen ist der Wert gemäß LocoNet **um 1 dekrementiert** zu übergeben. Beispiel:
`0x07 0x00 0xA4 0x00 0x81 0xF8 0x03`
bedeutet: „fordere Status aller Besetztmelder mit Reportadresse 1017 an (Reportadresse = 1017 = **0x03F8** +1 = 1016 + 1)“

0x82 **Statusabfrage für LISSY ab Z21 FW Version 1.23**
Bei Uhlenbrock LISSY entspricht hier die Reportadresse allerdings wieder der Rückmelderadresse. Die Art der darauf folgenden Rückmeldung(en) hängt stark vom konfigurierten Betriebsmodus des LISSY-Empfängers ab. Über die umfangreichen Einstellmöglichkeiten des LISSY-Empfängers können Sie sich im LISSY-Handbuch informieren.

Bitte beachten Sie, dass bei einer einzigen Anfrage ggf. mehre Besetztmelder gleichzeitig angesprochen werden, und daher in der Regel mehrere Antworten zu erwarten sind. Abhängig vom Hersteller des Besetztmelders kann nach dieser Anforderung teilweise der Status ein und des selben Eingangs mehrmals gemeldet werden!

Antwort von Z21:

Z21 an Client:

DataLen		**Header**		**Data**		
0x07 + ***n***	0x00	**0xA4**	0x00	**Typ** 8 bit	**Rückmelderadresse** 16 bit (**little endian**)	**Info**[***n***]

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x08000000
- und die Z21 eine entsprechende Meldung von einem Gleisbesetztmelder empfangen hat, **aufgrund einer Statusänderung** an dessen Eingang, **oder aufgrund einer expliziten Abfrage** durch einen LAN Client mittels oben beschriebenen Kommandos.

Rückmelderadresse Jedem Eingang des Besetztmelders ist eine eigenen Rückmelderadresse zugeordnet, welche vom Anwender konfiguriert werden kann (z.B. bei Uhlenbrock und Blücher mittels LNCV) und den überwachten Block eindeutig beschreibt.

Info[*n*] Byte-Array; Inhalt und Länge ***n*** abhängig von **Typ**, siehe unten

Typ **0x01** Für Besetztmelder-Typen wie Uhlenbrock 63320 oder Blücher GBM16**XL**, welche nur den Status „belegt“ und „frei“ melden (LocoNet OPC_INPUT_REP, X=1).

n=1

Status des zur Rückmelderadresse gehörenden Eingangs steht in **Info[0]:**
Info[0]=**0** ... Sensor ist **LO** (“frei”)
Info[0]=**1** ... Sensor ist **HI** (“belegt”)

0x02 **Transponder Enters Block**
0x03 **Transponder Exits Block**
Für Besetztmelder Typen wie Blücher GBM16**XN** etc welche die Information (z.B. Lokadresse) über das Fahrzeug im Block an die Zentrale melden (mittels LocoNet OPC_MULTI_SENSE Transponding Encoding von Digitrax).
Es wird neben der Rückmelderadresse noch eine sogenannte Transponder-adresse übertragen. Die Transponderadresse identifiziert das im Block befindliche Fahrzeug. Im Fall vom GBM16**XN** ist das die Lok-Adresse, welche vom Belegtmelder mittels RailCom ermittelt worden ist.

n=2

Die Transponderadresse befindet sich in **Info[0]** und **Info[1]**, 16 Bit little endian:
Info[0] ... Transponderadresse Low Byte
Info[1] ... Transponderadresse High Byte

Anmerkung: aufgrund einer Schwäche der LocoNet Spezifikation gibt es beim Wertebereich von OPC_MULTI_SENSE einen Interpretationsspielraum, welcher die Hersteller der Belegtmelder im unklaren lässt.. Daher gibt es im Fall von GBM16**XN** nach unseren Erfahrungen folgendes zu beachten:

- Zur Rückmelderadresse muss +1 addiert werden, um auf jene Rückmelderadresse zu bekommen, welche im GBM16**XN** konfiguriert ist.
- Je nach Konfiguration des GBM16**XN** wird im Bit unter der Maske 0x1000 die Richtung des Fahrzeugs auf dem Gleis codiert. Diese Konfiguration wird von uns nicht empfohlen, das dieses Bit mit dem Adressraum für lange Lok-Adressen kollidiert!

0x10 LISSY Lokadresse ab Z21 FW 1.23.

Diese Meldung wird an den Z21 LAN Client geschickt, wenn ein Uhlenbrock LISSY-Empfänger ein Fahrzeug meldet, welches mit einem LISSY-Sender ausgerüstet ist, und der LISSY-Empfänger auf das „Übergabeformat (ÜF) Uhlenbrock“ (LNCV 15=1) konfiguriert ist. Weiters hängt diese Meldung stark vom konfigurierten Betriebsmodus (LNCV2, …) des Lissy-Empfängers ab. Siehe LISSY-Handbuch.

n=3

Die Lokadresse befindet sich in Info[0] und Info[1], 16 Bit little endian:
Info[0] ... Lokadresse Low Byte
Info[1] ... Lokadresse High Byte
Loks haben einen Wertebereich von 1..9999
Wagen haben einen Wertebereich von 10000 bis 16382

Info[2] ... Zusatzinformation mit folgenden Bits: 0 **DIR1 DIR0** 0 **K3 K2 K1 K0**
DIR1=0: **DIR0** ist zu ignorieren
DIR1=1: **DIR0**=0 ist vorwärts, **DIR0**=1 ist rückwärts
K3..K0: 4 Bit Klasseninformation, welche im LISSY-Sender hinterlegt worden ist.

Beispielkonfiguration für Lissy-Empfänger 68610:

LNCV	Wert	Kommentar
2	98	optionaler Modul-Reset: setzt alle LNCV auf 0, außer LNCV 0 und 1 (Adresse)
2	0	Grundfunktion: Auslesen der Lokdaten über Doppelsensor mit Richtungsinformation
15	1	Sende Übergabeformat Uhlenbrock ans LocoNet

0x11 LISSY Belegtzustand ab Z21 FW 1.23.

Diese Meldung wird an den Z21 LAN Client geschickt, wenn ein Uhlenbrock LISSY-Empfänger eine Blockzustandsmeldung im „Übergabeformat (ÜF) Uhlenbrock“ versendet. Siehe LISSY-Handbuch.

n=1

Status des zur Rückmelderadresse gehörenden Blocks steht in Info[0]:
Info[0]=**0** ... Block ist frei
Info[0]=**1** ... Block ist belegt

Beispielkonfiguration für Lissy-Empfänger 68610:

LNCV	Wert	Kommentar
2	98	optionaler Modul-Reset: setzt alle LNCV auf 0, außer LNCV 0 und 1 (Adresse)
2	22	**Automatikfunktion mit Blockzustandsmeldung: Aufenthaltsstelle zeitgesteuert**
3	2	Automatik aktiv in beiden Fahrtrichtungen
4	3	Aufenthaltszeit 3 Sekunden
10	2	Blockoption: Blockzustandsänderung auf „frei“ nach 2 Sekunden
15	1	Sende Übergabeformat (ÜF) Uhlenbrock ans LocoNet

0x12 LISSY Geschwindigkeit ab Z21 FW 1.23.
Diese Meldung wird an den Z21 LAN Client geschickt, wenn ein Uhlenbrock LISSY-Empfänger für die Geschwindigkeitsmessung konfiguriert ist.
Siehe LISSY-Handbuch.

n=2

Die Geschwindigkeit befindet sich in Info[0] und Info[1], 16 Bit little endian:
Info[0] ... Geschwindigkeit Low Byte
Info[1] ... Geschwindigkeit High Byte

Beispielkonfiguration für Lissy-Empfänger 68610:

LNCV	**Wert**	**Kommentar**
2	98	optionaler Modul-Reset: setzt alle LNCV auf 0, außer LNCV 0 und 1 (Adresse)
2	0	Grundfunktion: Auslesen der Lokdaten über Doppelsensor mit Richtungsinformation
14	**15660**	**Geschwindigkeit Skalierungsfaktor = 1566 (Maßstab H0) * 10mm (Sensorabstand)**
15	1	Sende Übergabeformat (ÜF) Uhlenbrock ans LocoNet

Anm. **Typ** wird je nach Bedarf in Zukunft noch um weitere IDs erweitert werden.

10 CAN

10.1 LAN_CAN_DETECTOR

Ab Z21 FW Version 1.30.

Der Roco CAN-Belegtmelder 10808 wird ab FW Version 1.30 unterstützt. Der Belegtmelder kann vom LAN Client auf vier verschiedene Weisen verwendet werden:

1. **R-BUS-Emulation**: der CAN-Belegtmelder wird in der Z21 Firmware als R-BUS-Melder an den LAN-Client weitergeleitet. Der LAN-Client kann den CAN-Belegtmelder verwenden, wie es in Kapitel **7** Rückmelder – R-BUS beschrieben ist.

2. **LocoNet-Emulation**: der CAN-Belegtmelder wird in der Z21 Firmware als LocoNet-Melder an den LAN-Client weitergeleitet. Der LAN-Client kann den CAN-Belegtmelder verwenden, wie es in Kapitel **9.5** LAN_LOCO_NET_DETECTOR beschrieben ist (Typ 0x01 „belegt/frei“ und die Lokadresse mittels Typ 0x02 und 0x03 „Transponder Enters Block, Transponder Exits Block“).

3. **LISSY-Emulation**: der CAN-Belegtmelder wird in der Z21 Firmware durch LISSY/Marco-Meldungen emuliert. Der LAN-Client kann den CAN-Belegtmelder verwenden, wie es in Kapitel **9.5** LAN_LOCO_NET_DETECTOR beschrieben ist (Typ 0x10 „Lokadresse“ und Typ 0x11 „Belegtzustand“).

4. Direkter Zugriff durch den Befehl **LAN_CAN_DETECTOR** (siehe unten).

Die Art der Emulation kann über das Z21 Maintenance Tool konfiguriert werden. Die Werkseinstellung ist: **R-BUS-Emulation=ein**, **LocoNet-Emulation=ein**, LISSY-Emulation=aus.

Die schnellste und bezüglich Speicher und Bandbreite schonendste Methode ist jedoch der direkte Zugriff durch den Befehl **LAN_CAN_DETECTOR 0xC4**. Das empfiehlt sich vor allem dann, wenn sehr viele CAN-Belegtmelder gleichzeitig verendet werden sollen. Mit folgendem Kommando kann der Status der CAN-Belegtmelder direkt abgefragt werden:

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0xC4**	0x00	**Typ** 8 bit	**CAN-NetworkID** 16 bit (little endian)

Typ **0x00** Abfrage des CAN-Belegtmelders mit der gegeben CAN-NetworkID.
Die CAN-NetworkID **0xD000** bedeutet „**alle CAN-Belegtmelder**“.
Beispiel:
`0x07 0x00 0xC4 0x00 0x00 0x00 0xD0`
bedeutet: „fordere Status aller CAN-Belegtmelder an“

Bitte beachten Sie, dass bei einer einzigen Anfrage mehrere CAN-Belegtmelder gleichzeitig angesprochen werden, und daher in der Regel mehrere Antworten zu erwarten sind. Es kann der Status ein und desselben Eingangs je nach Konfiguration der Emulation auch mehrmals gemeldet werden!

Antwort von Z21:

Z21 an Client:

DataLen		Header		Data					
0x0E	0x00	**0xC4**	0x00	**NId** 16 bit	**Addr** 16 bit	**Port** 8 bit	**Typ** 8 bit	**Value1** 16 bit	**Value2** 16 bit

Diese Meldung wird asynchron von der Z21 an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00080000
- und die Z21 eine entsprechende Meldung vom CAN-Belegtmelder empfangen hat, **aufgrund einer Statusänderung** an dessen Eingang, **oder aufgrund einer expliziten Abfrage** durch einen LAN-Client mittels oben beschriebenen Kommandos.

Alle 16 bit Werte sind little endian codiert.

NId Unveränderbare CAN-NetworkID des Belegtmelders.

Addr Konfigurierbare Moduladresse des Belegtmelders. Jeder CAN-Belegtmelder hat eine Moduladresse, welche vom Anwender eingestellt werden kann.

Port Eingang des CAN-Belegtmelders (0 bis 7)

Typ
0x01 **Belegtstatus** des Eingangs (frei, besetzt, Überlast)

0x11 1. und 2. erkannte **Lokadresse** am Eingang
0x12 3. und 4. erkannte **Lokadresse** am Eingang
…
0x1F 29. und 30. erkannte **Lokadresse** am Eingang

Der Wert von Value1 und Value2 hängt vom Typ ab.

Falls Typ = 0x01 (Belegtstatus):
Value1
0x0000 F**rei, ohne Spannung**
0x0100 **Frei, mit Spannung**
0x1000 **Besetzt, ohne Spannung**
0x1100 **Besetzt, mit Spannung**
0x1201 **Besetzt, Überlast 1**
0x1202 **Besetzt, Überlast 2**
0x1203 **Besetzt, Überlast 3**

Falls Typ = 0x11 bis 0x1F (RailCom Lokadressen):
Typ 0x11 bis 0x1F bilden eine Liste von Lokadressen.
Diese Fahrzeugliste endet mit Lokadresse=0.
Value1 Erste erkannte Lokadresse im Abschnitt inkl. Richtungsinformation.
0 = keine Lokadresse erkannt (z.B. bei nicht-RailCom-fähigem Decoder, oder keine Lok) bzw. Ende der Lokadressen-Liste
Value2 Zweite erkannte Lokadresse im Abschnitt inkl. Richtungsinformation.
0 = keine Lokadresse erkannt bzw. Ende der Lokadressen-Liste
In den obersten beiden Bits von Value1 bzw. Value2 ist die Richtungsinformation codiert:
0 x Keine Richtung erkannt
1 0 Fahrzeug ist vorwärts auf das Gleis gestellt worden
1 1 Fahrzeug ist rückwärts auf das Gleis gestellt worden
In den untersten 14 Bits steht die Lokadresse.

10.2 CAN Booster

Ab Z21 FW Version 1.41.

Die LAN-Befehle für das CAN Booster Management mit den Roco CAN-Booster 10806, 10807 und 10869 werden ab FW Version 1.41 unterstützt. Die folgenden Befehle funktionieren selbstverständlich nur, wenn diese Booster über den CAN-Bus (= nicht per B-BUS) mit der Z21 verbunden sind.

10.2.1 LAN_CAN_DEVICE_GET_DESCRIPTION

Bezeichnung aus CAN-Booster auslesen.

Im CAN-Booster kann vom Anwender ein Name (Freitext) hinterlegt werden, damit er das Gerät später leichter wieder identifizieren kann.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x06	0x00	**0xC8**	0x00	**NId** 16 bit

NId ist die CAN-NetworkID des gewünschten Boosters (16 bit little endian, 0xC101 bis 0xC1FF).
Siehe auch weiter unten ***10.2.3*** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD.

Antwort von Z21:

DataLen		**Header**		**Data**	
0x16	0x00	**0xC8**	0x00	**NId** 16 bit	UINT8 **Name**[16]

Name entspricht der hinterlegten Bezeichnung als nullterminierter String. Der String sollte gemäß **ISO 8859-1** (*Latin-1*) codiert sein.

Hinweis: nicht zwei LAN_CAN_BOOSTER_GET_DESCRIPTION schnell hintereinander senden, sondern zuerst die Antwort abwarten und erst danach den zweiten Request absenden!

Hinweis: Mit ***10.2.3*** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD bekommen Sie die NetworkIDs aller im System angeschlossenen CAN-Booster.

10.2.2 LAN_CAN_DEVICE_SET_DESCRIPTION

Bezeichnung im CAN- Booster überschreiben.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x16	0x00	**0xC9**	0x00	**NId** 16 bit	UINT8 **Name**[16]

NId ist die CAN-NetworkID des gewünschten Boosters (16 bit little endian, 0xC101 bis 0xC1FF).
Siehe auch weiter unten ***10.2.3*** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD.

Name entspricht der zu hinterlegenden Bezeichnung als nullterminierter String. Der String sollte gemäß **ISO 8859-1** (*Latin-1*) codiert sein. Füllen Sie ggf. den Rest von Data mit 0x00 auf. Nach dem 16. Zeichen wird in der Zentrale automatisch abgeschnitten.
Nicht erlaubte Zeichen sind das Anführungszeichen “ (0x22) und der Backslash \ (0x5C).

Antwort von Z21:
Keine

10.2.3 LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD

Systemzustand des CAN-Boosters an den Client melden. Diese Meldung kommt pro CAN-Booster und Booster-Ausgang circa einmal pro Sekunde.

Diese Meldung wird asynchron vom der Zentrale an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00020000
- und mindestens ein Booster über CAN mit der Zentrale verbunden ist.

Z21 an Client:

DataLen		**Header**		**Data**
0x0E	0x00	**0xCA**	0x00	**CANBoosterSystemState** (10 Bytes)

CANBoosterSystemState ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	**Typ**	**Name**	**Wert**	
0	UINT16	**NId**	0xC101 … 0xC1FF	**CAN-NetworkID** des Boosters
2	UINT16	Booster_OutputPort	1 2	Ausgang erste Endstufe Ausgang zweite Endstufe (nur 10807)
4	UINT16	Booster_State	bitmask	siehe unten
6	UINT16	Booster_VCCVoltage	mV	Spannung an der Endstufe
8	UINT16	Booster_Current	mA	Strom an der Endstufe

Bitmasken für **Booster_State**:

```
#define bsBgActive                  0x0001 // Bremsgenerator aktiv (ZCAN SSP)
#define bsShortCircuit              0x0020 // Kurzschluss an Endstufe (ZCAN UES)
#define bsTrackVoltageOff           0x0080 // Gleisspannung ist abgeschaltet (OFF)
#define bsRailComActive             0x0800 // RailCom-Cutout aktiv
```

Ab Booster FW Version V1.11 (Booster Manangement)**:**

```
#define bsOutputDisabled            0x0100 // Booster Ausgang deaktiviert (by user)
```

10.2.4 LAN_CAN_BOOSTER_SET_TRACKPOWER

Booster Management durch Anwender: CAN-Booster Gleisausgänge deaktivieren und reaktivieren.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x07	0x00	**0xCB**	0x00	**NId** 16 bit	**Power** 8 bit

NId ist die CAN-NetworkID des gewünschten Boosters (16 bit little endian, 0xC101 bis 0xC1FF).
Siehe auch weiter oben ***10.2.3*** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD.

Power 0x00 … alle Booster Gleisausgänge deaktivieren
0xFF … alle Booster Gleisausgänge reaktivieren
Zusätzlich **ab Z21 FW Version V1.42** und **Booster FW Version V1.11**:
0x10 … ersten Booster Gleisausgang deaktivieren
0x11 … ersten Booster Gleisausgang reaktivieren
0x20 … zweiten Booster Gleisausgang deaktivieren (Z21 dual BOOSTER)
0x22 … zweiten Booster Gleisausgang reaktivieren (Z21 dual BOOSTER)

Hinweis: Booster Gleisausgänge können nur dann wieder tatsächlich eingeschaltet werden, wenn auch die Zentrale Z21 ebenfalls eingeschaltet ist und ein gültiges Gleissignal an die CAN-Booster sendet.
Die Einstellungen des Booster Managements werden nicht persistent gespeichert.

Antwort von Z21:
Bei Änderung des CANBoosterSystemState ***10.2.3*** LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD an Clients mit Abo.

11 zLink

Die erstmals mit Z21 single BOOSTER eingeführte **zLink-Schnittstelle** erlaubt es, auch Endgeräte mit kleinerem Microcontroller ohne eigenem LAN oder WLAN Interface in sein eigenes Netzwerk zu integrieren.

Endgeräte mit zLink Schnittstelle sind mit Stand 06/2021:

- 10806 Z21 single BOOSTER
- 10807 Z21 dual BOOSTER
- 10869 Z21 XL BOOSTER
- 10836 Z21 switch DECODER
- 10837 Z21 signal DECODER

11.1 Adapter

An die zLink Schnittstelle der oben genannten Geräte kann ein Adapter angeschlossen werden, über welchen das Endgerät mit der Außenwelt kommunizieren kann.
Ein solcher Adapter ist der **10838 Z21 pro LINK**.

11.1.1 10838 Z21 pro LINK

Der **10838 Z21 pro LINK** verbindet als **Gateway** die **zLink Schnittstelle mit dem WLAN** und kann so für folgende Aufgaben verwendet werden:

1. **Konfiguration** des Endgeräts (per Tasten & Display, Z21 App, Z21 Maintenance Tool am PC)
2. **Firmware Update** des Endgeräts (per Z21 Updater App, Z21 Maintenance Tool am PC)
3. Steuerung des Endgeräts durch WLAN Clients über das **Z21 LAN Protokoll**

Letzterer Punkt 3 war zuerst lediglich als Testschnittstelle gedacht, doch bald war klar, dass sich hier interessante Möglichkeiten in Richtung dezentraler, per WLAN vernetzter Anlagen eröffnen. Es bedeutet aus technischer Sicht, dass (mit Rücksicht auf den eingeschränkten Speicher) im jeweiligen Endgerät ein auf die Aufgaben des Endgeräts zugeschnittener **Z21 Protokoll-Stack** implementiert ist. Siehe auch *Tabelle 1 Meldungen vom Client an Z21* und *Tabelle 2 Meldungen von Z21 an Clients.* Über das WLAN/zLink Gateway können nun - so wie an eine Zentrale – wie gewohnt Kommandos per UDP an das Endgerät geschickt werden. Zum Beispiel können die Gleisausgänge eines Boosters über das WLAN/zLink Gateway ein- und ausgeschaltet werden, oder der Booster-Systemstatus abgefragt werden. Es können über diese Schnittstelle am Z21 switch DECODER aber auch Weichen direkt geschaltet werden, bzw. ebenso beim Z21 signal DECODER Signale direkt angesteuert werden, und das sogar *ohne jegliche Verbindung zum Hauptgleis der Zentrale*. Die Decoder können über die zLink Schnittstelle per CV-Schreibbefehle sogar konfiguriert werden.

Es wurde versucht, dass es für den WLAN Client möglichst transparent bleibt, ob er nun mit einer Zentrale oder über das WLAN/zLink Gateway mit einem Endgerät kommuniziert. Angesichts der teilweise sehr kleinen CPU im Endgerät sollten aber folgende Punkte beachtet werden:

- Eingeschränkte Bandbreite: die effektive Transferrate sollte insgesamt deutlich unter 1024 Bytes/s bleiben. Mehr ist bei den aktuell verfügbaren Endgeräten auch weder notwendig noch sinnvoll.
- Geben Sie dem Endgerät genügend Zeit, die Befehle und Daten zu verarbeiten. Halten Sie daher zwischen zwei Befehlen eine Pause von mindestens 50 ms ein.
- Z21 pro LINK vorzugsweise im Client Mode verwenden
- Wenn möglich nur ein WLAN Client mit Z21 pro Link verbinden, maximal sind 4 Clients erlaubt

Ein Betrieb per UPD Broadcasts ist zwar möglich, es wird aber empfohlen, dies nur zum Auffinden der Geräte im Netzwerk zu verwenden (siehe weiter unten). Danach können die Endgeräte über ihren Hardware-Typ (LAN_GET_HWINFO) und Seriennummer (LAN_GET_SERIAL_NUMBER), sowie über die

IP-Adresse des jeweiligen Z21 pro LINK eindeutig zugeordnet werden. Zusätzlich kann der Benutzer in jedem Endgerät einen Namen (Freitext) hinterlegen, der ebenfalls angezeigt werden kann.

Ein Befehl, der vom Z21 pro LINK nicht an sein Endgerät durchreicht, sondern selbst verarbeitet und beantwortet, ist LAN_ZLINK_GET_HWINFO.

11.1.1.1 LAN_ZLINK_GET_HWINFO

Mit diesem Befehl können die Eigenschaften des Z21 pro LINK vom LAN Client abgefragt werden.

Wenn dieses Kommando als UDP Broadcast versendet wird, dann ist es möglich anhand der Antworten die im WLAN angemeldeten Z21 pro LINK aufzufinden und über ihre MAC Adresse und zugewiesener IP Adresse zu verwalten.

Anforderung an Z21 pro LINK:

DataLen		**Header**		**Data**
0x05	0x00	**0xE8**	0x00	**0x06**

Data[0] = **0x06** = ZLINK_MSG_TYPE_HW_INFO

Antwort von Z21 pro LINK:

DataLen		**Header**		**Data**	
0x3F	0x00	**0xE8**	0x00	**0x06**	**Z_Hw_Info** (58 Bytes)

Data[0] = **0x06** = ZLINK_MSG_TYPE_HW_INFO

Z_Hw_Info ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	**Typ**	**Name**		**Beispiel**
0	UINT16	**HwID**		401 (0x191)
2	UINT8	FW_Version_Major		1
3	UINT8	FW_Version_Minor		1
4	UINT16	FW_Version_Build		3217 (0xC91)
6	UINT8[18]	**MAC_Address**	string	„EC FA BC 4F 04 C6“
24	UINT8[33]	**Name**	string	„this_is_a_quite_long_device_name“
57	UINT8	Reserved	0x00	0

HwID
401 = 0x191 … Adapter **10838 Z21 pro LINK**

MAC_Address
MAC Adresse des Adapters als nullterminierte Zeichenkette, 8-bit ASCII.

Name
Vom Anwender konfigurierbarer Name des Adapters als nullterminierte Zeichenkette. Maximal 32 Zeichen zuzüglich 0x00 Ende Kennung, Codierung 8-bit ISO 8859-1 (*Latin-1*).
Ignorieren Sie alle Zeichen nach dem ersten 0x00.

Beispiel:

```
                                        3f 00 e8 00 06 91          ?.....
01 01 01 91 0c 45 43 20 46 41 20 42 43 20 34 46  .....EC FA BC 4F
20 30 34 20 43 36 00 74 68 69 73 5f 69 73 5f 61   04 C6.this_is_a
5f 71 75 69 74 65 5f 6c 6f 6e 67 5f 64 65 76 69  _quite_long_devi
63 65 5f 6e 61 6d 65 00 00                       ce_name..
```

HwID: **0x191** = 401 = 10838 Z21 pro LINK
FW Version: V1.1.3217
MAC Adresse: „EC FA BC 4F 04 C6“
Name: „this_is_a_quite_long_device_name”

11.2 Booster 10806, 10807 und 10869

Unterstützte Befehle siehe *Tabelle 1 Meldungen vom Client an Z21* und *Tabelle 2 Meldungen von Z21 an Clients*. Zusätzlich wurden für die Booster einige neue Befehle eingeführt, die nur für die Booster gültig sind.

11.2.1 LAN_BOOSTER_GET_DESCRIPTION

Bezeichnung aus Booster auslesen.

Im Booster kann vom Anwender ein Name (Freitext) hinterlegt werden, damit er das Gerät später leichter wieder identifizieren kann.

Anforderung an BOOSTER:

DataLen		Header		Data
0x04	0x00	**0xB8**	0x00	-

Antwort von BOOSTER:

DataLen		Header		Data
0x24	0x00	**0xB8**	0x00	UINT8 **Name**[32]

Name entspricht der hinterlegten Bezeichnung als nullterminierter String. Der String sollte gemäß **ISO 8859-1** (*Latin-1*) codiert sein und aus Gründen der Kompatibilität zum CAN-Bus nicht länger als 16 Zeichen sein.

Sonderfall: Name[0] kann 0xFF sein, falls im Gerät noch nie eine Bezeichnung abgelegt worden ist. Dieser Fall muss als Leerstring interpretiert werden.

Beispiel: ”Test”

```
                                  24 00 b8 00 54 65              $...Te
73 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00    st..............
00 00 00 00 00 00 00 00 00 00 00 00 00 00          ..............
```

11.2.2 LAN_BOOSTER_SET_DESCRIPTION

Bezeichnung im Booster überschreiben.

Anforderung an BOOSTER:

DataLen		Header		Data
0x24	0x00	**0xB9**	0x00	UINT8 **Name**[32]

Name entspricht der zu hinterlegenden Bezeichnung als nullterminierter String. Der String sollte gemäß **ISO 8859-1** (*Latin-1*) codiert sein und aus Gründen der Kompatibilität zum CAN-Bus nicht länger als 16 Zeichen sein. Füllen Sie den Rest von Data mit 0x00 auf.
Nicht erlaubte Zeichen sind das Anführungszeichen “ (0x22) und der Backslash \ (0x5C).

Antwort von BOOSTER:
Keine

11.2.3 LAN_BOOSTER_SYSTEMSTATE_GETDATA

Anfordern des aktuellen Systemzustandes.

Anforderung an BOOSTER:

DataLen		Header		Data
0x04	0x00	**0xBB**	0x00	-

Antwort von BOOSTER:
Siehe unten **11.2.4** LAN_BOOSTER_SYSTEMSTATE_DATACHANGED

11.2.4 LAN_BOOSTER_SYSTEMSTATE_DATACHANGED

Änderung des Systemzustandes vom Booster an den Client melden.

Diese Meldung wird asynchron vom Booster an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000100
- den Systemzustand explizit angefordert hat,
  siehe oben ***11.2.3*** LAN_BOOSTER_SYSTEMSTATE_GETDATA.

BOOSTER an Client:

DataLen		Header		Data
0x1C	0x00	**0xBA**	0x00	**BoosterSystemState** (24 Bytes)

BoosterSystemState ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	Typ	Name		
0	INT16	Booster_1_MainCurrent	mA	Strom an der 1. Endstufe
2	INT16	Booster_2_MainCurrent	mA	Strom an der 2. Endstufe
4	INT16	Booster_1_FilteredMainCurrent	mA	Geglätteter Strom 1. Endstufe
6	INT16	Booster_2_FilteredMainCurrent	mA	Geglätteter Strom 2. Endstufe
8	INT16	Booster_1_Temperature	°C	Temperatur der 1. Endstufe
10	INT16	Booster_2_Temperature	°C	Temperatur der 2. Endstufe
12	UINT16	SupplyVoltage	mV	Versorgungsspannung
14	UINT16	Booster_1_VCCVoltage	mV	Spannung an der 1. Endstufe
16	UINT16	Booster_2_VCCVoltage	mV	Spannung an der 2. Endstufe
18	UINT8	CentralState	bitmask	siehe unten
19	UINT8	CentralStateEx	bitmask	siehe unten
20	UINT8	CentralStateEx2	bitmask	siehe unten
21	UINT8	Reserved1		
22	UINT8	CentralStateEx3	bitmask	siehe unten
23	UINT8	Reserved2		

Bitmasken für **CentralState**:

```
#define csTrackVoltageOff            0x02   // Die Gleisspannung ist abgeschaltet
#define csConfigMode                 0x10   // Konfigurationsmodus aktiv
#define csCanConnected               0x20   // CAN Verbindung mit Zentrale Ok
```

Bitmasken für **CentralStateEx**:

```
#define cseHighTemperature              0x01   // zu hohe Temperatur
#define csePowerLost                    0x02   // zu geringe Eingangsspannung
#define cseBooster_1_ShortCircuit       0x04   // Kurzschluss an 1. Endstufe
#define cseBooster_2_ShortCircuit       0x08   // Kurzschluss an 2. Endstufe
#define cseRevPol                       0x10   // Fehler Versorgungsspannung
#define cseNoDCCInput                   0x80   // kein DCC-Eingangssignal vorhanden
```

Bitmasken für **CentralStateEx2**:

```
#define cse2Booster_1_RailComActive     0x01   // RailCom aktiv 1. Endstufe
#define cse2Booster_2_RailComActive     0x02   // RailCom aktiv 2. Endstufe
#define cse2Booster_1_MasterSettings    0x04   // CAN Autosettings Ok 1. Endstufe
#define cse2Booster_2_MasterSettings    0x08   // CAN Autosettings Ok 2. Endstufe
#define cse2Booster_1_BgActive          0x10   // Bremsgenerator aktiv 1. Endstufe
#define cse2Booster_2_BgActive          0x20   // Bremsgenerator aktiv 2. Endstufe
#define cse2Booster_1_RailComFwd        0x40   // RailCom Forwarding aktiv 1. Endstufe
#define cse2Booster_2_RailComFwd        0x80   // RailCom Forwarding aktiv 2. Endstufe
```

Bitmasken **fürCentralStateEx3**:

```
#define cse3Booster_1_OutputInverted    0x01   // 1. Endstufe invertiert (Autoinvert)
#define cse3Booster_2_OutputInverted    0x02   // 2. Endstufe invertiert (Autoinvert)
Ab Booster FW Version V1.11:
#define cse3Booster_1_OutputDisabled    0x10   // 1. Endstufe deaktiviert (by user)
#define cse3Booster_2_OutputDisabled    0x20   // 2. Endstufe deaktiviert (by user)
```

11.2.5 LAN_BOOSTER_SET_POWER

Ab Booster FW Version 1.11: Booster Management durch Anwender.

Falls hier am Booster *alle* Gleisausgänge deaktiviert bzw. reaktiviert werden, dann entspricht dieser Befehl defacto einem LAN_X_SET_TRACK_POWER_OFF bzw. LAN_X_SET_TRACK_POWER_ON an den Booster. Mit LAN_BOOSTER_SET_POWER ist es dagegen möglich, am 10807 Z21 dual BOOSTER auch einen *einzelnen* Gleisausgang gezielt aus- und wieder einzuschalten.

Anforderung an BOOSTER:

DataLen		**Header**		**Data**	
0x06	0x00	**0xB2**	0x00	**BoosterPort 8 bit**	**BoosterPortState 8 bit**

BoosterPort
0x01 … ersten Booster Gleisausgang auswählen
0x02 … zweiten Booster Gleisausgang auswählen (nur Z21 dual BOOSTER)
0x03 … alle Booster Gleisausgänge auswählen

BoosterPortState
0x00 … ausgewählte Booster Gleisausgänge deaktivieren
0x01 … ausgewählte Booster Gleisausgänge reaktivieren

Hinweis: Booster Gleisausgänge können nur dann wieder tatsächlich eingeschaltet werden, wenn auch die Zentrale Z21 ebenfalls eingeschaltet ist und ein gültiges Gleissignal an die CAN-Booster sendet. Die Einstellungen des Booster Managements werden nicht persistent gespeichert.

Antwort von BOOSTER:
Bei Änderung des BoosterSystemState **11.2.4** LAN_BOOSTER_SYSTEMSTATE_DATACHANGED an Clients mit Abo.

11.3 Decoder 10836 und 10837

Unterstützte Befehle siehe *Tabelle 1 Meldungen vom Client an Z21* und *Tabelle 2 Meldungen von Z21 an Clients*. Es wurden einige neue Befehle eingeführt, die nur für die Decoder gültig sind.

11.3.1 LAN_DECODER_GET_DESCRIPTION

Bezeichnung aus Decoder auslesen.

Im Decoder kann vom Anwender ein Name (Freitext) abgespeichert werden, um das Gerät später leichter wieder identifizieren zu können.

Anforderung an DECODER:

DataLen		**Header**		**Data**
0x04	0x00	**0xD8**	0x00	-

Antwort von DECODER:

DataLen		**Header**		**Data**
0x24	0x00	**0xD8**	0x00	UINT8 **Name**[32]

Name Codierung siehe 11.2.1 LAN_BOOSTER_GET_DESCRIPTION

11.3.2 LAN_DECODER_SET_DESCRIPTION

Bezeichnung im Decoder überschreiben.

Anforderung an DECODER:

DataLen		**Header**		**Data**
0x24	0x00	**0xD9**	0x00	UINT8 **Name**[32]

Name Codierung siehe 11.2.2 LAN_BOOSTER_SET_DESCRIPTION.

Antwort von DECODER:
Keine

11.3.3 LAN_DECODER_SYSTEMSTATE_GETDATA

Anfordern des aktuellen Systemzustandes.

Anforderung an DECODER:

DataLen		**Header**		**Data**
0x04	0x00	**0xDB**	0x00	-

Antwort von DECODER:
Siehe unten ***11.3.4*** LAN_DECODER_SYSTEMSTATE_DATACHANGED

11.3.4 LAN_DECODER_SYSTEMSTATE_DATACHANGED

Änderung des Systemzustandes vom Decoder an den Client melden.

Diese Meldung wird asynchron vom Decoder an den Client gemeldet, wenn dieser

- den entsprechenden Broadcast aktiviert hat,
  siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000100
- den Systemzustand explizit angefordert hat,
  siehe *11.3.3* LAN_DECODER_SYSTEMSTATE_GETDATA.

Wenn sich der Signaldecoder trotz Abo per Broadcastflags nach 4 Sekunden nicht meldet, weil sich z.B. kein Signalbegriff geändert hat, dann kann bei Bedarf auch gepollt werden.

Die Antworten von 10836 Z21 switch DECODER und 10837 Z21 signal DECODER **unterscheiden sich im Aufbau und Inhalt** und können Anhand von **DataLen** unterschieden werden.

11.3.4.1 SwitchDecoderSystemState

10836 Z21 switch DECODER an Client:

DataLen		**Header**		**Data**
0x30	0x00	**0xDA**	0x00	**SwitchDecoderSystemState** (44 Bytes)

SwitchDecoderSystemState ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	**Typ**	**Name**		
0	INT16	Current	mA	Strom
2	INT16	**FilteredCurrent**	mA	Geglätteter Strom
4	UINT16	Voltage	mV	Interne Spannung (3.3V)
6	UINT8	**CentralState**	bitmask	siehe unten
7	UINT8	**CentralStateEx**	bitmask	siehe unten
8	UINT8[8]	**OutputStates**[0..7]		Status pro Ausgang
16	UINT8[8]	**OutputConfig**[0..7]		Betriebsmodus pro Ausgang
24	UINT8[4]	**OutputDimm**[0..7]		Dimmwert pro Ausgang
32	UINT16	**Address**		Erste Decoderadresse
34	UINT16	**Address2**		Zweite Decoderadresse
36	UINT8[6]	Reserved1		
42	UINT8	**Dimmed**		1 Bit pro Ausgang
43	UINT8	Reserved2		

FilteredCurrent
Summe aus internem Stromverbrauch + Stromverbrauch an den Klemmen

Bitmasken für **CentralState**:

```
#define csEmergencyStop          0x01   // Not-Aus für Decoder
#define csTrackVoltageOff        0x02   // Die Gleisspannung ist abgeschaltet
#define csShortCircuit           0x04   // Kurzschluss erkannt
#define csConfigMode             0x10   // Konfigurationsmodus aktiv
```

Bitmasken für **CentralStateEx**:

```
#define csePowerLost             0x02   // zu geringe Eingangsspannung
#define cseRCN213                0x20   // Adressierung gem. RCN213
#define cseNoDCCInput            0x80   // kein DCC-Eingangssignal vorhanden
```

OutputState
Zustand des Ausgangs

```
#define oUnknown                0x00
#define oRedActive              0x11
#define oRedInactive            0x01
#define oGreenActive            0x12
#define oGreenInactive          0x02
```

OutputConfig
Betriebsmodus des Ausgangs

```
#define ocfgNormal              0      // Impulsbetrieb (default)
#define ocfgBlinker             1      // Wechselblinker
#define ocfgBlinkSm             2      // Wechselblinker mit Ein- und Ausblenden
#define ocfg10775               3      // Momentbetrieb wie 10775
#define ocfgK84                 4      // Dauerbetrieb (zB für Beleuchtung)
#define ocfgK84Sm               5      // Dauerbetrieb mit Ein- und Ausblenden
```

OutputDimm
Dimmwert
0 … Dimmung deaktiviert, entspricht daher voller Ausgangsleistung
1 bis 100 … minimal bis maximal mögliche Ausgangsleistung

Address
Einer Decoderadresse entsprechen 4 Weichennummern. Das heißt:
Erste Decoderadresse = 1 … Weichennummer 1 bis 4
Erste Decoderadresse = 2 … Weichennummer 5 bis 8
Erste Decoderadresse = 3 … Weichennummer 9 bis 12
usw.

Address2
Zweite Decoderadresse **= 0**: zweite Decoderadresse entspricht **automatisch** „Erste Decoderadresse **+ 1**“
ansonst gilt:
Zweite Decoderadresse = 1 … Weichennummer 1 bis 4
Zweite Decoderadresse = 2 … Weichennummer 5 bis 8
Zweite Decoderadresse = 3 … Weichennummer 9 bis 12
usw.

Dimmed
1 Bit pro Ausgangspaar:
0 … Ausgangspaar wird nicht gedimmt
1 … Ausgangspaar wird gedimmt oder sanftes Auf-/Abblenden ist konfiguriert
LSB = Ausgangspaar 1; MSB = Ausgangspaar 8

11.3.4.2 SignalDecoderSystemState

10837 Z21 signal DECODER an Client:

DataLen		Header		Data
0x2E	0x00	**0xDA**	0x00	**SignalDecoderSystemState** (42 Bytes)

SignalDecoderSystemState ist wie folgt aufgebaut (die 16-bit Werte sind little endian):

Byte Offset	Typ	Name		
0	INT16	Current	mA	0 / Reserviert
2	INT16	FilteredCurrent	mA	0 / Reserviert
4	UINT16	**Voltage**	mV	Spannung an den Klemmen
6	UINT8	**CentralState**	bitmask	siehe unten
7	UINT8	**CentralStateEx**	bitmask	siehe unten
8	UINT8[2]	**OutputStates**[0..1]		Ein/Aus-Status für Ausgänge A1…B8
10	UINT8[2]	**BlinkStates**[0..1]		Blink-Status für Ausgänge A1…B8
12	UINT8[4]	**SignalDccExt**[0..3]	DCCext	Aktueller Signalbegriff 1. bis 4. Signal
16	UINT8[4]	SignalCurrAsp[0..3]	Index	Aktueller Signalbegriff 1. bis 4. Signal
20	UINT8[3]	Reserved1		
23	UINT8	**SignalCount**	2, 3, 4	Anzahl der verwendeten Signale
24	UINT8[4]	**SignalConfig**[0..3]	Signal-ID	Signalkonfiguration 1. bis 4. Signal
28	UINT8[4]	SignalInitAsp[0..3]	Index	Initialisierung 1. bis 4. Signal
32	UINT16	**Address**		Erste Decoderadresse
34	UINT16[4]	Reserved2		

Bitmasken für **CentralState**:

```
#define csEmergencyStop          0x01   // Not-Aus für Decoder
#define csTrackVoltageOff        0x02   // Die Gleisspannung ist abgeschaltet
#define csShortCircuit           0x04   // Kurzschluss erkannt
#define csConfigMode             0x10   // Konfigurationsmodus aktiv
```

Bitmasken für **CentralStateEx**:

```
#define csePowerLost             0x02   // zu geringe Eingangsspannung
#define cseEEPromError           0x10   // EEPROM Schreib/Lesefehler
#define cseRCN213                0x20   // Adressierung gem. RCN213
#define cseNoDCCInput            0x80   // kein DCC-Eingangssignal vorhanden
```

OutputStates
OutputStates[0]: LSB = Ausgang A1; MSB = Ausgang A8
OutputStates[1]: LSB = Ausgang B1; MSB = Ausgang B8

BlinkStates
BlinkStates[0]: LSB = Ausgang A1; MSB = Ausgang A8
BlinkStates[1]: LSB = Ausgang B1; MSB = Ausgang B8

SignalDccExt und **SignalConfig**
SignalConfig definiert als **Signal-ID** eindeutig den Signaltyp.
SignalDccExt definiert als **DCCext**-Wert den aktuellen Signalbegriff zur gegeben Signal-ID.
Werte für Signal-ID und DCCext siehe https://www.z21.eu/de/produkte/z21-signal-decoder/signaltypen.

Address
Einer Decoderadresse entsprechen 4 Signaladressen.
Der Signaldecoder belegt 4 Decoderadressen hintereinander und somit 4x4=16 Signaladressen.
Erste Decoderadresse = 1 … Signaldecoder belegt Signaladressen 1 bis 16
Erste Decoderadresse = 2 … Signaldecoder belegt Signaladressen 5 bis 20
Erste Decoderadresse = 3 … Signaldecoder belegt Signaladressen 9 bis 24
usw.

12 Modellzeit

Ab Z21 FW Version 1.43.

Mit Firmware Version 1.43 wurden die Möglichkeiten der bereits bestehenden LocoNet Fastclock stark erweitert, sodass nun die beschleunigte Modellzeit der Z21 auch den Teilnehmern am Gleis, X-BUS und LAN zur Verfügung steht. Die Modellzeit kann vom Anwender bis zum Faktor ≤ 63 beschleunigt werden. Das erlaubt dem fortgeschrittenen Anwender dann ein Fahren nach Fahrplan, trotz verkürzter Streckenlängen zwischen den Modellbahnstationen.

Die Z21 besitzt allerdings keine Echtzeituhr, die nach dem Abschalten der Zentrale weiterlaufen würde. Daher beginnt die Modellzeit immer bei der gleichen, durch den Anwender einstellbaren Startzeit. Das Verhalten beim Nothalt und Kurzschluss, sowie die Ausgabe auf Gleis, LocoNet, X-BUS und IP Multicast sind ebenfalls vom Anwender konfigurierbar.

- DCC-Zeitmeldungen am Gleis siehe RCN-211.
- Bei LocoNet kann vom Endgerät etwa alle 70 bis 100 Sekunden der sogenannte Clock Slot 0x7B gepollt werden. Siehe auch LocoNet Spec (z.B. personal edition zu Lernzwecken).
- Am X-BUS erfolgt die Zeitmeldung gemäß XpressNet™ V4.0 einmal pro Modellminute.
- Auf der LAN Schnittstelle kann die Modellzeit optional auch per „MRclock“ Multicast versendet werden. Das erlaubt die Verwendung von MRclock Clients wie zum Beispiel die Android MRclock App zur Anzeige der Modellzeit. Falls aktiviert, wird der MRclock Multicast dann einmal pro Modellminute (aber mindestens dreimal pro echter Minute) an die Adresse 239.50.50.20, Port 2000 versendet.

Zusätzlich gibt es auch Z21 LAN Befehle für die Modellzeit, die nun wie folgt beschrieben werden.

12.1 LAN_FAST_CLOCK_CONTROL

12.1.1 Modellzeit lesen

Mit folgendem Befehl kann die aktuelle Modellzeit ausgelesen werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
0x07	0x00	**0xCC**	0x00	0x21	0x2A	0x0B

Antwort von Z21:
Siehe unten ***12.2*** LAN_FAST_CLOCK_DATA.

12.1.2 Modellzeit setzen

Mit folgendem Befehl kann die Rate und die aktuelle Modellzeit auf eine gewünschte Zeit gesetzt werden.

Anforderung an Z21:

DataLen		**Header**		**Data**					
0x0A	0x00	**0xCC**	0x00	0x24	0x2B	DDDhhhhh	00mmmmmm	00rrrrrr	XOR-Byte

DDD Der gewünschte Modellzeit-Wochentag in 3 Bits.
Wertebereich von 0 = Montag bis 6 = Sonntag.

hhhhh Die gewünschte Modellzeit-Stunde in 5 Bits, Wertebereich 0 bis 23.

mmmmmm Die gewünschte Modellzeit-Minute in 6 Bits, Wertebereich 0 bis 59.

rrrrrr Die gewünschte Modellzeit-Rate (Beschleunigungsfaktor) in 6 Bits.
Wertebereich von 0 bis 63:
(0 … Modellzeit bleibt stehen. Nicht empfohlen, besser ist: ***12.1.4*** Modellzeit anhalten)
1 … Echtzeit
2 … Modellzeit läuft doppelt so schnell
3 … Modellzeit läuft dreimal so schnell
usw.
Hinweis: Die Rate wird in der Z21 persistent gespeichert.

XOR-Byte XOR Prüfsumme über Data

Antwort von Z21:
12.2 LAN_FAST_CLOCK_DATA an Clients mit Abo.

12.1.3 Modellzeit starten

Mit folgendem Befehl kann die Modellzeituhr gestartet (d.h. fortgesetzt) werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
0x07	0x00	**0xCC**	0x00	0x21	0x2C	0x0D

Antwort von Z21:
12.2 LAN_FAST_CLOCK_DATA an Clients mit Abo.

Hinweis: Der geänderte Zustand „fcFastClockEnabled“ wird in der Z21 persistent gespeichert.

12.1.4 Modellzeit anhalten

Mit folgendem Befehl kann die Modellzeituhr angehalten werden.

Anforderung an Z21:

DataLen		**Header**		**Data**		
0x07	0x00	**0xCC**	0x00	0x21	0x2D	0x0C

Antwort von Z21:
12.2 LAN_FAST_CLOCK_DATA an Clients mit Abo.

Hinweis: Der der geänderte Zustand „not fcFastClockEnabled“ wird in der Z21 persistent gespeichert.

12.2 LAN_FAST_CLOCK_DATA

Aktuelle Modellzeit an Clients melden. Diese Meldung wird asynchron von der Z21 an die Clients gemeldet, wenn diese

- den entsprechenden Broadcast aktiviert haben, siehe ***2.16*** LAN_SET_BROADCASTFLAGS, Flag 0x00000010, oder
- die Modellzeit explizit angefordert haben, siehe oben ***12.1.1*** Modellzeit lesen.

Diese Meldung wird bei laufender Modellzeituhr asynchron von der Z21 ca. einmal pro Modellminute an die Clients mit Abo gemeldet, aber auch wenn die Modellzeit gestartet, angehalten oder neu gesetzt wird.

Die Zentrale darf Zeitmeldungen auch auslassen, z.B. wenn sie nicht im Normalbetrieb läuft. Übersprungene Zeitmeldungen müssen von den Clients toleriert werden und können ggf. anhand des Beschleunigungsfaktors aus der letzten Zeitmeldung von den Clients selbst weiter berechnet werden.

Z21 an Client:

DataLen		**Header**		**Data**
0x0C	0x00	**0xCD**	0x00	**FastClockTime** (8 Bytes)

FastClockTime ist wie folgt aufgebaut:

Byte Offset	**Typ**	**Name**	**Wert**	
0	UINT8		0x66	
1	UINT8		0x25	
2	UINT8	**DDDh hhhh**		Modellzeit Wochentag und Stunde
3	UINT8	00**mm mmmm**		Modellzeit Minute
4	UINT8	**SH**ss ssss		Modellzeit Sekunde, mit STOP- und HALT-Flag
5	UINT8	00**rr rrrr**		Modellzeit Rate
6	UINT8	**FcSettings**		Modellzeit Einstellungen Flags
7	UINT8	XOR-Byte		XOR Prüfsumme über Data

DDD Der aktuelle Modellzeit-Wochentag in 3 Bits. Wertebereich 0 = Montag bis 6 = Sonntag.

hhhhh Die aktuelle Modellzeit-Stunde in 5 Bits, Wertebereich 0 bis 23.

mmmmmm Die aktuelle Modellzeit-Minute in 6 Bits, Wertebereich 0 bis 59.

S **STOP-Flag:** die Modellzeit läuft nicht.
Ursache kann sein, dass die Fastclock nicht enabled ist, oder die Rate = 0 ist, etc.

H **HALT-Flag:** die Modellzeit wurde vorübergehend angehalten.
Ursache kann ein Nothalt oder ein Kurzschluss am Gleis sein.

ssssss Die aktuelle Modellzeit-Sekunde in 6 Bits, Wertebereich 0 bis 59.

rrrrrr Die aktuelle Modellzeit-Rate (Beschleunigungsfaktor) in 6 Bits, Wertebereich 0 bis 63:
(0 … Modellzeit kann nicht laufen)
1 … Echtzeit
2 … Modellzeit läuft doppelt so schnell
3 … Modellzeit läuft dreimal so schnell
usw.

FcSettings Die aktuellen persistenten Modellzeit-Einstellungen Flags, bitcodiert.
Bedeutung siehe ***12.3*** LAN_FAST_CLOCK_SETTINGS_GET.

12.3 LAN_FAST_CLOCK_SETTINGS_GET

Mit folgendem Befehl können die persistenten Modellzeit-Einstellungen ausgelesen werden.

Anforderung an Z21:

DataLen		Header		Data
0x05	0x00	**0xCE**	0x00	0x04

Antwort von Z21:

DataLen		Header		Data			
0x08	0x00	**0xCE**	0x00	**FcSettings**	**Rate**	**StartDDDhhhhh**	**StartMMMMMM**

Die einzelnen Parameter in Data sind jeweils 8 bit breit.

FcSettings Die Modellzeit-Einstellungen Flags, bitcodiert, siehe unten.

Rate Die gewünschte Modellzeit-Rate (Beschleunigungsfaktor).
Wertebereich von 0 bis 63:
(0 … Modellzeit kann nicht laufen, nicht empfohlen)
1 … Echtzeit
2 … Modellzeit läuft doppelt so schnell
3 … Modellzeit läuft dreimal so schnell
usw.

StartDDDhhhhh Default-Startzeit Wochentag und Stunde beim Einschalten der Zentrale.
DDD ist der Wochentag in 3 Bits. Wertebereich 0 = Montag bis 6 = Sonntag.
hhhhh ist die Stunde in 5 Bits, Wertebereich 0 bis 23.

StartMMMMMM Default-Startzeit Minute beim Einschalten der Zentrale.
MMMMMM ist die Minute in 6 Bits, Wertebereich 0 bis 59

Bitmasken für **FcSettings**:

```
#define fcFastClockLocoNetEn           0x01 // Ausgabe am LocoNet (polled) aktivieren
#define fcFastClockXBUSEn              0x02 // Broadcast am XBUS aktivieren
//                                     0x04 // reserved
#define fcFastClockDCCEn               0x08 // DCC Broadcast am Gleis aktivieren
#define fcFastClockMRclockEn           0x10 // Multicast an MRclock clients aktivieren
//                                     0x20 // reserved
#define fcFastClockEmergenyHaltEn      0x40 // Modellzeit beim Nothalt autom. anhalten
#define fcFastClockEnabled             0x80 // Fastclock ist aktiviert
```

Alle hier als „reserved“ deklarierten Bits sind für zukünftige Erweiterungen reserviert und sollen weder ausgewertet noch verändert werden.

Das Flag **`fcFastClockEmergenyHaltEn`** bewirkt, dass während eines Nothalts oder Kurzschlusses die Modellzeit automatisch pausiert wird.

Das Flag **`fcFastClockEnabled`** ist *das* „Enable-Flag“ für die Modellzeit. Es wird so wie **Rate** aber nicht nur über den weiter unten beschriebenen Befehl LAN_FAST_CLOCK_SETTINGS_SET verändert, sondern indirekt auch über LAN_FAST_CLOCK_CONTROL durch das Starten oder Anhalten der Modellzeit.

Die **Werkseinstellung** ist für FcSettings=0x4F, Rate=1, StartDDDhhhhh=0 und StartMMMMMM=0.

12.4 LAN_FAST_CLOCK_SETTINGS_SET

Mit folgenden Befehlen können die persistenten Modellzeit-Einstellungen gezielt überschrieben werden. Die einzelnen Parameter in Data sind jeweils 8 bit breit.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x05	0x00	**0xCF**	0x00	**FcSettings**

Damit werden nur die Fastclock-Einstellungen **FcSettings** überschrieben.

Anforderung an Z21:

DataLen		**Header**		**Data**	
0x06	0x00	**0xCF**	0x00	**FcSettings**	**Rate**

Damit werden nur die Fastclock-Einstellungen **FcSettings** und die **Rate** überschrieben.

Anforderung an Z21:

DataLen		**Header**		**Data**			
0x08	0x00	**0xCF**	0x00	**FcSettings**	**Rate**	**StartDDDhhhhh**	**StartMMMMMM**

Damit werden die Fastclock-Einstellungen ***FcSettings***, die **Rate** sowie die **Default-Startzeit** überschrieben. Die Default-Startzeit ist jene Uhrzeit, die beim Einschalten der Zentrale übernommen wird.

Beschreibung der einzelnen Felder siehe oben ***12.3*** LAN_FAST_CLOCK_SETTINGS_GET.

Antwort von Z21:
Keine.

Anhang A – Befehlsübersicht

Client an Z21

Diese Meldungen können von einem Client an eine Z21 oder an ein zLink-Gerät gesendet werden.

Header	Daten			Name	LAN		zLink	
	X-Header	DB0	Parameter		Z21 Z21 XL	z21 z21start	Booster 10806 10807 10869	Decoder 10836 10837
0x10	-			LAN_GET_SERIAL_NUMBER	✓	✓	✓	✓
0x18	-			LAN_GET_CODE	✓	✓	×	×
0x1A	-			LAN_GET_HWINFO	✓	✓	✓	✓
0x30	-			LAN_LOGOFF	✓	✓	✓	✓
0x40	0x21	0x21	-	LAN_X_GET_VERSION	✓	✓	✓	✓
0x40	0x21	0x24	-	LAN_X_GET_STATUS	✓	✓	✓	✓
0x40	0x21	0x80	-	LAN_X_SET_TRACK_POWER_OFF	✓	✓	✓	✓
0x40	0x21	0x81	-	LAN_X_SET_TRACK_POWER_ON	✓	✓	✓	✓ (4)
0x40	0x22	0x11	Register	LAN_X_DCC_READ_REGISTER	✓	✓	×	×
0x40	0x23	0x11	CV-Adresse	LAN_X_CV_READ	✓	✓	×	✓
0x40	0x23	0x12	Register, Wert	LAN_X_DCC_WRITE_REGISTER	✓	✓	×	×
0x40	0x24	0x12	CV-Adresse, Wert	LAN_X_CV_WRITE	✓	✓	×	✓
0x40	0x24	0xFF	Register, Wert	LAN_X_MM_WRITE_BYTE	✓	✓	×	×
0x40	0x43	Weichen-Adresse		LAN_X_GET_TURNOUT_INFO	✓	✓	×	✓
0x40	0x44	Zubehördecoder-Adresse		LAN_X_GET_EXT_ACCESSORY_INFO	✓	✓	×	✓ (3)
0x40	0x53	Weichen-Adresse, Schaltbefehl		LAN_X_SET_TURNOUT	✓	✓ (1)	×	✓
0x40	0x54	Zubehördecoder-Adresse, Zustand		LAN_X_SET_EXT_ACCESSORY	✓	✓ (1)	×	✓
0x40	0x80	-		LAN_X_SET_STOP	✓	✓	×	✓ (5)
0x40	0x92	Lok-Adresse		LAN_X_SET_LOCO_E_STOP	✓	✓	×	×
0x40	0xE3	0x44	Lok-Adresse	LAN_X_PURGE_LOCO	✓	✓	×	×
0x40	0xE3	0xF0	Lok-Adresse	LAN_X_GET_LOCO_INFO	✓	✓	×	×
0x40	0xE4	0x1s	Lok-Adresse,Geschwindigkeit	LAN_X_SET_LOCO_DRIVE	✓	✓ (1)	×	×
0x40	0xE4	0xF8	Lok-Adresse, Funktion	LAN_X_SET_LOCO_FUNCTION	✓	✓ (1)	×	×
0x40	0xE4	Group	Lok-Adresse, Funktionsgruppe	LAN_X_SET_LOCO_FUNCTION_GROUP	✓	✓ (1)	×	×
0x40	0xE5	0x5F	Lok-Adresse, Binärzustand	LAN_X_SET_LOCO_BINARY_STATE	✓	✓	×	×
0x40	0xE6	0x30	POM-Param, Option 0xEC	LAN_X_CV_POM_WRITE_BYTE	✓	✓	×	✓
0x40	0xE6	0x30	POM-Param, Option 0xE8	LAN_X_CV_POM_WRITE_BIT	✓	✓	×	×
0x40	0xE6	0x30	POM-Param, Option 0xE4	LAN_X_CV_POM_READ_BYTE	✓	✓	×	✓
0x40	0xE6	0x31	POM-Param, Option 0xEC	LAN_X_CV_POM_ACCESSORY_WRITE_BYTE	✓	✓	×	✓
0x40	0xE6	0x31	POM-Param, Option 0xE8	LAN_X_CV_POM_ACCESSORY_WRITE_BIT	✓	✓	×	×
0x40	0xE6	0x31	POM-Param, Option 0xE4	LAN_X_CV_POM_ACCESSORY_READ_BYTE	✓	✓	×	✓
0x40	0xF1	0x0A	-	LAN_X_GET_FIRMWARE_VERSION	✓	✓	✓	✓
0x50	Broadcast-Flags			LAN_SET_BROADCASTFLAGS	✓	✓	✓	✓
0x51	-			LAN_GET_BROADCASTFLAGS	✓	✓	✓	✓
0x60	Lok-Adresse			LAN_GET_LOCOMODE	✓	✓	×	×
0x61	Lok-Adresse, Modus			LAN_SET_LOCOMODE	✓	✓	×	×
0x70	Funktionsdecoder-Adresse			LAN_GET_TURNOUTMODE	✓	✓	×	×
0x71	Funktionsdecoder-Adresse, Modus			LAN_SET_TURNOUTMODE	✓	✓	×	×
0x81	Gruppenindex			LAN_RMBUS_GETDATA	✓	✓	×	×
0x82	Adresse			LAN_RMBUS_PROGRAMMODULE	✓	✓	×	×
0x85	-			LAN_SYSTEMSTATE_GETDATA	✓	✓	×	×
0x89	Adresse			LAN_RAILCOM_GETDATA	✓	✓	✓	×
0xA2	LocoNet-Meldung			LAN_LOCONET_FROM_LAN	✓	✓ (1)(2)	×	×
0xA3	Lok-Adresse			LAN_LOCONET_DISPATCH_ADDR	✓	×	×	×
0xA4	Typ, Reportadresse			LAN_LOCONET_DETECTOR	✓	✓ (2)	×	×
0xC4	Typ, NId			LAN_CAN_DETECTOR	✓	×	×	×
0xC8	NetID			LAN_CAN_DEVICE_GET_DESCRIPTION	✓	×	×	×
0xC9	NetID, Name			LAN_CAN_DEVICE_SET_DESCRIPTION	✓	×	×	×
0xCB	NetID, PowerState			LAN_CAN_BOOSTER_SET_TRACKPOWER	✓	×	×	×
0xCC	Fastclock Start/Stop/Get/Set Command			LAN_FAST_CLOCK_CONTROL	✓	✓	×	×
0xCE	Len			LAN_FAST_CLOCK_SETTINGS_GET	✓	✓	×	×
0xCF	Fastclock Settings			LAN_FAST_CLOCK_SETTINGS_SET	✓	✓	×	×
0xB2	BoosterPort, BoosterPowerState			LAN_BOOSTER_SET_POWER	×	×	✓	×
0xB8	-			LAN_BOOSTER_GET_DESCRIPTION	×	×	✓	×
0xB9	String			LAN_BOOSTER_SET_DESCRIPTION	×	×	✓	×
0xBB	-			LAN_BOOSTER_SYSTEMSTATE_GETDATA	×	×	✓	×
0xD8	-			LAN_DECODER_GET_DESCRIPTION	×	×	×	✓
0xD9	String			LAN_DECODER_SET_DESCRIPTION	×	×	×	✓
0xDB	-			LAN_DECODER_SYSTEMSTATE_GETDATA	×	×	×	✓
0xE8	0x06	-		LAN_ZLINK_GET_HWINFO	×	×	✓ (6)	✓ (6)

Tabelle 1 Meldungen vom Client an Z21

(1) z21start: nur mit Freischaltcode (Artikelnummer 10814 oder 10818)
(2) z21, z21start: virtueller LocoNet-Stack (z.B. bei GBM16XN mit XPN-Interface)
(3) ab Decoder FW V1.11
(4) Decoder: Signallampen wieder einschalten (nur 10837)
(5) Decoder: zeigt Haltebegriff, wenn in CV38 das zweite Bit (0x02) gesetzt ist (nur 10837)
(6) Wird vom 10838 Z21 pro LINK beantwortet, nicht vom Endgerät (Booster oder Decoder)

Z21 an Client

Diese Meldungen können von einer Z21 oder von einem zLink-Gerät an einen Client gesendet werden.

Header	Daten			Name	LAN		zLink	
	X-Header	DB0	Daten		Z21 Z21 XL	z21 z21start	Booster 10806 10807 10869	Decoder 10836 10837
0x10	Serialnumber			Antwort auf LAN_GET_SERIAL_NUMBER	✓	✓	✓	✓
0x18	Code			Antwort auf LAN_GET_CODE	✓	✓	×	×
0x1A	HWType, FW Version (BCD)			Antwort auf LAN_GET_HWINFO	✓	✓	✓	✓
0x40	0x43	Weichen-Information		LAN_X_TURNOUT_INFO	✓	✓ (1)	×	✓
0x40	0x44	Zubehördecoder-Information		LAN_X_EXT_ACCESSORY_INFO	✓	✓ (1)	×	✓ (3)
0x40	0x61	0x00	-	LAN_X_BC_TRACK_POWER_OFF	✓	✓	✓	×
0x40	0x61	0x01	-	LAN_X_BC_TRACK_POWER_ON	✓	✓	✓	×
0x40	0x61	0x02	-	LAN_X_BC_PROGRAMMING_MODE	✓	✓	×	×
0x40	0x61	0x08	-	LAN_X_BC_TRACK_SHORT_CIRCUIT	✓	✓	× (4)	× (4)
0x40	0x61	0x12	-	LAN_X_CV_NACK_SC	✓	✓	×	×
0x40	0x61	0x13	-	LAN_X_CV_NACK	✓	✓	×	✓
0x40	0x61	0x82	-	LAN_X_UNKNOWN_COMMAND	✓	✓	✓	✓
0x40	0x62	0x22	Status	LAN_X_STATUS_CHANGED	✓	✓	✓	✓
0x40	0x63	0x21	XBus Version, ID	Antwort auf LAN_X_GET_VERSION	✓	✓	✓	✓
0x40	0x64	0x14	CV-Result	LAN_X_CV_RESULT	✓	✓	×	✓
0x40	0x81	-		LAN_X_BC_STOPPED	✓	✓	×	×
0x40	0xEF	Lok-Information		LAN_X_LOCO_INFO	✓	✓ (1)	×	×
0x40	0xF3	0x0A	Version (BCD)	Antwort auf LAN_X_GET_FIRMWARE_VERSION	✓	✓	✓	✓
0x51	Broadcast-Flags			Antwort auf LAN_GET_BROADCASTFLAGS	✓	✓	✓	✓
0x60	Lok-Adresse, Modus			Antwort auf LAN_GET_LOCOMODE	✓	✓	×	×
0x70	Funktionsdecoder-Adresse, Modus			Antwort auf LAN_GET_TURNOUTMODE	✓	✓	×	×
0x80	Gruppenindex, Rückmelder-Status			LAN_RMBUS_DATACHANGED	✓	✓	×	×
0x84	SystemState			LAN_SYSTEMSTATE_DATACHANGED	✓	✓	×	×
0x88	RailCom Daten			LAN_RAILCOM_DATACHANGED	✓	✓	✓	×
0xA0	LocoNet-Meldung			LAN_LOCONET_Z21_RX	✓	×	×	×
0xA1	LocoNet-Meldung			LAN_LOCONET_Z21_TX	✓	✓ (2)	×	×
0xA2	LocoNet-Meldung			LAN_LOCONET_FROM_LAN	✓	✓ (2)	×	×
0xA3	Lok-Adresse, Ergebnis			LAN_LOCONET_DISPATCH_ADDR	✓	×	×	×
0xA4	Typ, Rückmelderadresse, Info			LAN_LOCONET_DETECTOR	✓	✓ (2)	×	×
0xC4	Belegtmeldung			LAN_CAN_DETECTOR	✓	×	×	×
0xC8	NetID, Name			Antwort LAN_CAN_DEVICE_GET_DESCRIPTION	✓	×	×	×
0xCA	CANBoosterSystemState			LAN_CAN_BOOSTER_SYSTEMSTATE_CHGD	✓	×	×	×
0xCD	Fastclock Time			LAN_FAST_CLOCK_DATA	✓	✓	×	×
0xCE	Fastclock Settings			LAN_FAST_CLOCK_SETTINGS_GET	✓	✓	×	×
0xB8	String			Antwort auf LAN_BOOSTER_GET_DESCRIPTION	×	×	✓	×
0xBA	BoosterSystemState			LAN_BOOSTER_SYSTEMSTATE_DATACHANGED	×	×	✓	×
0xD8	String			Antwort auf LAN_DECODER_GET_DESCRIPTION	×	×	×	✓
0xDA	DecoderSystemState			LAN_DECODER_SYSTEMSTATE_DATACHANGED	×	×	×	✓
0xE8	0x06	Z_Hw_Info		Antwort auf LAN_ZLINK_GET_HWINFO	×	×	✓ (5)	✓ (5)

Tabelle 2 Meldungen von Z21 an Clients

(1) z21start: vollfunktionsfähig nur mit Freischaltcode (Artikelnummer 10814 oder 10818)
(2) z21, z21start: virtueller LocoNet-Stack (z.B. bei GBM16XN mit XPN-Interface)
(3) ab Decoder FW V1.11
(4) Kurzschluss wird im entsprechenden Booster/Decoder-SystemState gemeldet
(5) Wird vom 10838 Z21 pro LINK beantwortet, nicht vom Endgerät (Booster oder Decoder)

Abbildungsverzeichnis

Tabellenverzeichnis