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

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.

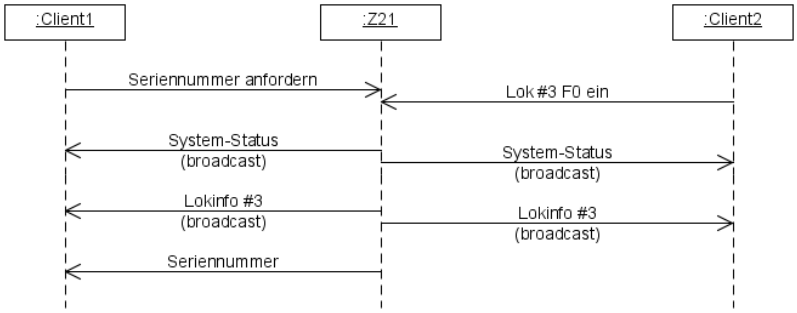

Abbildung 1 Beispiel Sequenz Kommunikation

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

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	**0x30**	**0x12**	0x60

DB1 … X-Bus Version 3.0
DB2 … ID der Zentrale, 0x12 = Z21

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 werden automatisch gesendet *(zukünftige Erweiterung)*~~

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***

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, 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	reserved		

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
```

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   // Z21 (Hardware-Variante ab 2012)
#define D_HWT_Z21_NEW      0x00000201   // Z21 (Hardware-Variante ab 2013)
#define D_HWT_SMARTRAIL    0x00000202   // SmartRail (ab 2012)
#define D_HWT_z21_SMALL    0x00000203   // z21 Starterset-Variante (ab 2013)
```

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)

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.

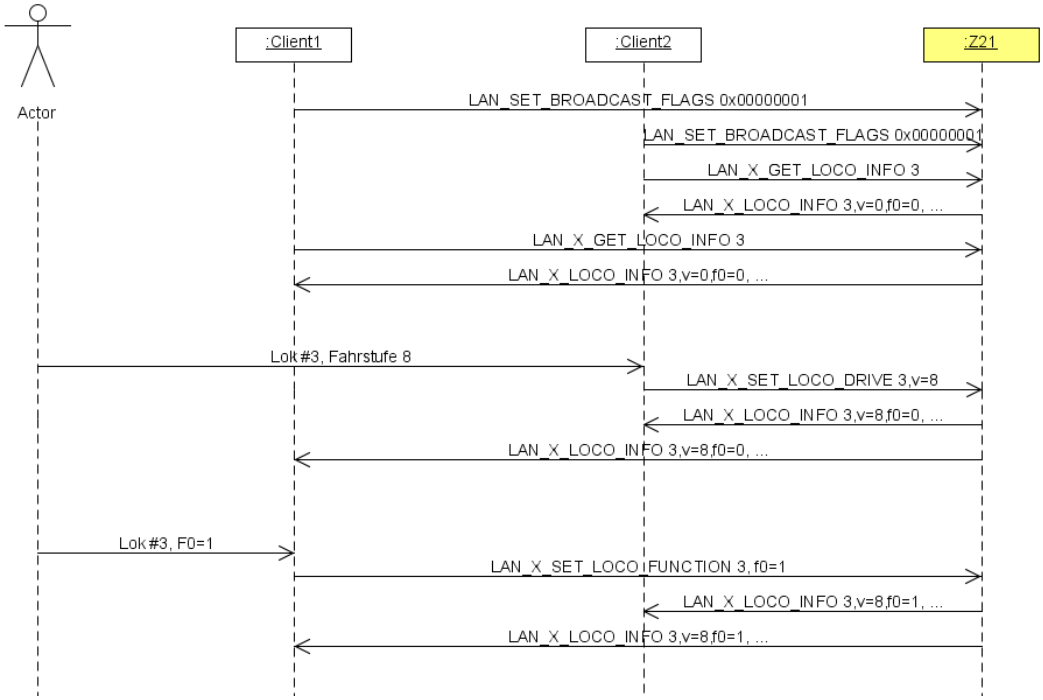

Abbildung 2 Beispiel Sequenz Lok-Steuerung

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“.

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 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**	**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** S=0 oder 1: 14 Fahrstufen
S=2: 28 Fahrstufen
S=3: 128 Fahrstufen

RVVVVVVV R ... Richtung: 1=vorwärts
V ... Geschwindigkeit: abhängig von den Fahrstufen, Codierung wie bei DCC

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

Anmerkung: die Fahrstufen werden automatisch in der Zentrale persistent abgespeichert.

4.3 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**	**TTNNNNN**	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, 0=F0 (Licht), 1=F1 usw.

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

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
- 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***.

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	0000**BKKK**	**B**=1 ... die Lok wird von einem anderen Gerät (X-BUS Handregler) gesteuert („*besetzt*“) **KKK** ... Fahrstufeninformation: 0=14, 2=28, 4=128
DB3	**RVVVVVVV**	**R** ... Richtung: 1=vorwärts **V** ... Geschwindigkeit: abhängig von Fahrstufen, Codierung wie bei DCC
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)
DB***n***		optional, für zukünftige Erweiterungen

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

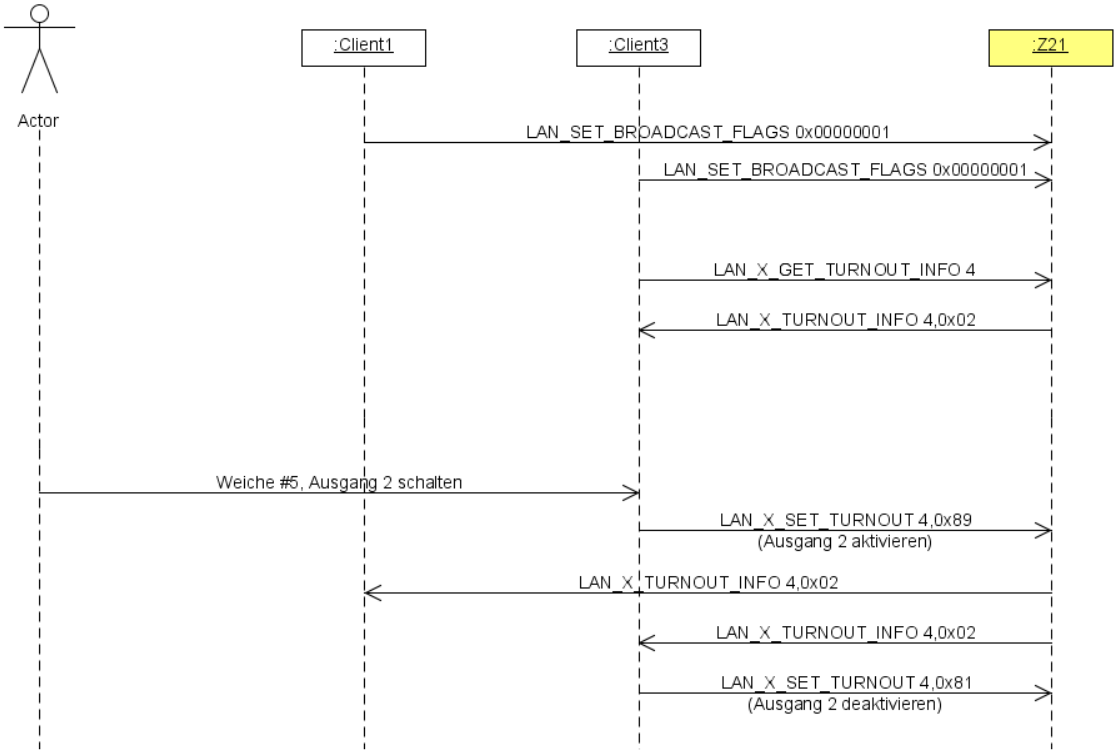

Abbildung 5 Beispiel Sequenz Weiche schalten

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.

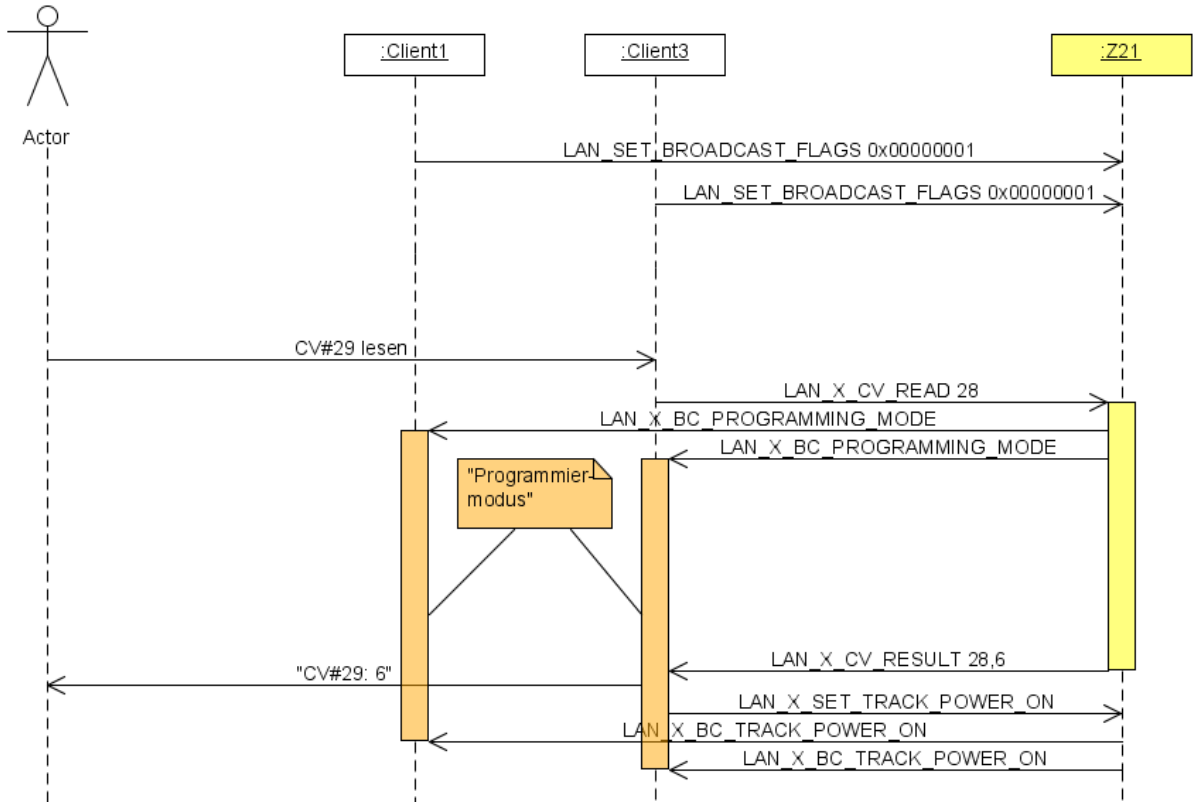

Abbildung 6 Beispiel Sequenz CV Lesen

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 normalem 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 normalem 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 normalem 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	**111010MM**	**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 normalem 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 normalem 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 normalem 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	**111010MM**	**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**“

7 Rückmelder – R-BUS

Die Rückmeldemodule (Bestellnummer 10787) 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.

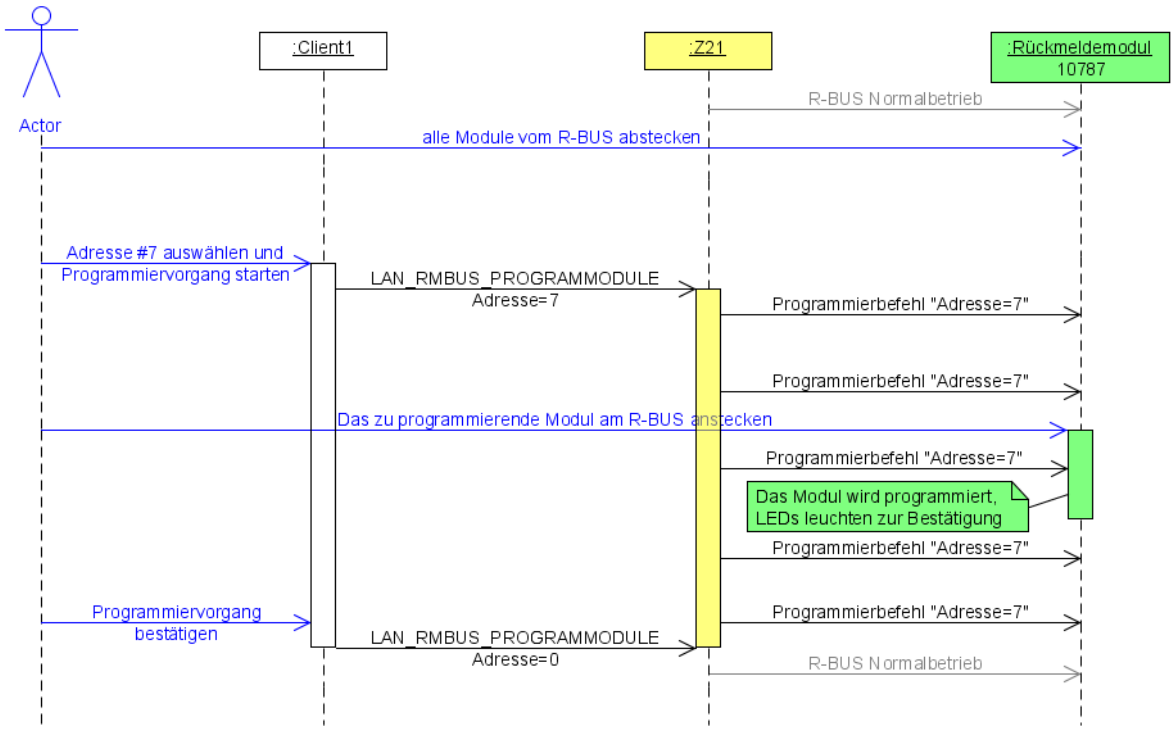

Abbildung 7 Beispiel Sequenz Rückmeldemodul programmieren

8 RailCom

Da die Normierung von RailCom einer stetigen Weiterentwicklung unterliegt, wird von unserer Seite an einer flexibleren Weiterleitung der Daten gearbeitet. Diese Erweiterung des Z21 LAN Protokolls ist zur Zeit in Arbeit und wird voraussichtlich in Form von neuen, zusätzlichen Kommunikations-Befehlen in einer neuen Firmware-Version folgen, sowie zu gegebener Zeit in einer neuen Version von der „Z21 LAN Spezifikation“ an dieser Stelle beschrieben werden.

Mit den folgenden rudimentären Befehlen kann man sich trotzdem einmal mit dem Thema RailCom vertraut machen.

Beachten Sie bitte, dass der Decoder zuerst einmal RailCom fähig sein muss, was alles andere als selbstverständlich ist, und außerdem CV28 und CV29 korrekt konfiguriert sein müssen (siehe Decoderanleitung des Herstellers). Zuletzt muss natürlich noch die Option „RailCom“ in den Einstellungen der Z21 aktiviert sein.

8.1 LAN_RAILCOM_DATACHANGED

Diese Meldung wird von der Z21 an den Client gesendet, welcher die RailCom-Daten explizit angefordert hat, siehe unten *8.2* LAN_RAILCOM_GETDATA.

Z21 an Client:

DataLen		**Header**		**Data**
len	0x00	**0x88**	0x00	Array von **RailComDaten[n]**

DataLen: Die Datenlänge **len** variiert mit der Anzahl der erkannten RailCom Decoder. Siehe Anmerkung unten.

n: Anzahl der erkannten RailCom-Decoder;

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	UINT32	ErrorCounter	Empfangsfehlerzähler in Z21
10	UINT8	Reserved1	experimentell, siehe Anmerkung
11	UINT8	Reserved2	experimentell, siehe Anmerkung
12	UINT8	Reserved3	experimentell, siehe Anmerkung

Anmerkung: Es gilt für Firmware ≤ V1.12:

- es gilt $0 \le n \le 19$; und len = 4 +(n*13) sowie n=(len-4)/13
- *Reserved1* ... RailCom Daten Geschwindigkeit (Message Type Identifier 3 „speed/load“, muss nicht jeder Decoder können)
- *Reserved2*... Options (experimentell)
  Bitmasken für Options:
  `#define rcoSpeed 0x01` // Railcom „Speed“ wurde vom Decoder mind. einmal gesendet
- *Reserved3* ... RailCom Daten Temperatur (Message Type Identifier 8 „Temperature“, muss nicht jeder Decoder können)

8.2 LAN_RAILCOM_GETDATA

RailCom Daten von Z21 anfordern.

Anforderung an Z21:

DataLen		**Header**		**Data**
0x04	0x00	**0x89**	0x00	-

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.

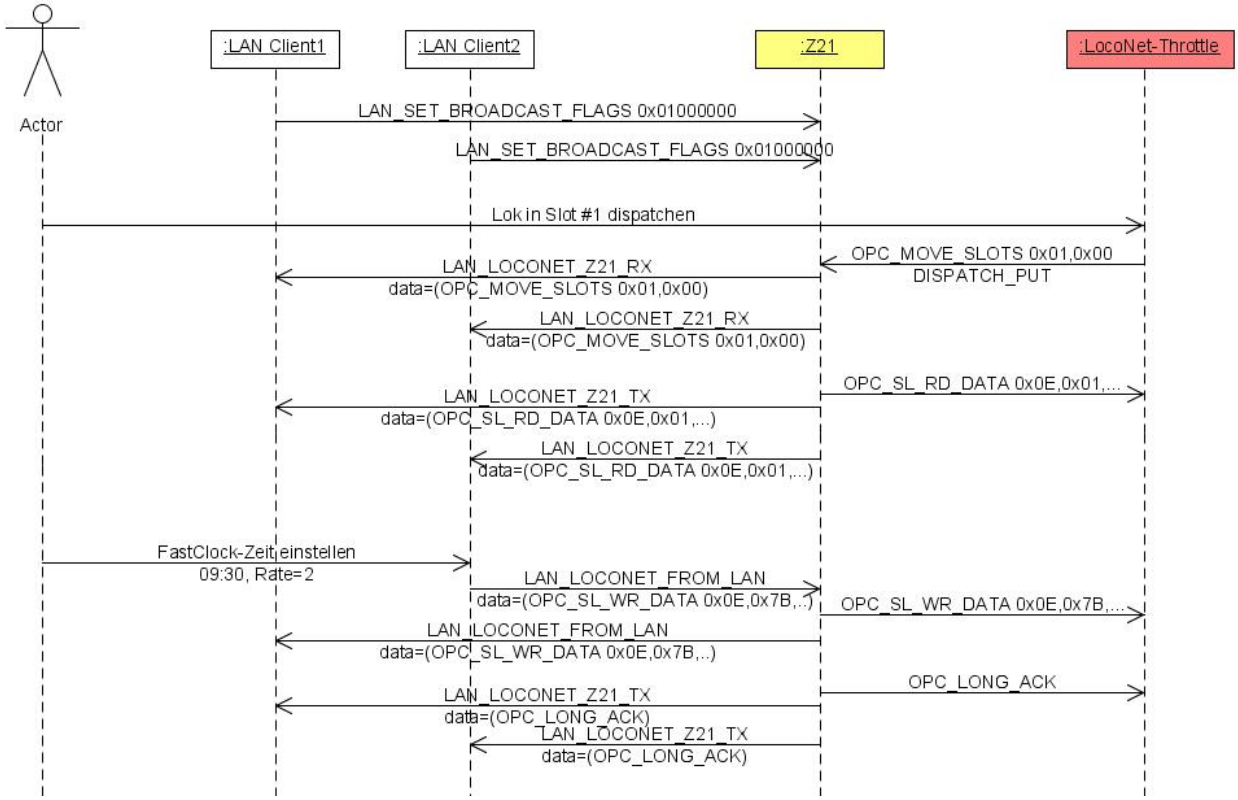

Abbildung 8 Beispiel Sequenz Ethernet/LocoNet Gateway

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

9.4 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. Gegebenfalls 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.

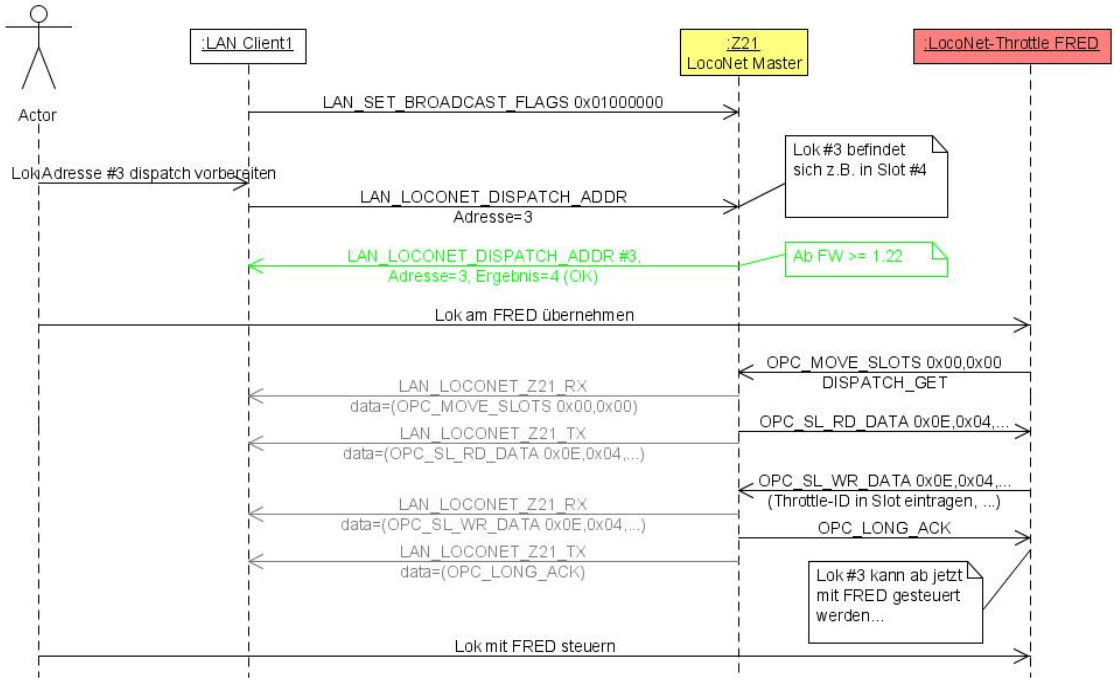

Abbildung 9 Beispiel Sequenz LocoNet Dispatch per LAN-Client

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 „ÜF (Übergabeformat) Uhlenbrock“ (LNCV 15) 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.

0x11 **LISSY Belegtzustand ab Z21 FW 1.23.**
Diese Meldung wird an den Z21 LAN Client geschickt, wenn ein Uhlenbrock LISSY-Empfänger als Belegtmelder konfiguriert ist. 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

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

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

Anhang A – Befehlsübersicht

Client an Z21

Header	Parameter			Name
	X-Header	DB0	Parameter	
0x10	-			LAN_GET_SERIAL_NUMBER
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
0x40	0x23	0x11	CV-Adresse	LAN_X_CV_READ
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	0x53	Weichen-Adresse, Schaltbefehl		LAN_X_SET_TURNOUT
0x40	0x80	-		LAN_X_SET_STOP
0x40	0xE3	0xF0	Lok-Adresse	LAN_X_GET_LOCO_INFO
0x40	0xE4	0x1s	Lok-Adresse,Geschwindigkeit	LAN_X_SET_LOCO_DRIVE
0x40	0xE4	0xF8	Lok-Adresse, Funktion	LAN_X_SET_LOCO_FUNCTION
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	-			LAN_RAILCOM_GETDATA
0xA2	LocoNet-Meldung			*LAN_LOCONET_FROM_LAN*
0xA3	Lok-Adresse			*LAN_LOCONET_DISPATCH_ADDR*
0xA4	Typ, Reportadresse			*LAN_LOCONET_DETECTOR*

Tabelle 1 Meldungen vom Client an Z21

Z21 an Client

Header	Daten			Name
	X-Header	DB0	Daten	
0x10	Serialnumber			Antwort auf LAN_GET_SERIAL_NUMBER
0x1A	HWType, FW Version (BCD)			Antwort auf LAN_GET_HWINFO
0x40	0x43	Weichen-Information		LAN_X_TURNOUT_INFO
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
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
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			Antwort auf LAN_GET_TURNOUTMODE
0x80	Gruppenindex, Rückmelder-Status			LAN_RMBUS_DATACHANGED
0x84	SystemState			LAN_SYSTEMSTATE_DATACHANGED
0x88	RailComDaten[n]			LAN_RAILCOM_DATACHANGED
0xA0	LocoNet-Meldung			*LAN_LOCONET_Z21_RX*
0xA1	LocoNet-Meldung			*LAN_LOCONET_Z21_TX*
0xA2	LocoNet-Meldung			*LAN_LOCONET_FROM_LAN*
0xA3	Lok-Adresse, Ergebnis			*LAN_LOCONET_DISPATCH_ADDR*
0xA4	Typ, Rückmelderadresse, Info			*LAN_LOCONET_DETECTOR*

Tabelle 2 Meldungen von Z21 an Clients

Abbildungsverzeichnis

Tabellenverzeichnis