Norm	RCN-218 **DCC-Protokoll** **DCC-A – Automatische Anmeldung**	RailCommunity
Ausgabe 27.11.2022		RailCommunity – Verband der Hersteller Digitaler Modellbahnprodukte e.V.

Inhalt

1 Allgemeines

1.1 Zweck der Norm

Diese Norm beschreibt DCC-A, ein automatisches Anmeldeverfahren für DCC. Damit wird die Anwenderfreundlichkeit von Modellbahnsteuerungen signifikant erhöht. Bei Anwendung dieser Norm wird der Benutzer bei der Vergabe von Adressen und Zuweisung von Funktionen entlastet. Ziel ist es, z.B. ein Fahrzeug nach dem Aufgleisen sofort mit Namen und allen Eigenschaften im Stellpult verfügbar zu haben.

Anwendung dieser Norm bietet:

- Super schnelles Anmelden - echtes Plug&Play. Auspacken, aufs Gleis, Loslegen.
- Direkte Verfügbarkeit der Eigenschaften des Decoders / des Fahrzeugs
- Aufwandsarme Implementierung sowohl für Zentralen als auch für Decoder
- Kompatibilität zu vorhandenen Decodern und Zentralen

Diese Norm setzt auf die in den Normen [RCN-211] und [RCN-217] beschriebenen Paketstrukturen für DCC bzw. für RailCom auf.

Für die Implementierungsunterstützung in der Zentrale und dem Decoder stehen Header-Dateien und Beispiele zur Prüfsummenberechnung bei der RailCommunity bereit.

Diese Norm beschreibt nur den Datenaustausch zwischen Decoder und Zentrale, welcher für die Anmeldung und die dringend zur Anmeldung benötigten Daten umfasst. Weiterer Datenaustausch erfolgt dann basierend auf der zugeteilten DCC-Adresse.

1.2 Anforderungen

Um diese Norm zu erfüllen, ist es erforderlich, dass alle hier genormten Befehle und Datenstrukturen unterstützt werden. Optionale Bestandteile sind separat gekennzeichnet.

Der korrekte und schnelle Empfang der in dieser Norm definierten Nachrichten ist für den Anmeldevorgang wichtig. Damit dies gewährleistet ist, sind folgende Vorgaben einzuhalten:

- Nachrichtenabsicherung gemäß Kapitel 1.3.
- Nachrichten entsprechend dieser Norm dürfen direkt aufeinander folgen, ein ev. in anderen Normen definierter Mindestabstand zweier Nachrichten an den gleichen Decoder ist nicht vorgesehen.

1.3 Nachrichtenabsicherung

Wesentlich für die Übertragungssicherheit ist eine komplette Auswertung des Bittimings der DCC-Kodierung, der Decoder muss beide Flanken des DCC-Signals bei der Auswertung beachten. Nachrichten, die bei der Auswertung Fehler zeigen sind zu verwerfen.
Aus Gründen der Harmonisierung mit [S-9.2.1.1] wird bei Nachrichten zusätzlich eine Absicherung mit einer 8-Bit CRC eingeführt. Da es damit zwei Prüfbytes gibt – das per EXOR generierte Prüfbyte nach [RCN-211] und die neu eingeführte CRC-Prüfsumme – werden im Folgenden die Begriffe EXOR-Byte (oder kurz EXOR) und CRC-Byte (oder kurz CRC) verwendet.

Es gelten folgende Regeln:

a) DCC-Nachrichten:
DCC-Nachrichten werden immer und generell durch ein abschließendes XOR geschützt. Bei Nachrichten entsprechend dieser Norm, deren erstes Byte 254 ist, wird eine CRC vor dem XOR eingefügt, wenn die Nachricht (ohne jegliche Prüfsumme) 6 oder mehr Bytes einschließlich der Adresse 254 enthält. Nachrichten, deren CRC-Prüfsumme (sofern vorhanden) oder XOR fehlerhaft ist, sind zu verwerfen.

b) RailCom-Nachrichten:
Hier wird der Einsatz der CRC bei der jeweiligen Nachricht beschrieben.

c) Berechnung der CRC:
Auf der Senderseite wird das gemäß Polynom $x^8 + x^5 + x^4 + 1$ über die Nachricht gebildet, beginnend beim ersten Byte der Nachricht, initialisiert auf 0 (bzw. die Nummer des Datenraums, siehe Abschnitt 4.3), nicht invertiert. Empfängerseitig wird die CRC mit dem gleichen Polynom über die gesamte Nachricht inkl. CRC gebildet, das Ergebnis muss 0 sein. Beispiele, Informative Anmerkungen und Algorithmusvorschläge siehe Anhang C: Berechnung CRC.

1.4 Glossar, Definitionen

Innerhalb dieser Norm gelten folgende Festlegungen:

- Die Bits eines Bytes werden von 0 bis 7 gezählt. Bit 0 ist das niederwertigste Bit und steht ganz rechts und Bit 7 ist das höchstwertigste und steht ganz links.
- Zwischen den Bits 4 und 3 wird für eine bessere Lesbarkeit ein Strich eingefügt: 7654-3210.
- Abgesehen von der Beschreibung des Aufbaus der DCC-Pakete im Abschnitt 3 werden
  - die Nullen zwischen den Bytes eines DCC-Paketes,
  - die Synchronbits
  - das einleitende Byte 1111-1110 und
  - die Prüfbytes CCCC-CCCC (CRC) und/oder PPPP-PPPP (XOR) nicht dargestellt.

  Damit sind die DCC-Pakete immer 2 bzw. 3 Bytes länger als der dargestellte Befehl.
- Es wird häufig das ganze Byte dargestellt; Bits ohne Bedeutung im aktuellen Kontext werden mit 'x' gekennzeichnet.
- Folgende Zeichen werden zur Kennzeichnung der Bedeutung eines Bits verwendet:

  0 Bitwert 0
  1 Bitwert 1
  A Adressbit
  B Befehlsbit
  C CID = Kennung der Zentrale (siehe Glossar)
  G Adressierungsgruppe
  H Herstellerkennung entsprechend [S-9.2.2 Appendix A]
  M Adressierungsmodus
  N Nummer des Datenraumes
  P Prüfbits
  S Session ID (Sitzungsnummer)
  U Unique ID Teil des Herstellers (32 Bit, Produktkennung + Seriennummer)

V Konfigurationsvariablenbits (Adressierung)
x Platzhalter für ein Bit, dessen Wert von der Art des Pakets und des Befehls abhängt und an der Stelle nicht näher betrachtet wird.

- Zusätzlich werden folgende Begriffe festgelegt:

Unique ID: Die von Hersteller in den Baustein (Decoder/Zentrale) fest programmierte, eineindeutige Kennung, bestehend aus 12 Bit Herstellerkennung und 32 Bit herstellerspezifischer Nummer (z.B. Produktindex und Seriennummer). Sofern eine Eingabe/Darstellung erforderlich ist, soll diese im folgenden Format erfolgen, wobei hier ausnahmsweise jeder Buchstabe für ein Hex Zeichen (Nibble), also jeweils 4 Bit, steht:
HHHUUUUUUUuu,
HHH bezeichnet die 12 Bit Herstellerkennung entsprechend [S-9.2.2 Appendix A], wobei im ersten Nibble die obersten 4 Bit der Herstellerkennung stehen,
UUUUUUUuu bezeichnet die eindeutige Kennung (als 32 Bit Hex), diese wird (wie sonst auch in DCC typisch) als Big Endian dargestellt. **uu** bezeichnet also das niederwertigste Byte.
Die Darstellung bei Ausgaben soll in der Form V… P …. …. erfolgen, wobei anstelle der Punkte die Hexadezimalwerte für HHH und UUUUUUUU in obiger Reihenfolge eingetragen werden. (V, P steht für Vendor, Product)

ACK: Acknowledge entsprechend [RCN-217] in der Codierung 0x0F

DID: Die Unique ID eines Decoders.

CID: Die Kennung der Zentrale. Hierbei handelt es sich um einen 16 Bit Wert, welcher durch einen Hash der Herstellerkennung und der Unique ID der Zentrale berechnet wird. Sie dient dazu, dass Decoder einen Zentralenwechsel erkennen können und sich dann entsprechend neu anmelden.

Session ID: Eine Variable, welche die aktuelle Betriebsphase kennzeichnet. Die Session ID soll bei jedem Einschalten der Zentrale um 1 inkrementiert werden.

Backoff: Sollte ein Decoder nach einer versuchten Anmeldung keine Bestätigung erhalten, so beantwortet dieser eine (variable) Anzahl von LOGON_ENABLE-Nachrichten nicht mehr.

2 Genereller Ablauf

Decoder werden bei einer DCC-Ansteuerung parallel angesteuert. Deshalb ist ein Adressierungsschema erforderlich. Zur Unterscheidung von mehreren Decodern wird eine eindeutige Kennung verwendet (Unique ID). Ausgehend von dieser Unique ID wird den Decodern eine verkürzte (Sitzungs-)Adresse zugewiesen, um im laufenden Betrieb die knappe Ressource 'DCC-Bandbreite' möglichst optimal zu nutzen. Sofern möglich, wird bei der Sitzungsadresse die bisherige Decoderadresse verwendet. Hierzu wird zu Beginn eine Anmeldeprozedur durchgeführt, um diese Zuordnung durchzuführen und die Decoder- und Fahrzeugeigenschaften für die Steuerung des Fahrzeuges bekannt zu machen.

Die automatische Anmeldung unterteilt sich in folgende Hauptphasen:

- Vereinzelungsphase:
  Hier werden die vorhandenen Decoder ermittelt und fallweise auftretende Zugriffkonflikte gelöst. Am Ende der Vereinzelungsphase sind der Zentrale die DIDs der vorhandenen Decoder bekannt.
- Bekanntmachungsphase:
  Hier tauschen Zentrale und Decoder Informationen über zu verwendende Adresse, Loknamen, vorhandene Funktionen usw. aus.
- Registrierung:
  Der Decoder wird an der Zentrale registriert und kann in Folge betrieblich kontrolliert ('gesteuert') werden.

Zur Beschleunigung des Anmeldeverfahrens kann und soll die Zentrale versuchen, aus der letzten Betriebsphase bereits bekannte Decoder direkt ohne Vereinzelungsphase zu registrieren.

2.1 Vereinzelungsphase

Die Zentrale sendet Aufforderungen an die Decoder, sich anzumelden (Befehl: LOGON_ENABLE). Diese Anmelde-Aufforderungen enthalten eine eindeutige Kennung, bestehend aus Zentralen/Anlagen ID (CID) und einer Session ID. Die Decoder können anhand dieser Kennung die Zentrale nach einem Power Cycle wieder erkennen. Die Decoder antworten auf den Befehl LOGON_ENABLE nach bestimmten Regeln mit ihrer Unique ID (RailCom-Nachricht ID 15 Decoder-Unique Anmeldung).

Wenn bereits viele Decoder in der Zentrale bekannt sind oder lokale RailCom-Detektoren verwendet werden, wird diese Phase kurz sein. Bei Kollisionen von gleichzeitigen Antworten mehrerer Decoder ist die Erkennung nicht sicher, es wird dann eine Vereinzelung mittels dynamischen Backoffs der Decoder durchgeführt. Die Vereinzelung erfolgt (gekennzeichnet durch die Kodierung des LOGON_ENABLE Befehls) getrennt für Zubehördecoder und mobile Decoder oder gemeinsam.

2.2 Bekanntmachungsphase

Die Zentrale bestätigt die Anmeldung und spricht den Decoder über seine DID an (Befehle: SELECT / GET_DATA_*). Der Decoder weiß nun, dass seine Anmeldung erkannt wurde und übermittelt in der zugehörigen RailCom-Antwort eine Zusammenfassung seiner wichtigsten Steuerparameter wie z.B. die gewünschte Decoderadresse bzw. weitere Informationen.

2.3 Registrierung

Die Zentrale weist dem Decoder eine Fahrzeugadresse zu, welche in dieser Sitzung zu verwenden ist. (LOGON_ASSIGN). Der Decoder beantwortet diese Nachricht mit einer Revisionskennung (inkrementelle Nummer / Hash seiner CVs), damit kann die Zentrale überprüfen, ob die (eventuell bereits bekannten) Parameter des Decoders weiterhin gültig sind oder ob weitere Ausleseaktionen bzw. Einmessen erforderlich sind.
Die zugewiesene Adresse gilt nur sitzungsbezogen, für eine dauerhafte Zuweisung der übermittelten Adresse in die CV1 bzw. CV17/18 (und CV29 Bit5) ist der entsprechende DCC-Befehl zu verwenden (vorzugsweise die kurze Form des Konfigurationsvariablen Zugriffsbefehls). Ebenso verliert mit der Registrierung eine eventuell vorhandene Mehrfachtraktionsadresse (aus CV19) ihre Gültigkeit. Die Mehrfachtraktion wird erst wieder gültig, wenn sie erneut mit dem Befehl Mehrfachtraktionsadresse gesetzt oder per POM DCC-Befehl bestätigt wurde. Die zugewiesene Adresse gilt für alle Fahrzeugbefehle.

2.4 Beschleunigtes Einlesen/Schreiben von Decoderparametern

Die begrenzte Bandbreite von DCC und RailCom erfordert eine effiziente Nutzung, um Daten in akzeptabler Geschwindigkeit aus dem Decoder in die Zentrale zu bringen.

Hierzu werden die relevanten Informationen im Decoder in Datenräume entsprechend Abschnitt 5 unterteilt. Jeweils ein kompletter Datenraum kann mit den Befehlen SELECT / GET_DATA_* ausgelesen werden bzw. mit SELECT / SET_DATA_* beschrieben werden. Die Gruppen umfassen u.a. die Protokollfähigkeiten, grundlegende Decodereigenschaften wie Name, Funktionen, Lokbild und direkter CV-Zugriff.

Darüber hinausgehende Daten sollen mit einem allgemeingültigen Transferverfahren nach [S-9.2.1.1] übertragen werden, dort ist sowohl direktes Lesen als auch Lesen im Hintergrund möglich.

3 DCC-Befehle

Generell gilt für alle DCC-Befehle zur automatischen Anmeldung folgendes Format:

{Synchronbits} 0 1111-1110 0 {Befehlsbyte}
{[0 Parameterbyte(s)]} 0 {CRC 0} PPPP-PPPP 1

Generell beginnen alle Befehle zur automatischen Anmeldung mit **1111-1110**. Das darauf folgende Befehlsbyte gibt die Art des Befehls an, weitere Bytes legen die Parameter fest. Ein DCC-Paket kann dabei bis zu 15 Bytes lang werden. Je nach Befehlslänge wird eine CRC eingefügt (siehe Kapitel 1.3).

3.1 Befehlscodierung

Das erste Byte eines Befehls legt die Art des Befehls fest. Nachfolgend eine Übersicht aller Befehle, sortiert nach dem Wert des ersten Bytes. In den folgenden Abschnitten werden die Befehle dann nach ihrer Funktion sortiert beschrieben.

Befehlsbyte (zweites Byte im Paket)	Länge	Ausw.	RailCom-Antwort	Bedeutung
0000-0000	1	Hist.	Data Stream	GET_DATA_START: Datenabfrage erster Transportbefehl[1]
0000-0001	1	Hist.	Data Stream	GET_DATA_CONT: Datenabfrage Transportbefehl[1]
0000-0010	10	Hist.	Data Stream	SET_DATA_START: Daten Schreiben erster Befehl
0000-0011	10	Hist.	Data Stream	SET_DATA_CONT: Daten Schreiben
0000-0100 bis **0000-1111**	-	Hist.	keine	reserviert
0001-0000 bis **1011-1111**	-	-	keine	reserviert
1100-HHHH	-	-	keine	reserviert für Updateprotokoll
1101-HHHH	7 … 12	DID	Data Stream / ACK	SELECT: Anwahl des Decoders, zugleich Anmeldebestätigung
1110-HHHH	8	DID	ID13, Stat (direkte Antwort)	LOGON_ASSIGN: Adresszuordnung, Betriebserlaubnis
1111-0001 bis **1111-1011**	-	-	-	reserviert

[1] Befehl ist optional

1111-11xx	4	Backoff	ID15, Anmeldung (direkte Antwort)	LOGON_ENABLE: Anmeldeaufforderung (xx = Mode)

Hinweise:

- Die Spalte Länge gibt die gesamte Befehlslänge in Bytes (ohne einleitendes Byte 0xFE und Prüfbytes) an.
- Die Befehlsanordnung ist so gewählt, dass innerhalb des Decoders eine schnelle Auswahl der nötigen RailCom-Antwort erfolgen kann (Gruppierung).
- Ausw. bezeichnet die im Decoder nötige Auswertung:
  - Hist.: Auswertung und RailCom-Antwort abhängig von der Vorgeschichte
  - DID: Auswertung und RailCom-Antwort nur bei zutreffender Decoder-Unique ID
  - Backoff: Auswertung und RailCom-Antwort je Befehl und Algorithmus (nur wenn noch nicht angemeldet)

3.2 LOGON_ENABLE

Dieser Befehl ist 4 Byte lang und hat das Format:

1111-11GG CCCC-CCCC CCCC-CCCC SSSS-SSSS

Parameter	Bedeutung
GG	Die Adressierungs**g**ruppe bestimmt, welche Decoder reagieren dürfen:
	00 ALL: Alle Decoder
	01 LOCO: Nur Fahrzeugdecoder
	10 ACC: Nur Zubehördecoder
	11 NOW: Alle Decoder (ohne Beachtung von Backoff)
CCCC-CCCC	CID, erst MSByte, dann LSByte; Zentralenkennung
SSSS-SSSS	Session ID (Sitzungsnummer)

Das ist die Anmeldeaufforderung, diese wird von der Zentrale periodisch ausgesendet. Die Anmeldeaufforderung ist mindestens alle 300ms auszusenden. Nach dem Systemstart empfiehlt sich eine häufigere Aussendung.

Als Antwort senden die Decoder ihre Unique ID (DID) bzw. geben keine Antwort, wenn der Decoder aktuell eine BACKOFF-Wartezeit abwarten muss. Bei 'NOW' antworten alle noch nicht angemeldeten Decoder, anschließend wird das decoderinterne Backoff wieder neu zufällig im Bereich 0...7 gesetzt.

3.3 SELECT

Der Befehl SELECT besteht aus einem Adressteil (7 Bytes), gefolgt von einem Unterbefehlsbyte und optional weiteren Parametern. Er hat folgendes Format:

1101-HHHH HHHH-HHHH UUUU-UUUU UUUU-UUUU UUUU-UUUU UUUU-UUUU BBBB-BBBB (NNNN-NNNN VVVV-VVVV VVVV-VVVV vvvv-vvvv SSSS-SSSS)

Parameter	Bedeutung
HHHH...HHHH	12 Bit Herstellerkennung entsprechend [S-9.2.2 Appendix A], die unteren 8 Bit der Herstellerkennung stehen im zweiten Byte des Befehls.
UUUU...UUUU	32 Bit, eindeutige Kennung des Decoders (Teil 2 der Unique ID)
BBBB-BBBB	Unterbefehlsbyte (siehe 3.3.1 ff)
NNNN-NNNN	(optional, abhängig von **BBBB-BBBB**) Nummer des angefragten Datenraumes
VVVV-VVVV **VVVV-VVVV** **vvvv-vvvv**	(optional, abhängig von **BBBB-BBBB** und **NNNN-NNNN**) CV-Adresse bei Zugriff auf den CV-Datenraum. **vvvv-vvvv** bezeichnet dabei das niederwertige Byte; das erste **VVVV-VVVV** entspricht CV31, das zweite **VVVV-VVVV** entspricht CV32.
SSSS-SSSS	(optional, abhängig von **BBBB-BBBB**) Anzahl der angefragten CVs

Mit diesem Befehl wählt die Zentrale einen Decoder aus. Durch diese Nachricht erkennen Decoder, dass ihre Anmeldung erkannt wurde und sie dürfen keine weiteren Anmeldungen mit ID15 senden.

Der Befehl SELECT wird mittels ACK (8 Bytes) oder mit dem angefragten Datenstrom bestätigt. *(Informativ: damit ist es möglich, in einer Minimalimplementierung ohne die Auswertung von GET_DATA auszukommen.)*

Im Befehl SELECT gibt der Unterbefehl die vom Decoder auszuführende Operation an. Die Kodierung dieses Unterbefehls erfolgt analog zu [S-9.2.1.1], wobei hier nur ein Teil der dort möglichen Operationen verwendet werden.

Unterbefehlsbyte	Bedeutung
1111-1111	Read ShortInfo
1111-1110	Read Block
1111-1101	reserviert. (Informativ: in [S-9.2.1.1]: Read Background)
1111-1100	reserviert für Write Block
1111-1011	Setze Decoder-internen Status
1111-1010 … **0000-0000**	reserviert

3.3.1 Unterbefehl Read ShortInfo

Der Unterbefehl für das Lesen der Kurzinformation zur Anmeldung ist wie folgt definiert:

1111-1111

Nach dem Lesebefehl antwortet der Decoder in der folgenden Cutout mit den angefragten Daten ShortInfo (siehe Abschnitt 5 Datenräume.). Es folgt keine anschließende GET_DATA_* Phase.

3.3.2 Unterbefehl ReadBlock

Der Unterbefehl für blockweises Lesen ist wie folgt definiert:

1111-1110 NNNN-NNNN

Parameter	Bedeutung
NNNN-NNNN	Die Nummer des Datenraumes, der abgefragt wird.

Zur Definition der Datenräume siehe Abschnitt 5 Datenräume.

Nach einem SELECT mit Lesebefehl ReadBlock antwortet der Decoder in der folgenden Cutout mit den angefragten Daten. Sollte dies aus Timinggründen bei der Auswertung nicht möglich sein, so muss die Antwort spätestens beim folgenden Befehl GET_DATA_START erfolgen. Alle folgenden GET_DATA_CONT-Befehle sind lückenlos zu nutzen.

Datenräume haben generell eine variable Länge. Bei CV-Zugriffen wird der Befehl um die Parameter **VVVV-VVVV VVVV-VVVV vvvv-vvvv SSSS-SSSS** ergänzt. Diese Parameter bezeichnen 24-Bit Adresse, bei welcher das Lesen begonnen wird und die Anzahl an CVs, die gelesen werden sollen. Werden die Parameter **VVVV-VVVV VVVV-VVVV vvvv-vvvv SSSS-SSSS** gesendet, obwohl dieses nicht erforderlich wäre, werden sie ignoriert.

In der Antwort zu diesem Befehl und bei folgenden GET_DATA_CONT-Befehlen sendet der Decoder die Daten dieses Datenraumes. Er inkrementiert dabei selbständig den Zugriff innerhalb des Datenraumes. Bei Wechsel oder erneutem Aufruf des Datenraumes beginnt der Decoder im neuen Datenraum immer mit Index 0 bzw. bei CV-Zugriff mit der angegebenen Adresse.

3.3.3 Unterbefehl ReadBackground

Der Unterbefehl für Lesen im Hintergrund ist wie folgt definiert:

1111-1101 NNNN-NNNN

Nur informativ. Wird bei SELECT nicht verwendet.

3.3.4 Unterbefehl WriteBlock

Der Unterbefehl für blockweises Schreiben ist wie folgt definiert:

1111-1100 NNNN-NNNN

Hinweis: Der Unterbefehl WriteBlock wird in dieser Norm zurückgestellt und es wird auf die Programmiermöglichkeiten mittels POM und XPOM verwiesen.

3.3.5 Unterbefehl Setze Decoder-internen Status

Der Unterbefehl Setze Decoder-internen Status ist wie folgt definiert:

1111-1011 NNNN-NNNN

NNNN-NNNN = **1111-1111** bedeutet Löschen der Changeflags.

Weitere Werte für NNNN-NNNN sind noch nicht festgelegt und reserviert.

3.4 GET_DATA_*

Der Befehl GET_DATA_* ist 1 Byte lang und hat das Format:

0000-000x

Es gibt zwei Untervarianten dieses Befehls:

GET_DATA_START: **0000-0000**

Das ist der erste Befehl nach einer Decoderauswahl (z.B. durch SELECT). Weitere GET_DATA_CONT-Befehle können folgen. Dieser Befehl folgt unmittelbar (back-to-back) nach einer Decoderauswahl.

GET_DATA_CONT: **0000-0001**

Das ist ein Datentransportbefehl. Weitere GET_DATA_CONT-Befehle können folgen.

Diese Befehle dienen zum Abholen von Daten, wenn der Decoder vorher mittels SELECT (oder einem anderen entsprechenden Befehl) selektiert wurde. Der Decoder liefert die Daten des abgerufenen Datenraumes wie im Kapitel 4.4 beschrieben. Dieser Befehl ermöglicht damit schnelles Auslesen von Blöcken aus Datenräumen. (Hinweis: Die Befehlsdauer beträgt 6,4ms, so dass ein Block von 30 Bytes in etwa 32ms übertragen werden kann). Sofern der Decoder keine Datenräume anbietet, kann die Implementierung von GET_DATA entfallen.

Dabei gelten folgende Regeln:

1. Der Decoder darf die Cutout nach GET_DATA_START nur nutzen, wenn er im Befehl unmittelbar vor GET_DATA_START adressiert wurde.
2. Die Cutout nach GET_DATA_CONT darf nur verwendet werden, wenn der Decoder seit dem letzten Start seiner Übertragung (mittels GET_DATA_START) kontinuierlich DCC empfangen und einschließlich der Befehlsinterpretation fehlerfrei auswerten konnte.

3. Wenn das Ende der zu übertragenden Daten erreicht ist, füllt der Decoder ev. noch fehlende Bits zur Codierung des letzten RailCom-Symbols mit 0-Bits auf. Anschließend wird die restliche Aussendung in dieser Cutout mit ACK aufgefüllt.
4. Wenn der Decoder einen Datenraum nicht kennt, so antwortet er mit einem HEADER = 0, gefolgt von ACKs.

3.5 SET_DATA_*

Hinweis: SET_DATA wurde in dieser Version der Norm zurückgestellt und es wird auf die Programmiermöglichkeiten mittels POM/XPOM verwiesen (siehe Abschnitt 6.2 Lesen von Decoderparametern).

3.6 LOGON_ASSIGN

Der Befehl LOGON_ASSIGN ist 8 Byte lang und hat das Format:

1110-HHHH HHHH-HHHH UUUU-UUUU UUUU-UUUU UUUU-UUUU uuuu-uuuu 11AA-AAAA AAAA-AAAA

Parameter	Bedeutung
HHHH...HHHH	12 Bit Herstellerkennung entsprechend [S-9.2.2 Appendix A], die unteren 8 Bit der Herstellerkennung stehen im zweitem Byte der Nachricht.
UUUU...UUUU	32 Bit, eindeutige Kennung des Decoders (Teil 2 der Unique ID)
11	Reserviert, mit 11 vorzubelegen. Nachricht ist bei anderen Werten zu ignorieren.
AAAA...AAAA	Zugewiesene Decoderadresse, Adresszuordnung siehe Anhang D

Mit diesem Befehl registriert die Zentrale den Decoder, dieser wird anschließend unter der übermittelten Adresse **AA-AAAA AAAA-AAAA** angesprochen. Der Decoder darf die Registrierung nur akzeptieren, wenn er die CID (Zentralenkennung) und die Session ID bereits erfasst hat.

Der Decoder antwortet in der direkt auf den LOGON_ASSIGN folgenden Cutout mit einer Nachricht, welche eine Zusammenfassung / Änderungsindex (Hash) seiner CVs enthält. Damit kann die Zentrale überprüfen, ob die (eventuell bereits bekannten) Parameter des Decoders weiterhin gültig sind oder ob weitere Ausleseaktionen bzw. erneutes Einmessen erforderlich sind.

Die Adresszuweisung gilt nur sitzungsbezogen. Für eine dauerhafte Zuweisung der übermittelten Adresse in die CV1 bzw. CV17/18 (und CV29 Bit5) oder CV19 ist ein POM DCC-Befehl oder der Befehl Mehrfachtraktionsadresse setzen zu verwenden.

Die Zuweisung einer kurzen Adresse mit dem Wert 0 bedeutet: Anmeldung wurde erkannt, die Zentrale will aber den Decoder aktuell nicht ansprechen. ('Parken').

Mit der automatischen Anmeldung wird eine eventuell vorhandene Mehrfachtraktionsadresse nicht mehr beachtet. Erst wenn die Zentrale die Mehrfachtraktionsadresse erneut mit einem Mehrfachtraktionsbefehl oder POM DCC-Befehl zugewiesen hat, wird die Mehrfachtraktion wieder aktiviert.

4 RailCom-Nachrichten

Generell werden bei den Antworten zu DCC-A-Nachrichten die beiden RailCom-Kanäle 1 und 2 gebündelt. Die Aufteilung der Zeiten bleibt jedoch erhalten (wie in der RCN217 angegeben). Durch die Bündelung entstehen Nachrichten mit immer 8 Byte, nach der 6-8-Kodierung verbleiben damit 48 Bit. Diese 48 Bit werden wie folgt aufgeteilt (Big Endian):

47 – 44	**43 – 40**	**39 – 36**	**35 – 32**	**31 – 0**
ID	EXT1	EXT2	EXT3	DATA

(Hinweis: EXT1 und EXT2 sind in Kanal 1, EXT3 führt den Kanal 2 an.)

Eine Sonderrolle nimmt die Nutzung der RailCom-Cutout bei der Datenraumübertragung mittels GET_DATA_* ein. Hier werden die RailCom-Daten einfach als Bytefolge betrachtet, ID und EXT1 bilden dabei das erste Byte, EXT2 und EXT3 das zweite Byte, DATA[31-24] das dritte Byte, usw.

Die Datenabsicherung erfolgt immer mit CRC, einzige Ausnahme: ID15, Decoder-Unique.

4.1 ID15 - Decoder-Unique (Anmeldung)

Diese Nachricht wird als Antwort auf den DCC-Befehl LOGON_ENABLE gesendet.

ID15 - Decoder-Unique		
ID	**1111**	ID15, Kennung für Decoder Unique
EXT1	**HHHH**	4 Bit Herstellerkennung (entsprechend [S-9.2.2 Appendix A]), hier die obersten 4 Bits der 12 Bit Herstellerkennung
EXT2, EXT3	**HHHH**...**HHHH**	8 Bit Herstellerkennung (entsprechend [S-9.2.2 Appendix A]), hier die untersten 8 Bits der 12 Bit Herstellerkennung
DATA	**UUUU...UUUU**	32 Bit eindeutige Kennung des Decoders (Teil 2 der Unique ID)

Die 12-bit Herstellerkennung ergibt sich zu EXT1*256+EXT2*16+EXT3. Bei dieser Nachricht erfolgt keine CRC-Absicherung.

4.2 ID13 - Decoder-State

Diese Nachricht wird als Antwort auf den DCC-Befehl LOGON_ASSIGN gesendet und enthält Informationen, ob und welche Konfiguration des Decoders verändert wurde. Zudem sind hier weitere Auskunftsbits zu den (Protokoll-)Fähigkeiten des Decoders enthalten. Diese Nachricht bestätigt zudem die erfolgreiche Zuweisung der Adresse.

ID13 - Decoder-State

ID	**1101**	ID13, Decoder-State
EXT1, EXT2	**FFFF-FFFF**	Changeflags, welche Hinweise auf geänderte Decoderparameter geben. EXT1 enthält die MSBs.
EXT3, DATA[31..24]	**CCCC-CCCC-CCCC**	Changecount, 12 Bit. EXT3 enthalt die MSBs.
DATA[23..8]	**DDDD**...**DDDD**	Auskunft über Protokollfähigkeiten des Decoders, Spiegel der Bytes 2 und 3 des Datenraumes 0. Siehe hierzu auch Kap. 5.2. Extended Capabilities.
CRC	**CCCC-CCCC**	

Die Changeflags liefern dem steuernden System Informationen, ob sich relevante CVs verändert haben. Dadurch ist bei einer erneuten Anmeldung schnell zu erkennen, ob vorhandene Werte für Fahrverhalten und Bedienung unverändert geblieben sind.

Changeflags	
Bit	**Bedeutung**
F0	(LSB) Das letzte Rücksetzen der Changeflags erfolgte an einer Zentrale mit anderer ID.
F1	Die Firmware des Decoders wurde verändert.
F2	Das Fahrverhalten des Decoders hat sich geändert. Z.B. Motorparameter, Beschleunigung, Bremsen, Geschwindigkeit. bei Zubehördecodern: Schaltzeiten haben sich geändert
F3	Die Funktionszuordnung des Decoders hat sich geändert.
F4	GUI-Daten (wie Lokname, Lokbild) wurden verändert.
F5	Mehrfachtraktion (CV19) ist aktiviert.
F6	reserviert.
F7	(MSB) reserviert.

Der Decoder State ist vom Decoder zu verwalten, d.h. bei jeder Veränderung einer CV oder Änderung der Firmware müssen die entsprechenden Changeflags gesetzt werden und der Zähler ist zu inkrementieren. Eine Zentrale sollte nach einer Programmierung den Zähler über ein LOGON_ASSIGN auslesen und für die Unique ID speichern um zu erkennen, ob eine Änderung von einer anderen Zentrale durchgeführt wurde. Ein Wert von 0xFFF entspricht einem neu programmierten Decoder und ist daher immer als geänderter Wert zu betrachten.

Der Unterbefehl Setze Decoder-internen Status mit **NNNN-NNNN** = **1111-1111** löscht alle Changeflags.

4.3 Datenraumübertragung mittels GET_DATA_*

Bei der Datenübertragung mittels GET_DATA_* wird eine Aussendung vom Decoder über eine oder mehrere Cutouts verteilt, dabei gilt immer folgender (Gesamt-)Aufbau:

HEADER [DATEN] CRC

Der HEADER ist ein Byte und enthält Informationen über die folgenden Daten.

HEADER		
Bit 7	**H**	0: Standardformat mit variabler Gesamtlänge, wobei eine oder mehrere Cutouts benutzt werden. Die folgenden Bits in der Headerdefinition sind gültig. 1: Daten gemäß Sonderformat A mit einer Gesamtlänge von 6 Byte, wobei nur eine Cutout benutzt wird.
Bit 6	**R**	Response (*) 0: Antwortnachricht auf eine Anfrage 1: Nachricht, welche ohne Anforderung übermittelt wird
Bit 5	**C**	0: Startblock 1: Fortsetzungsblock
Bit 4…0	**SIZE**	Anzahl der Nutzbytes in diesem Block. Die Gesamtzahl der Bytes ergibt sich zu SIZE+2.

(*): Bei Abfragen nach dieser Norm handelt es sich immer um eine Antwortnachricht.

Bei Antworten auf Datenraum-Lesebefehle wird die Nummer des angefragten Datenraumes als erstes Byte (Startwert) für die CRC-Berechnung verwendet. Die CRC wird ausgehend vom Startwert über HEADER und alle DATEN gebildet und an die Nachricht angehängt.

Wenn der Datenraum größer als 31 Bytes ist muss ab dem zweiten Block im HEADER Bit 5 gesetzt werden. Der Startwert der CRC bleibt dabei die Nummer des Datenraums. Sollte der Datenraum einschließlich CRC mit Byte 31 enden, so ist noch ein leerer Fortsetzungsblock mit HEADER=0x20 zu übertragen.

Sonderformat A:

Um beim Lesen des Datenraumes Decoder Shortinfo besonders schnell und effizient agieren zu können, wird dieser Datenraum immer mit einer festen Gesamtlänge von 6 Bytes kodiert. Das MSB des ersten Bytes ist in diesem Fall = 1. Der Startwert der CRC ist hier 0.

5 Datenräume

Historisch sind die Informationen im Decoder in CVs kodiert, diese sind über einen gewissen Bereich verstreut und auch nicht überall einheitlich implementiert. Für eine automatische Anmeldung ist eine schnelle und effiziente Übertragung dieser Informationen erforderlich, diese werden daher zu Datenräumen zusammengefasst. So kann mit wenigen Transfers die relevante Information transportiert werden.

Es gibt 256 mögliche Datenräume. Davon sind die Räume 0...7 in dieser Spezifikation definiert, alle weiteren Datenräume sind reserviert. Es werden aber Bereiche für bestimmte Sonderfunktionen wie z.B. herstellerspezifische Datenräume spezifiziert.

Ob ein Datenraum verfügbar ist, wird vom Decoder in einem Bitfeld angezeigt, welches selbst wie als Datenraum gelesen werden kann. Die Verfügbarkeit der wichtigsten Datenräume (Extended Capabilites, ShortGUI sowie allgemeiner CV-Zugriff) werden schon in der ShortInfo angekündigt.

Datenraum	
Nummer (dez)	**Inhalt**
--	Shortinfo (feste Größe = 6 Byte), enthält Informationen zu den Protokollfähigkeiten des Decoders und die bisherige DCC-Adresse des Decoders.
0	Extended Capabilities (Variable Größe, bis zu 31 Byte), enthält Informationen zu weiteren Protokollfähigkeiten des Decoders. (reserviert für zukünftige Erweiterungen)
1	SpaceInfo (Variable Größe, bis zu 31 Byte), enthält Informationen zu verfügbaren Datenräumen.
2	ShortGUI (Variable Größe, bis zu 28 Byte), enthält Informationen zum Namen, Bildindex und Funktionsumfang.
3	CV-Block, max. Größe des Datenraumes ist 2^{24}, aus diesem Datenraum wird ein Block der max. Größe 31 gelesen. (Variable Größe)
4	
5	
6	
7	

Datenräume mit variabler Länge werden in adressierten Paketen mit einer Größe von max. 31 Byte übertragen. Bei Datenblöcken mit variabler Größe setzt sich der Datenblock generell wie folgt zusammen:

HEADER [DATEN] CRC

5.1 Datenraum Shortinfo

Dieser Datenraum ist 6 Bytes groß, benutzt das Sonderformat A und enthält die wesentlichen Informationen zum Anmelden des Decoders:

Byte	Bit	Enthaltene Daten
0	7	1: kennzeichnet das Sonderformat
0	6	Reserviert; mit 0 vorzubelegen
0	5 – 0	Fahrzeugadresse (Wunschadresse), Bit 13...8. (höherwertige Bits) (Siehe Anhang D: Adressen)
1	7 – 0	Fahrzeugadresse (Wunschadresse), Bit 7...0. (niederwertiges Byte) (Siehe Anhang D: Adressen) Hinweis: durch eine nicht abbildbare Wunschadresse mit den höherwertigen Bits = 0x3F zeigt der Decoder an, das er sich aktuell im FW-Updatemode befindet und nur FW-Update möglich ist.
2	7 – 0	Bei Fahrzeugdecodern: Höchste verwendete Funktionsnummer einschließlich Funktionszuordnung. Bei Standardzubehördecodern: Höchste Weichenpaarnummer (z.B. ein Decoder mit 4 Schaltpaaren meldet 3). Bei erweiterten Zubehördecodern: Höchster vorkommender Begriff. (z.B. ein Decoder mit den Begriffen Halt, Fahrt und langsame Fahrt meldet 2).
3	7	Decoder unterstützt FW-Update über DCC-A
3	6	Decoder unterstützt XPOM
3	5	Decoder unterstützt SELECT + GET_DATA_* (=ReadBlock)
3	4	Decoder unterstützt ReadBlock über XDCC
3	3	Decoder unterstützt ReadBackground über XDCC
3	2 – 0	Reserviert; mit 0 vorzubelegen
4	**7**	Decoder unterstützt Datenraum 7: Herstellerspezifische Angaben
4	6	Decoder unterstützt Datenraum 6: Produktname
4	5	Decoder unterstützt Datenraum 5: Langer Name
4	4	Decoder unterstützt Datenraum 4: Zuordnung Funktionsicons
4	3	Decoder unterstützt Datenraum 3: CV (Lesen/Schreiben über DCC-A und/oder XDCC, abhängig von den Bits 3-3, 3-4 und 3-5.)
4	2	Decoder unterstützt Datenraum 2: ShortGUI
4	1	Decoder unterstützt Datenraum 1: SpaceInfo
4	0	Decoder unterstützt Datenraum 0: Extended Capabilities
5	7 – 0	CRC über die Bytes 0, 1, 2, 3 und 4

5.2 Datenraum 0 Extended Capabilities

Dieser Datenraum hat eine variable Größe und enthält Informationen über die (weiteren) Protokollfähigkeiten des Decoders. Aus diesem Datenraum sind mit dieser Norm die ersten vier Bytes definiert, weitere Bytes sind für zukünftige Erweiterungen reserviert. Undefinierte / reservierte Bits sind mit 0 vorzubelegen.

Um eine aufwandsarme Implementierung zu ermöglichen, sind diese ersten vier Bytes bereits (als Kopie) bei der Anmeldung in ShortInfo bzw. Decoder-State (ID13) enthalten. Hierbei werden das Byte 0 nach Datenbyte 3 von ShortInfo, das Byte 1 nach Datenbyte 4 von ShortInfo, das Byte 2 nach DATA[23..16] und das Byte 3 nach DATA[15..8] von DecoderState (Kapitel 4.3) gespiegelt.

5.3 Datenraum 1 SpaceInfo

Dieser Datenraum hat eine variable Größe und enthält als Bitfeld Information über verfügbare Datenräume. Eine 1 gibt an, ob der jeweilige Datenraum verfügbar ist. Bytes nach dem letzten verfügbaren Datenraum müssen nicht mehr übertragen werden.

Byte	Bedeutung
0	HEADER
1 – …	Bitfeld Verfügbarkeit Datenraum. Bit 0 im Byte 1 zeigt die Verfügbarkeit des Datenraumes 0 an, Bit 1 im Byte 1 zeigt die Verfügbarkeit des Datenraumes 1 an, …, Bit 0 im Byte 2 zeigt die Verfügbarkeit des Datenraumes 8 an, usw.
…	CRC

Hinweis: Das Byte 1 ist in der Anmeldeauskunft ShortInfo gespiegelt.

5.4 Datenraum 2 ShortGUI

Dieser Datenraum hat eine variable Größe und enthält in gepackter Form Informationen für einfache Bediengeräte. Der Decoder sollte alle zur Verfügung stehenden Funktionen bis max. F68 übertragen. Die übertragenen Daten umfassen:

Byte	Bedeutung
0	HEADER
1 – 8	Benutzerdefinierter Name des Decoders, utf8 kodiert. Wenn weniger als 8 Byte verwendet werden, ist der Rest mit 0 aufzufüllen.
9	16 Bit Nummer des Fahrzeugbildes (MSB) entsprechend [TN-218]
10	16 Bit Nummer des Fahrzeugbildes (LSB) entsprechend [TN-218]
11	Bits 0…3: vereinfachtes Prinzipsymbol
11	Bits 4…5: reserviert
11	Bits 6…7: Funktionsauskunft F0

12 – ..	Funktionsauskunft F1 ... Fx: je zwei Bit pro Funktion, diese geben Auskunft, ob die Funktion benutzbar ist und ob diese schaltend oder tastend verwendet wird.
…	CRC

Soll ein bestimmtes Fahrzeug ein spezielles Bild erhalten, welches nicht in der allgemeinen Liste der Bildnummern in [TN-218] enthalten ist, so ist dieses Bild in der Zentrale mit einem Dateinamen abzulegen, der die Unique ID enthält. Der Dateinamen beginnt mit einen ‚x', gefolgt von der Unique ID in hexadezimaler Schreibweise, optional einem Namen, der von der Unique ID durch einen Unterstrich ‚_' getrennt wird, und einer dem Bildformat entsprechenden Endung.

Der Index für das Prinzipsymbol ist folgendermaßen festgelegt:

Index	**zu verwendendes Prinzipsymbol (mobile Decoder)**	**zu verwendendes Prinzipsymbol (stationäre Decoder)**
0000	Dampflok	Weichen/Schaltdecoder
0001	Diesellok	Lichtsignal
0010	E-Lok	Formsignal
0011	Dieseltriebwagen	Schranken
0100	E-Triebwagen (S-Bahn, ICE)	Drehscheibe
0101	Steuerwagen, Zugschluss	(Anlagen-)Beleuchtung
0110	Personenwagen	Ampel
0111	Güter- und Güterzugbegleitwagen	
1000	Dienstfahrzeuge (Kran, Schneeschleuder, ..)	
1001	Allg. Funktionsdecoder	
1010	Car: Pkw	
1011	Car: Lkw	
1100	Car: Bus	
1101	Car: Sonstige	
1110	Sonstiger Decoder (Mobil)	
1111	Sonstige	Sonstige

Die Funktionsauskunft ist folgendermaßen codiert:

Bit 1, Bit 0	Bedeutung
00	Funktion ist nicht vorhanden
01	Funktion ist vorhanden und schaltend benutzbar (z.B. Licht)
10	Funktion ist vorhanden und tastend benutzbar (im Stile eines retriggerbaren Monoflops, also aktiv, solange Funktionsbefehle empfangen werden) (z.B. Signalhorn)
11	Funktion ist vorhanden und tastend benutzbar (nur Einzelausführung) (z.B. Ansagetext).

5.5 Datenraum 3 CV-Read

Dieser Datenraum hat eine variable Größe und enthält die angefragten CV-Werte.

Aufbau: HEADER [Daten] CRC.

5.6 Datenraum 4 Icon Zuordnung

Dieser Datenraum hat eine variable Größe und enthält Informationen über eine statische Zuordnung von Funktions-Nummern zu Funktions-Icons. Dynamische Zuordnungen von Funktionen können hier nicht dargestellt werden und benötigen eine andere bisher nicht festgelegte Datenstruktur. Daher ist im Fall von dynamischen Funktionen der Datenraum 4 nur eine Rückfallebene. Die Icon-Nummern 224 bis 255 sind für benutzerdefinierte Textnachrichten reserviert.

maximale Anzahl Bytes	Bedeutung
1	HEADER
1	Funktionsnummer
2	Nummer des zuzuordnenden Icons Die Tabelle der Icon-Nummern steht in [TN-218]
1	Funktionsnummer
2	Nummer des zuzuordnenden Icons
	…
1	CRC

Die Zahl der Bytes pro Icon ergibt sich aus den Daten. Die genaue Kodierungsregel ist in TN-218 festgelegt.

5.7 Datenraum 5 Langer Name

Dieser Datenraum hat eine variable Größe und enthält den vom Benutzer vergebenen langen Namen und die vom Benutzer vergebene Beschreibung. Die Informationen werden als Strings nach utf8 codiert gesendet, die Trennung erfolgt jeweils durch ein 0-Byte.

maximale Anzahl Bytes	Bedeutung
1	HEADER
63	langer Name
63	Benutzer Beschreibung
1	CRC

5.8 Datenraum 6 Produktinformationen

Dieser Datenraum hat eine variable Größe und enthält den Namen des Herstellers, den Produktnamen, die Hardware Version und die Software Version. Die Informationen werden als Strings nach utf8 codiert gesendet, die Trennung erfolgt jeweils durch ein 0-Byte.

Dieser Datenraum ist nur lesbar.

maximale Anzahl Bytes	Bedeutung
1	HEADER
41	Herstellername des Decoders
41	Produktname des Decoders
21	Hardware Version des Decoders
21	Software Version des Decoders
1	CRC

5.9 Datenraum 7 Fahrzeugspezifischer Datenraum

Dieser Datenraum hat eine variable Größe und enthält Informationen zum Fahrzeug, in dem sich der Decoder befindet. Hier ist keine Beschreibung vorgesehen, sondern nur der ggf. gekürzte Name des Herstellers sowie die Produktnummer zur Identifikation des Fahrzeugs durch eine Zentrale, die für diesen Hersteller eine Datenbank besitzt.

maximale Anzahl Bytes	Bedeutung
1	HEADER
8	Herstellername des Fahrzeugs
16	Alphanumerische Produktnummer des Fahrzeugs
100	URL für ein Fahrzeugbild
1	CRC

5.10 Weitere Datenräume

Folgende weitere Datenräume sind vorgesehen, aber in dieser Version der Norm noch nicht genau festgelegt.

Displaytext	Benutzermeldung des Decoders
ConsistPartner	Mögliche Partnerdecoder für Mehrfachtraktionsbildung

6 Implementierung in der Zentralen

Diese Kapitel beschreibt das Verhalten von Zentralen im Laufe der Anmeldung und wie diese mit Sonderfällen umgeht.

Bei jedem Neustart einer Zentrale ist die Session ID zu inkrementieren. Nach 255 folgt 0.

6.1 Anmeldung

- Beim Systemstart sendet die Zentrale ein LOGON_ENABLE(NOW), um alle Decoder anzumelden. Die Reaktion wird je nach Art des Rückmeldesystems (lokale Detektoren / globale Detektoren) eine Reihe von ID15 Nachrichten bzw. erkannten Kollisionen sein.
- Jeweils die mit einer ID15-Nachricht bekannt gemachten Decoder werden ausgelesen (SELECT & READ ShortInfo), sie sind damit bekannt und reagieren nicht mehr auf LOGON_ENABLE.
- Wenn Kollisionen erkannt wurden, wird eine Vereinzelung gestartet: die Zentrale sendet eine Folge von LOGON_ENABLE(ALL), je nach aktuellem Backoff-Wert werden sich die Decoder vereinzeln und erfolgreich eine ID15 Nachricht absetzen können.
- Wenn keine neuen Decoder mehr eingehen und auch keine Kollisionen mehr erkannt werden, kann die Zentrale wieder auf LOGON_ENABLE(NOW) wechseln, um eine schnelle Anmeldung neu aufgegleister Fahrzeuge zu ermöglichen.
- In der Zentrale von einer früheren Betriebsphase bekannte Decoder können probehalber nach dem ersten LOGON_ENABLE ausgelesen werden, dies verkürzt i.d.R. die Anmeldephase.

6.2 Lesen von Decoderparametern

Das Lesen von Decoderparametern ist abhängig davon, welche Lesemethode der Decoder unterstützt.

- Das empfohlene Vorgehen:
  Nach erfolgter Anmeldung wird baldmöglichst die Adresse zugewiesen und weitere Information werden über XDCC (siehe [S-9.2.1.1]) abgeholt. Hierbei können die Daten des momentan in der Benutzerbedienung befindlichen Decoders mittel BLOCK_READ erfasst werden, weitere Decoder lassen sich 'unsichtbar' mittels BACKGROUND_READ lesen.
- Fallback 1:
  Bis zur allgemeinen Etablierung von XDCC oder bei Zuweisung identischer DCC-Adressen (Lok und Steuerwagen) ist der Datenraum 'ShortGUI' und CV-Lesen über eine Adressierung mittels DCC-A möglich.
- Fallback 2:
  Sollte ein Decoder Lesen weder über XDCC noch über DCC-A ermöglichen, bleibt als (sehr langsame) Ultima Ratio das normale CV-Lesen über POM oder XPOM.
  Dafür sind die über CV 31 = 2 erreichbaren CV-Seiten reserviert. Über CV 32 = 0 bis CV 32 = 7 werden die Datenräume 0 bis 7 erreicht (Siehe [RCN-225].

7 Verhalten von Decodern

Dieses Kapitel beschreibt das Verhalten von Decodern im Laufe der Anmeldung und wie diese mit Sonderfällen umgehen. Wenn die automatische Anmeldung in CV# 28 Bit 7 nicht freigegeben ist, ignoriert der Decoder alle Befehle an die Adresse 254 und verhält sich wie auf einer Anlage ohne automatische Anmeldung.

7.1 Neustart

Ein Neustart des Decoders kann entweder frisches Aufgleisen oder auch nur ein vorübergehendes Kontaktproblem sein. Zudem weiß der Decoder a priori nicht, ob er von einer Zentrale mit oder ohne Anmeldung kontrolliert wird.

Wenn nach einem Start binnen 700ms keine LOGON_ENABLE-Nachricht empfangen wurde, so soll der Decoder von einer Zentrale ohne Anmeldung ausgehen und darf auf Befehle an die normale Fahrzeugadresse reagieren.

Detektiert ein Decoder beim Start einen sich bereits drehenden Motor, so kann er von einem mangelnden Kontakt ausgehen und darf auf der bisherigen (zugewiesenen Sitzungs-) Adresse Steuerbefehle annehmen.

Wenn ein Decoder die bisherige Zentrale anhand der CID erkennt und sich die Session ID um nicht mehr als 4 inkrementiert hat, so soll der Decoder davon ausgehen, dass er in der Zentrale bekannt ist. Der Decoder kann direkt auf der zugewiesenen Adresse starten und muss nicht erst die Anmeldeprozedur durchlaufen.

Wenn der Decoder dreimalig eine ID15-Antwort auf LOGON_NOW (ohne andere, dazwischen liegende LOGON_* Nachrichten) gesendet hat und nicht per SELECT angesprochen wurde, so soll der Decoder von einer gescheiterten Anmeldung ausgehen und den Fahrbetrieb nicht aufnehmen, sondern mit doppelblinkendem Stirnlicht stehen bleiben. Empfohlenes Blinkmuster: 100ms an, 300ms aus, 100ms an, 1500ms aus.

Wenn der Decoder beim LOGON_ASSIGN die kurze Adresse 0 zugeteilt bekommt, so wurde die Anmeldung von der Zentrale erkannt, die Zentrale will aber den Decoder aktuell nicht ansprechen. Wurde dem Decoder vorher eine andere Adresse zugewiesen, hat der Decoder seinen Zustand abgesehen von der Motoransteuerung beizubehalten ('Parken'). Wurde vorher keine andere Adresse zugewiesen, ist die Zuweisung der Adresse 0 als Fehlerfall zu werten und der Decoder soll diesen Fehlerzustand anzeigen (z.B. blinkendes Fahrlicht). In jedem Fall sind weitere Anmeldungen mit ID15 nicht zulässig.

7.2 Backoff

Sollte ein Decoder nach einer versuchten Anmeldung keine Bestätigung erhalten, so beantwortet er eine bestimmte Anzahl von LOGON_ENABLE-Nachrichten nicht mehr. Die Anzahl der zu ignorierenden Anmeldeaufforderungen bestimmt er mit einer Zufallszahl, in deren Erzeugung die Unique ID eingeht. Die erste Zufallszahl ist aus einem Bereich von 0 bis 7 zu wählen. Wenn der LOGON erneut nicht betätigt wird, so wird die Zahl aus einem Bereich 0 bis 15 gewählt. Wenn der LOGON erneut nicht betätigt wird, so wird die Zahl aus

einem Bereich 0 bis 31 gewählt. Wenn der LOGON erneut nicht betätigt wird, so wird die Zahl aus einem Bereich 0 bis 63 gewählt und wird in Folge nicht mehr weiter vergrößert.

Wenn eine LOGON_ENABLE_NOW empfangen wird, so ignoriert der Decoder den aktuellen Backoff-Wert und versucht sich anzumelden. Der Backoff-Wert wird im Anschluss wieder aus dem Bereich 0 – 7 genommen.

Bei der initialen Berechnung und bei der Fortführung des Backoff-Wertes ist ein echter Zufallsprozess zu verwenden. Sofern ein solcher auf dem Decoder nicht verfügbar ist, kann ersatzweise ein Pseudozufallsprozess dafür verwendet werden, hierbei ist jedoch auf eine entsprechend lange Entropie zu achten (z.B. durch wiederholtes Einrechnen der Unique ID). In [TN-9.2.1.1] gibt es einen entsprechenden, leicht implementierbaren Algorithmusvorschlag.

Anhang A: Verweise auf andere Normen

A.1 Normative Verweise

Die hier aufgeführten Normen sind ganz oder in dem beim Zitat angegebenen Rahmen einzuhalten, um diese Norm zu erfüllen.

[RCN-211] RCN-211 DCC Paketstruktur, Adressbereiche und globale Befehle

[RCN-217] RCN-217 DCC-Rückmeldeprotokoll RailCom

[RCN-225] RCN-225 DCC Konfigurationsvariablen

[TN-218] TN-218 DCC Bild- und Icon-Nummern für DCC-A

A.2 Informative Verweise

Die hier aufgeführten Normen und Dokumente haben rein informativen Charakter und sind nicht Bestandteil dieser Norm.

[S-9.2.1.1] NMRA: S-9.2.1.1 Advanced Extended Packet Formats

[TN-9.2.1.1] NMRA TN-9.2.1.1 Advanced Extended Packet Formats

[S-9.2.2 Appendix A] NMRA: S-9.2.2 Appendix A DCC Manufacturer ID codes

Anhang B: Historie

Datum	Kapitel	Änderungen gegenüber der vorhergehenden Version	Version
27.11.2022	5.6 bis 5.9	Datenräume 4 bis 7 ergänzt. Hinweis auf TN-218 mit dem Bild- und Icon-Nummern	1.1
10.09.2021		Erste Version	1.0

Anhang C: Berechnung CRC

C.1 Polynom

Das in dieser Norm verwendete Polynom stellt einen Kompromiss zwischen Implementierungs- und Übertragungsaufwand versus Datensicherheit dar. Die meisten Übertragungsfehler lassen sich durch eine Timinganalyse des DCC-Signals hinreichend gut ausfiltern, die CRC minimiert das Restrisiko eines Fehlers. Die hier verwendete 8-Bit CRC wird mit den Polynom $x^8+ x^5+ x^4+ 1$ über die Nachricht gebildet, beginnend beim ersten Byte der Nachricht, Initialisierungswert = 0 (bzw. die Nummer des Datenraums, siehe Abschnitt 4.3), nicht invertiert. Dieser Wert wird an der angegebenen Stelle in die DCC-Nachricht eingefügt. Beim Empfänger wird diese Berechnung über die gesamte Nachricht (ohne abschließendes XOR) durchgeführt. Bei Fehlerfreiheit der Übertragung ist das Ergebnis 0 und damit ist das leicht innerhalb der Empfangslogik eines Decoders umsetzbar.

Die Berechnung der CRC verursacht eine XOR-Operation und einen Tabellenzugriff. Die Entscheidung, ob die CRC korrekt ist, ergibt sich aus dem ersten Byte der Nachricht und der Länge und ist damit auch innerhalb der Empfangslogik umsetzbar.

Im Adressraum 253 (Advanced Extended DCC) ist eine identische Regel vorgesehen.

C.2 Berechnungsbeispiel (codeoptimiert)

Zur vereinfachten Berechnung der CRC legt man eine vorberechnete Tabelle im Speicher ab:

```
unsigned char crc_array[256] = {
      0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83,
      0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
      0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e,
      0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc,
      0x23, 0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0,
      0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62,
      0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d,
      0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff,
      0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5,
      0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07,
      0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58,
      0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a,
      0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6,
      0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24,
      0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b,
      0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
      0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f,
      0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd,
      0x11, 0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92,
      0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50,
      0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c,
      0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee,
      0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1,
      0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73,
      0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49,
      0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b,
      0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4,
      0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16,
      0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a,
      0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
      0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7,
      0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,
      };
```

Nun kann man jeweils 8 Divisionsschritte zu einem Tabellenzugriff vereinfachen:

```
crc_value = crc_array[message[i] ^ crc_value];
```

Die Berechnung in der Zentrale ergibt sich dann zu:

```
crc_value = 0;
for (i=0; i<sizeof(DCC_payload); i++) {
      crc_value = crc_array[DCC_payload[i] ^ crc_value];
      }
```

Die Überprüfung im Decoder erfolgt analog:

```
crc_value = 0;
for (i=0; i<sizeof(DCC_with_CRC); i++) {
      crc_value = crc_array[DCC_payload [i] ^ crc_value];
      }
if (crc_value != 0) error();
```

C.3 Berechnungsbeispiel (speicherplatzoptimiert)

Sollte Speicherplatz auf der Implementierungsplattform die beschränkende Ressource sein, so kann die CRC auch ohne Tabelle berechnet werden. Beispielhaft eine Implementierung mittels Bitabfrage:

```
unsigned char crc_calc(unsigned char data) {
      unsigned char result = 0;

      if(data & 1)    result ^= 0x5e;
      if(data & 2)    result ^= 0xbc;
      if(data & 4)    result ^= 0x61;
      if(data & 8)    result ^= 0xc2;
      if(data & 0x10) result ^= 0x9d;
      if(data & 0x20) result ^= 0x23;
      if(data & 0x40) result ^= 0x46;
      if(data & 0x80) result ^= 0x8c;
      return result;
}
```

Die Berechnung ergibt sich dann zu:

```
crc_value = 0;
for (i=0; i<sizeof(DCC_payload); i++) {
      crc_value = crc_calc(DCC_payload[i] ^ crc_value];
      }
```

C.4 Berechnungsbeispiel (Größen- und codeoptimiert)

Als drittes Beispiel eine Aufteilung in Tabellenverfahren (mit kleineren Tabellen) und kurzem Code:

```
unsigned char crc_nibble1[16] = {
      0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83,
      0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
      };

unsigned char crcnibble2[16] = {
      0x00, 0x9d, 0x23, 0xbe, 0x46, 0xdb, 0x65, 0xf8,
      0x8c, 0x11, 0xaf, 0x32, 0xca, 0x57, 0xe9, 0x74
};
```

```
unsigned char crc_calc(unsigned char data) {
      unsigned char result = 0;

      result = crc_nibble1[data&0xf] ^ crc_nibble2[data>>4];
      return result;
}
```

C.5 Beispieldaten

Zur (vereinfachten) Implementierungskontrolle hier ein Zahlenbeispiel:
`0x0B 0x0A 0x00 0x00 0x8E 0x40 0x00 0x0D 0x67 0x00 0x01 0x00` ergibt ein CRC-Byte `0x4C`.

Anhang D: Adressen

Bei der Übermittlung der Wunschadresse sowie bei der Adresszuweisung mittels LOGON_ASSIGN werden die (historisch gewachsenen) Adressen auf 14 Bit abgebildet.

Diese Zuordnung wird wie folgt vorgenommen: (A13-A8, A7-A0)

A13.. A8	Decoder	Beschreibung
0..0x27 (0.. 39)	Fahrzeugdecoder, lange Adresse	Adresse ergibt sich zu A13...A0 CV17 = A13...A8 + 0xC0 CV18 = A7...A0
0x28..0x2F (40..47)	Erweiterter Zubehördecoder	Adresse ergibt sich zu A10...A0; das ist die lineare Adresse nach RCN213.
0x30..0x37 (48..55)	Standard Zubehördecoder	Adresse ergibt sich zu A10...A0; das ist die lineare Adresse nach RCN213.
0x38	Fahrzeugdecoder, kurze Adresse	Adresse ergibt sich zu A6...A0 (entspricht CV1)
0x39..0x3E	-	reserviert
0x3F	-	Decoder ist im FW-Updatemode, nur Firmwareupdate möglich

Informativ: Bei mobilen Decodern soll der Übergang von kurzer zu langer Adresse von 127 auf 128 erfolgen. Beim höherwertigen Byte sind die oberen beiden Bits reserviert und mit 1 zu belegen. Die Übertragung einer kurzen Adresse wird so z.B. zu 0xF8 im höherwertigen Byte und der kurzen Adresse im niederwertigen Byte.
