

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Autor: Knut Schwichtenberg

Team: Stefan Krauß, Knut Schwichtenberg, Erwin Stegmaier

Version 1.1 vom 20091018

Hinweis

Aus privater Wissbegier wurde von einigen technisch interessierten Modellbahnern (dem Team) das CAN-Bus-Protokoll erforscht. In diesem Dokument werden die entsprechenden Beobachtungen und Schlussfolgerungen beschrieben; es gibt die private Meinung der Autoren wieder. Die vorliegende Beschreibung ist keine offizielle Formatspezifikation und nicht von denen an der Entwicklung oder Vermarktung des CAN-Bus-Protokolls beteiligten Firmen autorisiert. Das Dokument richtet sich an technisch interessierte Modellbahner und sollte als journalistische Arbeit verstanden werden. Es werden keine kommerziellen Interessen verfolgt.

Haftungsausschluss

Autor und Team schließen jegliche Ansprüche durch Schäden aus, die durch falsche, unvollständige oder ungenaue Informationen in diesem Dokument entstehen könnten. Insbesondere wird darauf hingewiesen, dass alle Informationen nur informativ zu verstehen sind, eine Eignung für einen bestimmten Zweck nicht garantiert werden kann und eventuell Schutzrechte Dritter betroffen sein können. Es ist damit zu rechnen, dass das Dokument unvollständig und teilweise nicht korrekt ist.

Urheberrechte

Das Urheberrecht am vorliegenden Dokument liegt beim Autor. Alle Texte, Zeichnungen und Bilder dürfen nicht ohne schriftliches Einverständnis des Autors ganz oder in Teilen verwendet werden. Der Autor ist über die folgende E-Mail-Adresse bzw. Internet-Seite zu erreichen:

[kschwi at web \(dot\) de](mailto:kschwi@web.de)
<http://www.skrauss.de>

Die unveränderte PDF-Datei darf privat weitergegeben und zu privaten Zwecken ausgedruckt werden. Eine kommerzielle Verwendung ist ausgeschlossen.

Dieser Urheberrechtshinweis bezieht sich nicht auf die dargestellten Sachverhalte und technischen Verfahren. Diese können durch Urheber-, Patent- oder andere Schutzrechte Dritter geschützt sein. Die Wiedergabe erfolgt in diesem Dokument nur zu Informationszwecken.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 1 von 56

Inhaltsverzeichnis

1	Markenrechte	5
2	Abkürzungen	6
3	Definitionen	7
4	Allgemeines.....	8
4.1	Projektziele.....	8
4.2	Patente.....	8
4.3	Projektteile	8
5	CAN-Bus Überblick	9
5.1	Elektrische Parameter beider CAN-Protokolle	9
5.2	Genereller Aufbau von CAN-Frames.....	9
5.3	Anwenderprotokoll M-CAN / E-CAN.....	10
5.4	Generelle Funktionsweise des M-CAN.....	10
5.4.1	Adresse eines CAN-Bus-Teilnehmers.....	11
6	Phasen der Kommunikation	13
6.1	CAN-Bus belegt.....	14
6.2	Anmeldung	14
6.3	Aufrüst- / Betriebsphase	20
6.3.1	Frame-ID-Struktur nach Anmeldung	20
6.3.2	Aufrüstungssequenz	22
6.4	Slave Description / Lokstack	23
6.4.1	Request Lokstackgröße	24
6.4.2	Request Slave-Name	24
6.5	System-Status-Handle	25
6.6	System-Status-Austausch	26
6.7	Lokstack-Austausch	27
6.7.1	Get Lok-Typ	29
6.7.2	Request Schienenformat.....	30
6.7.3	Funktionen f1- / Funktionswerte	31
6.7.4	Übertragung des Loknamens	33
6.7.5	Request / Set Lok-Zuordnung	33
6.7.6	Bisher ungeklärte Telegramme	34
7	Betrieb.....	35
7.1	Allgemeines.....	35
7.2	Zyklische Überwachung	35
7.2.1	Slave angeschlossen	35
7.2.2	Slave entfernt / Verzögerung im Zyklus	36
7.3	Steuern.....	36
7.4	Status Änderung Stopp / Go / Überstrom.....	37
7.5	Umtaufen eines Slaves	38
7.6	Zuordnung /Abmelden einer Lok	39
7.7	Hinzufügen / Löschen einer Adresse aus dem Lokstack.....	41
7.7.1	Löschen einer Adresse aus dem Lokstack.....	41
7.7.2	Hinzufügen einer Adresse zum Lokstack	42
8	Herunterfahren des Masters.....	43

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

9	Flashen einer MS1	44
9.1	Überblick	44
9.2	Setzen des Flashbits im Anmelde-Frame.....	44
9.3	ID-Aufteilung beim Flashen	45
9.4	Signalisierung der Flashbereitschaft an die Zentrale.....	45
9.5	ID-Abfragen.....	46
9.6	Flash-Abbruch	47
9.7	Flash-Freigabe:	47
9.8	Flashen-Ende-Frame	49
9.9	Beispiel eines Flashvorgangs.....	50
9.10	Das BCI-File-Format	51
10	Offene Punkte.....	52
11	Telegramm Trace.....	53

Abbildungsverzeichnis

Abbildung 6-1 Kommunikationsphasen.....	13
Abbildung 6-2 Sequenz einer Slave-Anmeldung	15
Abbildung 6-3 Sequenz einer Slave-Aufrüstung	23
Abbildung 6-4 Sequenz der Datenübertragung für eine Lok.....	29

1 Markenrechte

Märklin und mfx sind eingetragene Marken der Firma Gebrüder Märklin & Cie. GmbH, 73033 Göppingen, Deutschland. Trix ist eine eingetragene Marke der Märklin Holding GmbH, 73037 Göppingen, Deutschland.

ESU ist eine eingetragene Marke der Firma ESU electronic solutions ulm GmbH & Co. KG, 89081 Ulm, Deutschland.

Motorola ist eine eingetragene Marke der Motorola Inc., Schaumburg, Illinois, USA.

Bosch ist ein eingetragenes Warenzeichen der Robert Bosch GmbH, Deutschland

Alle anderen in diesem Dokument benutzten Warenzeichen sind Eigentum der entsprechenden Inhaber.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 5 von 56

2 Abkürzungen

In der folgenden Tabelle sind alle Abkürzungen, die im Weiteren benutzt werden in alphabetisch sortierter Reihenfolge aufgeführt.

Abkürzung	Bedeutung
0b?	Darstellung eines bzw. einer Gruppe von Bits, d.h. einer binären Information im Wertebereich 0 oder 1 je Stelle
0x????	Darstellung einer hexadezimalen Zahl. In diesem Beispiel mit 4 Stellen in dezimalen Zahlenraum von 0 bis 65535.
CAN	Controller Area Network
CS	Control Station (hier: die von ESU entwickelte Variante auch CS1 genannt)
CS2	Central Station 2 (Märklin-Entwicklung)
DCC	Digital Command Control (Digital Standard der NMRA)
DLC	Data Length Code = Anzahl der Nutzdatenbytes im Frame
E-CAN	ECoS CAN siehe ECoSlink
ECoS	Markenname der Steuerungszentral der Firma ESU
ECoSlink	Markenname des CAN-Bussprotokolls der Firma ESU
ESU	Elektronik Solution Ulm
HW	Hardware
ID	Identification (siehe auch Kapitel „3 Definitionen“)
LSB	Least Significant Bit entspricht der Wertigkeit 2^0
M-CAN	Märklin- / MS1-Variante des CAN-Protokolls
MID	Anmeldungsnummer (Modifier-ID)
MM	Märklin Motorola Schienensignal
Mfx	Hier: Nur neues Schienenformat der Firma Märklin incl. RDS-Rückmeldung
mfx-UID	UID eines mfx-Dekoders
MS / MS1	Mobile Station 1. Version. Hierbei handelt es sich allein um die von Märklin vertriebene Mobile Station. Die von Trix vertriebene Version sieht gleich aus, basiert nicht auf einem CAN-Bus zur Kommunikation.
MSB	Most Significant Bit
NMRA	National Model Railroad Association
Rx/Tx	Rx = Empfangen, Tx = Senden
UID	Unique Identification (eineindeutige Seriennummer eines Endgeräts z.B. MS1, Lok-Dekoder, ...)

3 Definitionen

In der folgenden Dokumentation werden Begriffe benutzt, die in einem technischen Kontext definiert sind. Leider sind manche Begriffe mehrfach belegt (z.B. ID), so dass in der folgenden Definition versucht hier für dieses Dokument eine eindeutige Sprache zu finden.

Begriff	Definition
Flashen	Programmieren eines Flash-Speichers
Frames oder CAN-Frame	CAN-Bus-Telegramm bestehend aus ID und Daten
ID	29 Bit lange Identifikation eines CAN-Frames
Master	In diesem Zusammenhang: Hardware, welche die Adressen der CAN-Bus-Teilnehmer vergibt und die Wandlung von logischen Adressen in das Schienenformat vornimmt.
Selectrix	Digital-Format der Firma Trix
Slave	Steuereinrichtung für eine bestimmte Aufgabe am CAN-Bus
Sniffer	Multi-Protokoll-Konverter, der verschiedenen Digital-Signale interpretieren kann und entsprechende CAN-Bus-Frames daraus generiert.
UID	Eindeutige Gerätekenung. In diesem Dokument wird UID als eindeutige Kennung eines CAN-Bus-Teilnehmers verwendet.

4 Allgemeines

4.1 Projektziele

Aufgrund der Schweigsamkeit der Firma ESU bzgl. der von ihnen verwendeten Protokolle und Funktionen dient diese Dokumentation der Zusammenfassung und Auswertung vieler Messergebnisse am CAN-Bus. Sie soll Interessierten helfen, selbstgebaute Geräte an die CS1, MS1 Master bzw. mit bestimmten funktionalen Einschränkungen an eine ECoS anzuschließen. Es wird damit auch möglich, an bestehende Projekte eine MS1 als Loksteuerung anzuschließen.

4.2 Patente

Bosch ist der Lizenzinhaber der CAN-Bus-Patente. Anwender des CAN-Busses können im Gegensatz zu Entwicklern von CAN-Bus-Hardware diese frei nutzen, da Lizenzgebühren in den CAN-Controller-Kosten enthalten sind.

4.3 Projektteile

Das gesamte Projekt besteht im Wesentlichen aus folgenden Teilen:

- a) Das CAN-Protokoll als Kommunikation zwischen der MS1-Slave und einer Zentrale.
- b) Das CAN-Protokoll als Kommunikation zwischen ECoS und anderen Endgeräten (z.B. Sniffer).

In diesem Teil der Dokumentation wird das CAN-Protokoll zu einer Märklin-Zentrale (genauer: zwischen MS1 und einer Zentrale), im Folgenden M-CAN genannt, beschrieben. Wie sich nun herausgestellt hat, verwendet die CS2 auch das M-CAN-Protokoll zwischen MS1 und CS2.

Da nun Märklin auch die 2. Version der MobileStation angekündigt hat, wird die alte MS (1. Ausführung) MS1 genannt und die neuere Ausführung MS2. Diese Dokumentation ist jedoch schon vor dem Bekanntwerden einer neuen MobileStation (MS2) entstanden. Da der M-CAN bisher ausschließlich für die Kommunikation zwischen einer Zentrale und einer MS1 verwendet wurde, bezeichnen wir hier die MS1 nur mit MS, was gleichbedeutend mit der Bezeichnung MS1 sein soll. Die MS2 verwendet das neue Märklin CAN-Busprotokoll zur Kommunikation.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 8 von 56

5 CAN-Bus Überblick

Es gibt drei CAN-Bus-Protokolle:

1. Das Protokoll zwischen MS1 und CS1 bzw. ECoS.
Es wurde mit der Märklin-MS1 im Rahmen der damals neuen Gerätegeneration von „Märklin Systems“ eingeführt und wird hier deshalb M-CAN genannt. Dieses Protokoll wird in diesem Dokument beschrieben.
2. Das Protokoll zwischen CS1/ECoS und dem Sniffer-Modul oder dem Booster.
Es wurde von ESU mit der ECoS eingeführt und wird hier deshalb als E-CAN bezeichnet. Das Protokoll wird inzwischen auch von der CS1 unterstützt, die beim Update das Sniffer-Modul der ECoS erhalten hatte.
3. Das Protokoll der CS2/MS2.
Dieses Protokoll hat nichts mit den anderen beiden zu tun und wird hier nicht beschrieben, ist aber von Märklin veröffentlicht worden.

Zum Teil können die Protokolle parallel zueinander auf einem Bus verwendet werden. Das M-CAN-Protokoll kann sowohl parallel zum E-CAN genutzt werden. Hier gibt es lediglich Einschränkungen im Funktionsumfang und in der Adressierung. Ebenso kann es parallel zum Protokoll der CS2/MS2 genutzt werden. E-CAN und das Protokoll der CS2/MS2 schließen sich jedoch aus.

Aufgrund der weitreichenden Definition der CAN-Bus Definition werden an dieser Stelle nur Daten auf Anwenderebene betrachtet. Alle administrativen / organisatorischen Protokollanteile bzw. Busarbitrierung werden durch die CAN-HW geliefert und müssen daher vom Anwenderprotokoll nicht gehandhabt werden.

5.1 Elektrische Parameter beider CAN-Protokolle

Ähnlich wie bei dem RS-485 Bus (XBus von Lenz) werden auch am CAN-Bus die Signale differentiell übertragen. Gemäß der gewählten Kabel, Stecker und Buchsen und der gewählten Baudrate sollten mindestens 100m problemlos funktionieren.

Als CAN Low-Level-Protokoll wird CAN 2.0B verwendet, es stehen also 29-Bit-IDs zur Verfügung. Baudrate sind 250 kBaud.

Es erfolgt eine automatische Terminierung auf dem letzten Gerät. Das gilt auch, wenn ein CAN-Bus Verteiler vorhanden ist. Ist nur eine Zentrale am Bus, terminiert sich diese Zentrale selbst.

5.2 Genereller Aufbau von CAN-Frames

Jeder CAN-Frame besteht aus einer ID, einigen Steuerflags, einer Angabe der Nutzdatenlängen und aus 0 bis 8 Datenbytes.

Die ID kann für beliebige Zwecke definiert werden. Durch die Methode der ID-Vergabe wird also letztendlich die Priorität des Frames bestimmt, so dass sichergestellt ist, dass eine ID nur von einem einzigen Sender auf den Bus gelegt wird. Bei Kollisionen „überlebt“ der Frame mit der niedrigsten ID.

Die Nutzdaten können zum Übertragen beliebiger Informationen genutzt werden. Die Nutzdatenlänge ist variabel, was vom M-CAN wie vom E-CAN-Protokoll auch benutzt wird.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 9 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Im Weiteren werden Frames bei der ID-Analyse wie folgt dargestellt (Beispiel aus der Anmeldung):

Frame							
29 Bit ID						DLC	Max. 8 Daten Bytes
Priorität	UID	Anmelde- stufe	MID	Richtung = gesendet von	Knoten- nummer	8	Data [0 .. 7]
Bit 28 .. 26	Bit 25 .. 16	Bit 15 .. 13	Bit 12 .. 8	Bit 7	Bit 6 .. 0		

5.3 Anwenderprotokoll M-CAN / E-CAN

Physikalisch und in seiner Grundkonzeption ist der CAN-Bus ein Broadcast-Protokoll. Die Frames enthalten keine Adressen, sondern beschreiben über die ID nur den Inhalt der Daten. Der Sender legt den Frame auf den Bus, den dann jeder angeschlossene Knoten empfangen und auswerten kann, z.B. ID 55 = Geschwindigkeit im Rad vorne links. In dem Beispiel sendet also der Geschwindigkeitsgeber je nach Definition (entweder zyklisch oder bei Änderung) die Geschwindigkeit und jeder interessierte Empfänger wertet über Empfangsfilter aus, ob er an dieser Meldung interessiert ist oder nicht. Die IDs werden deshalb so gruppiert, dass sie leicht gefiltert werden können. Außerdem wird über die ID die Priorität auf dem Bus festgelegt. Eine niedrigere ID hat eine höhere Priorität.

Die übliche Nutzung des CAN-Busses im Automobilbereich entsprach nicht den Anforderungen von ESU für eine Modelbahnsteuerung. Beim M-CAN- und E-CAN-Protokoll wurde mit den CAN-Mechanismen ein adressorientiertes Protokoll aufgesetzt. Das M-CAN-/E-CAN-Protokoll hat daher die folgenden Eigenschaften:

- Adressorientierung: Jeder Kommunikationsteilnehmer hat eine eindeutige Adresse
- Master-Slave-Topologie: Es gibt einen Master (MS1-Master, CS1 / ECoS und CS2), der die Teilnehmer der Kommunikation (Slaves) zyklisch pollt bzw. ihnen die Steuer-Informationen bei Bedarf zukommen lässt.
- M-CAN/E-CAN ist ein zustandsorientiertes Protokoll (basiert auf dynamisch zugewiesenen Objekten). Nach einer Anmeldung eines Slaves bei einem Master ist es möglich Objekte (Züge, Weichen,...) zu steuern bzw. über Änderungen informiert zu werden. Geht ein Slave offline (z.B. abgezogen), muss er sich beim Master wieder erneut anmelden.

5.4 Generelle Funktionsweise des M-CAN

Bei den durchgeführten Messungen wurden 4 Frame-Gruppen beobachtet. Diese sind

1. Initialisierungsgruppe
Damit wird die Verbindung zwischen zwei Geräten hergestellt und dem Teilnehmer eine eindeutige Knotennummer gegeben.
2. Aufrüst-Gruppe
Darin sind z.B.: die Übertragung der Loknamen und -funktionen usw. enthalten.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 10 von 56

3. Steuer-Gruppe
Damit werden die Fahrbefehle und Funktionsbefehle übertragen.
4. Software Update
Damit wird eine neue Firmware zur MS1 übertragen.

Derzeit sind nur die ersten drei Frame-Gruppen detailliert analysiert worden und werden im Folgenden dargestellt. Die vierte Gruppe ist nur teilweise analysiert. Dieses Teilergebnis ist ebenfalls dokumentiert.

Wie oben aufgeführt, wird durch das Anwenderprotokoll mit Adressen (hier sind Lok-, Weichen- und andere Adressen gemeint) gearbeitet. Adressen gibt es auf den unterschiedlichen Ebenen. Die Gemeinsamkeit aller Adressen ist, dass spätestens nach einer Anmeldung nur noch logische Adressen genutzt werden. Die Umsetzung zwischen logischer und physikalischer Adresse findet im Master statt.

Nur die Zentrale kennt den Zusammenhang zwischen physikalischer Lokadresse (Digital-Adresse bei DCC- und MM*-Dekodern) und dem Objekt-Handle (hier die logische Adresse) in den Lokstacks der angemeldeten Fahrregler. Bei mfx-Loks ist sogar noch eine weitere logische Ebene dazwischen, weil auf der Schiene mit logischen (zugewiesenen) Loknummern und nicht mit der mfx-UID gefahren wird. Die MSse müssen von der gefahrenen Lok nicht einmal wissen, ob es sich um eine MM, DCC, mfx oder Selectrix-Lok handelt, da alle Lokobjekte logisch gleiche Eigenschaften haben, und sich nur in der Anzahl der Funktionen und Loknamen unterscheiden. Erst mit der Version 3.0 der ECoS Firmware wurde das Schienenprotokollkennung an eine angeschlossene MS1 übertragen. Gefahren wird auf dem M-CAN-Bus grundsätzlich normiert auf 127 Fahrstufen.

5.4.1 Adresse eines CAN-Bus-Teilnehmers

Jede CAN-Bus Hardware verfügt über eine eindeutige Identifizierung, im weiteren UID genannt. Der Master weist im Rahmen der Initialisierungsgruppe jedem CAN-Teilnehmer eine Knotennummer im Bereich (1 – 125) zu. Bei jeder Kommunikation im Normalbetrieb mit dem Master wird die Knotennummer als Teil in der ID verwendet. Bei der Anmeldung selber wird die reservierte Knotennummer 126 verwendet. Sollte es zum Flashen einer MS1 kommen, wird die dafür reservierte Knotennummer 127 verwendet. Der vorgesehene 7-Bit-Bereich ist damit vollständig ausgefüllt.

Knotennummer / -Bereich	Bedeutung
0	Master Knotennummer für die Verbindung mit dem Slave-Knoten 1 (Nur zwischen MS1-Master und MS1-Slave verwendet)
1 ... 125	Nur ungrade Zahlen: Endgeräte (Slave / MS1)
2 ... 124	Nur gerade Zahlen: Adresse des Masters
126	Knotennummer während der Anmeldung
127	Knotennummer während des Flashens

In jedem CAN-Frame, bis auf die Ausnahme der Anmeldung und dem Flashen, enthält die CAN-ID immer die Sender-Knotennummer (7 Bits). Nicht angemeldete CAN-Knoten können nicht mit dem Master kommunizieren und werden von allen Teilnehmern ignoriert.

Die Master-Knotennummer ist die Knotennummer, welche sich die MS1 als ihren persönlichen Master für diese Anmeldung merkt. Sie ist eine Nummer niedriger als die Slave-

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 11 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Knotennummer, welche der MS1 bei der Anmeldung zugewiesen wird. Jede angemeldete MS1 belegt somit 2 von den max. möglichen 126 Nummern, obwohl das eigentlich nicht nötig wäre. Weitere Details zur Struktur der Frame-ID siehe Kapitel „6.3.1 Frame-ID-Struktur nach Anmeldung“.

Mit einem Filter empfängt jede MS1 nur die Frames, welche in der ID auch die Knotennummer ihres zugewiesenen Masters enthält. Ein ID-Filter in der CAN-HW ist somit nutzbringend einsetzbar. Somit kann jeder CAN-Knoten vom Master explizit angesprochen werden, ohne dass der Slave-Knoten aufwändig unterscheiden muss, ob die Information auch wirklich für ihn selbst ist. In der Zentrale merkt man sich eine einmal angemeldete MS1 sowie die zugewiesenen Loks anhand deren UID dauerhaft. Die zugewiesenen Knotennummern wie auch alle anderen Objekt-Handle sind nur für die aktuell durchgeführte Anmeldung gültig. Der Slave merkt sich diese nur im Betrieb, weil sie sich nach jedem Einschalten, was eine Neuansmeldung am Bus nach sich zieht, ändern kann, wenn neue CAN-Teilnehmer mit einer niedrigeren UID dazukommen.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 12 von 56

6 Phasen der Kommunikation

Das folgende Sequenzdiagramm stellt eine Übersicht über die Kommunikationsphasen mit ihren Elementen dar. Die Pfeilrichtung zeigt an, welcher Kommunikationsteilnehmer das Element initiiert. Detailliert werden die einzelnen Elemente in den folgenden Kapiteln beschrieben.

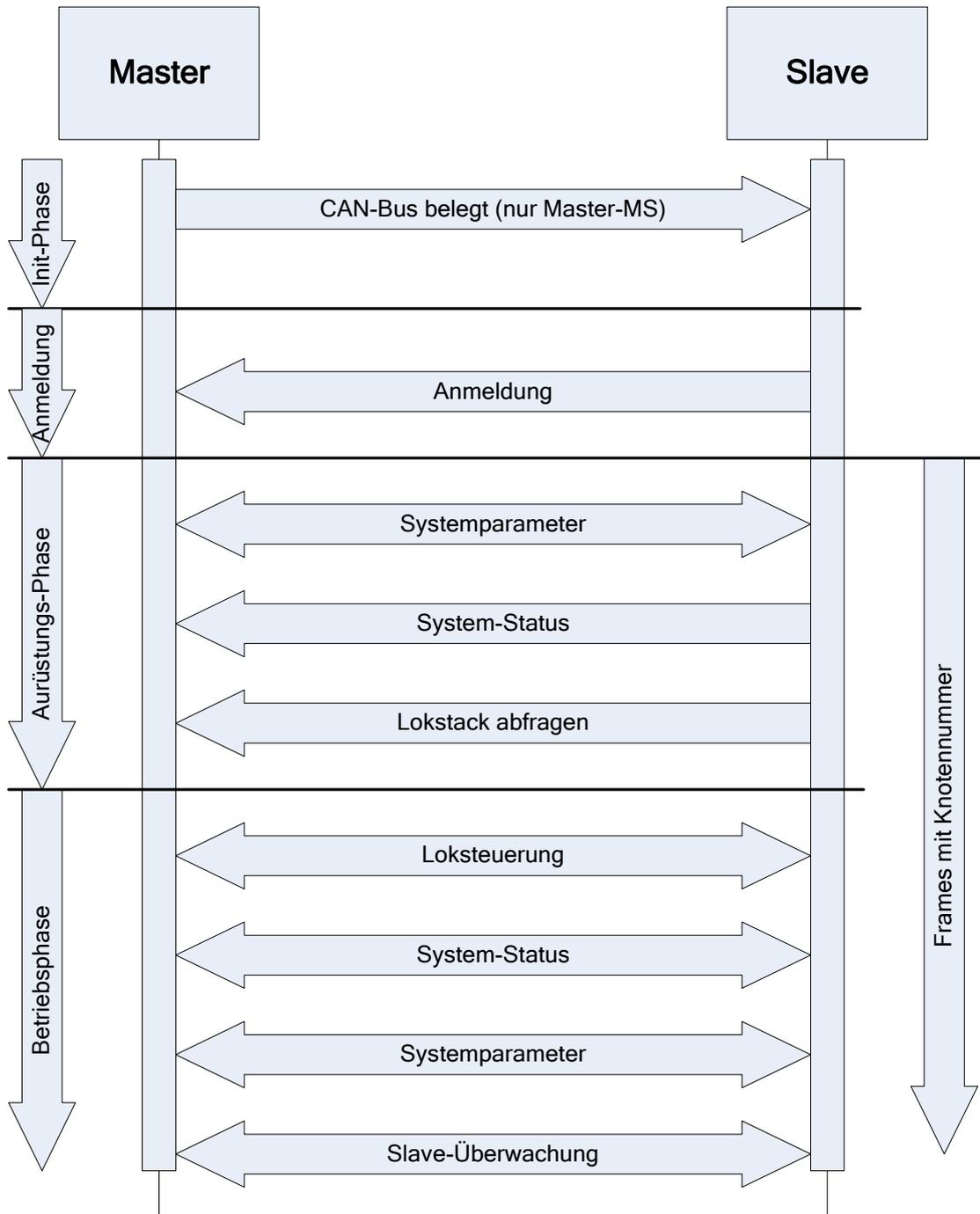


Abbildung 6-1 Kommunikationsphasen

6.1 CAN-Bus belegt

Mit diesem Frame kontrolliert der Master ob mindestens ein Slave am CAN-Bus angeschlossen ist. Dieser Frame wurde nur bei der Master-MS1 nachgewiesen. Seine Bedeutung ist unklar und vor allem für den Betrieb irrelevant! CS1 und ECoS verwenden ihn nicht. Betrachtet man den Befehlsaufbau, ist dieser Frame ein Ping-Befehl nur ohne sinnvolle Nutzdaten und mit höchster Priorität.

ID	0x380
Initiator	Master
Übertragungsverhalten	Burst / zyklisch
Anzahl Burst Wiederholungen	8

Der Master sendet diesen Frame 8-mal. Die Struktur sieht folgendermaßen aus:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		
0b000	0x0000	0b111	0b00000	8	0, 0, 0, 0, 0, 0, 0, 0

Wenn ein anderer CAN-Teilnehmer, also die CAN-HW, diesen Frame mit einem ACK auf dem Physical Layer quittiert, sich aber sonst nicht meldet oder irgendeinen CAN-Frame sendet, sendet der Master unbeirrt diesen Frame fortwährend weiter.

6.2 Anmeldung

Ziel der Anmeldung ist es, dem Master einen Slave bekannt zu machen und das Knotennummern-Paar festzulegen. Ein weiteres Ziel kann es sein, einen Firmware-Update zu initiieren. Wird bei einer MS1 aus Versehen ein Firmware-Update initiiert, dann gibt es keine Rückkehr mehr in den Betriebsmodus. Anscheinend wird ein Bootloader initiiert, der resetfest auf dem Firmware-Update beharrt. Ein Anschluss dieser MS1 an eine CS1 / ECoS ist dann notwendig.

Mit Einschalten der Betriebsspannung an einem Slave beginnt dieser mit der Anmeldung. Dazu sind keine weiteren Vorbereitungen durch den Master notwendig. Da bei diesem Vorgang diverse Frames ausgetauscht werden, wird der Ablauf im folgenden Sequenzdiagramm dargestellt.

In den runden Klammern werden dabei die sich ändernden Werte im Frame aufgeführt. Teilweise sind die Inhalte unbekannt, so dass nur der Begriff „Daten“ verwendet wird. Ist nicht beabsichtigt, dass die MS1 geflashed wird, sind die Angaben wie BLayout, HW-Version, SW-Versionen usw. irrelevant. Sie werden bei einer Anmeldung an einer CS2 mit 0x00 von der CS2 vorbesetzt. Eine Zentrale wie die ECoS bzw. CS1 erkennt an den von einer MS1 übergebenen Versionen, ob ein Firmware-Update notwendig ist.

In der folgenden Grafik sind die Telegramme vom Master zum Slave als „Quittung“ benannt. „Quittung“ in diesem Zusammenhang steht für ein Antwort-Telegramm der Software. Das hat nichts mit den Transport-Quittungen (Acknowledge) der CAN-Bus-Hardware zu tun. Bei den in Klammern stehenden Informationen sind nicht alle Infos aufgeführt, die übertragen werden. Es dient lediglich als Sammelbegriff.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 14 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

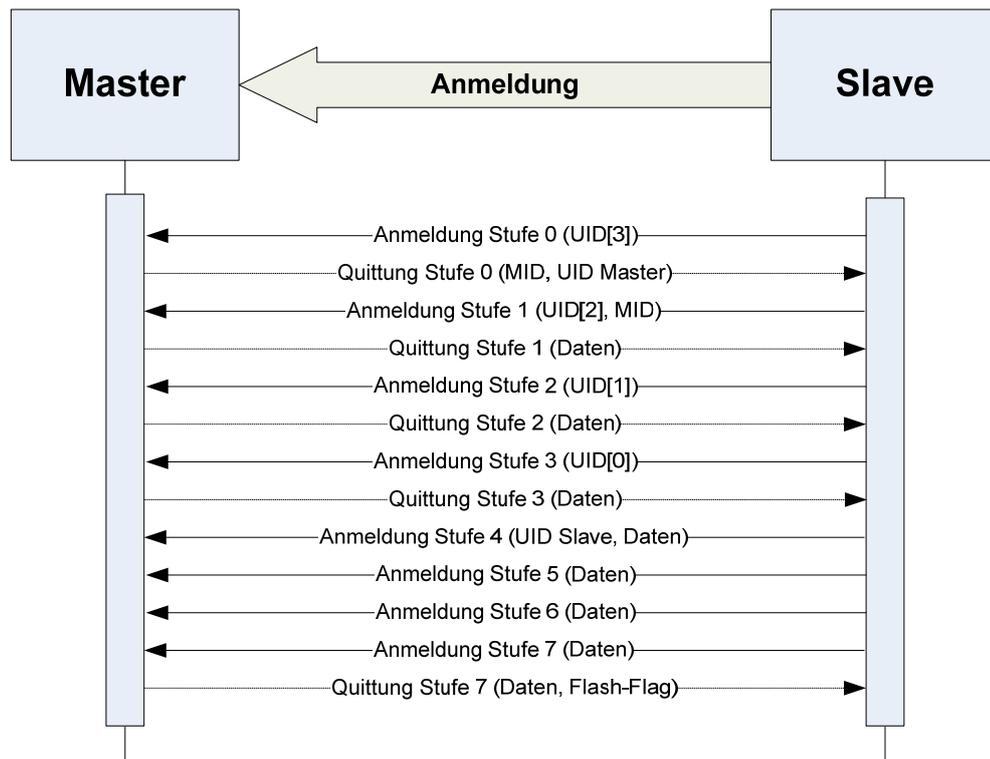


Abbildung 6-2 Sequenz einer Slave-Anmeldung¹

Die Quittung von Stufe 7 vom Master an den Slave kann auch schon nach der Anmeldung Stufe 4 gesendet werden.

In der folgenden Tabelle werden die einzelnen Telegramme detailliert aufgeführt.

Bei den ersten vier Telegrammen vom Slave in Richtung Master werden keine Nutzdaten übertragen. Die Seriennummer des Slave (UID) wird bytewise beginnend mit dem höchstwertigen Byte als Bestandteil der Frame-ID übertragen. Dieses Vorgehen wird benutzt, um bei einer gleichzeitigen Anmeldung mehrerer Slaves an einem Master die CAN-Bus Arbitrierung zur Sequentialisierung zu nutzen. Dies kommt dann vor, wenn die Zentral mit mehreren daran angeschlossenen Slaves eingeschaltet wird. Da die UID in 4 Teilen übertragen wird, kann eine unterschiedliche UID zu gleichen CAN-Frame führen – z.B. UID 0000fffe und 0000ffff. Daher reicht aber der reine CAN-Mechanismus, d.h. Sequentialisierung aufgrund eines unterschiedlichen UID-Bytes im CAN-Frame, nicht aus. Betrachten wir diese Problem mal im Detail.

1. CAN-Frame mit unterschiedlichem UID-Teil:
kein Problem, durch CAN-Hardware unterschieden und priorisiert (auf letzteres kommt es aber eigentlich nicht an, wichtig ist nur, dass zwei Frames rüber kommen, auch wenn diese zufällig gleichzeitig gesendet werden). Der Master vergibt zwei unterschiedliche Knotennummern und es geht kollisionsfrei(!) mit zwei durch die Knotennummern unterschiedene Anmeldestränge weiter.
2. Es ergibt sich (zufällig) eine gleiche CAN-ID,
weil es noch keine Unterscheidung gab und beide Slaves denselben UID-Teil haben und sich auch noch gleichzeitig anmelden! Nun haben wir das, was auf dem CAN-Bus eigentlich nicht passieren darf, nämlich zwei CAN-Frames mit der gleichen ID zur gleichen Zeit. Hier funktioniert die Kollisionsauflösung des CAN-Busses

¹ Mit der Notation UID[x] wird entsprechend der C-Syntax das Element 'x' aus dem Array UID adressiert

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

nicht, es wird von niemand bemerkt, dass es für diesen Frame zwei Sender gab. Nur in diesem Fall ist es notwendig, dass keine Nutzdaten gesendet werden, weil die sich gegenseitig stören würden, was nicht zur Kollisionserkennung herangezogen würde. Bemerkte es schon, aber als Daten- und CRC-Fehler. Das ist aber eine andere Klasse, nämlich ein Übertragungsfehler und rührt nicht notwendigerweise aus einer Kollision her. Zurück zur Anmeldung: In diesem Fall geht es eben für beide Slaves ein Stück gemeinsam weiter (mit der Anmelde-Knotennummer 126), was ja auch nicht schlimm ist. Das bedeutet aber auch, dass sich die Knotennummer für einen der beiden Anmelder später noch ändern wird!

In der folgenden Tabelle wird das ID-Bit 7 (Richtung) der Lesbarkeit wegen symbolisch dargestellt. Folgende Zuordnung gilt:

- 0 vom Slave
- 1 vom Master

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 16 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	UID	Stufe	MID = Veränderungs- zähler	Richtung gesendet von	Knoten- nummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 16	Bit 15 .. 13	Bit 12 .. 8	Bit 7	Bit 6 .. 0									
0b111	UID Slave [3]	0b000	0	Slave	126	0	--	--	--	--	--	--	--	--
0b111	UID Slave [3]	0b000	0 .. 31	Master	126	8	UID[3] Master	UID[2] Master	UID[1] Master	UID[0] Master	0x00	0x01	0x00	0x00 = MS-Master 0x01 = CS- + E- Master
0b111	UID Slave [2]	0b001	Übernahme 0 .. 31 vom Master	Slave	126	0	--	--	--	--	--	--	--	--
0b111	UID Slave [2]	0b001	0 .. 31 von Master	Master	126	8	0x00	0x00	0x00	NMRA Hersteller-ID 0x97 = ESU	0x01 MS-Master 0x00 CS-Master 0x00 EC-Master	0x00 MS-Master 0x00 CS-Master 0x00 EC-Master	0xC0 MS-Master 0x01 CS-Master 0x01 EC-Master	0x10 MS-Master 0x01 CS-Master 0x03 EC-Master
0b111	UID Slave [1]	0b010	0 .. 31 von Master	Slave	126	0	--	--	--	--	--	--	--	--
0b111	UID Slave [1]	0b010	0 .. 31 von Master	Master	126	8	BL [0]: 0x01	BL [1]: 0x00	0x00	?? Power ?? 0x69	0x00	0x00	? HW-Version ? 0x40	? HW-Version ? 0x10
0b111	UID Slave [0]	0b011	0 .. 31 von Master	Slave	126	0	--	--	--	--	--	--	--	--
0b111	UID Slave [0]	0b011	0 .. 31 von Master	Master	126	8	0x00	0x00	0x40	0x10	? SW-Version ? 0x01	? SW-Version ? 0x03	0x00	?? MS-Master 0x84 ?? ECoS: 0x22 ?? CS 0x86
0b111	0	0b100	0 .. 31 von Master	Slave	126	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	0x00	0x02	0x00	0x03
0b111	0	0b101	0 .. 31 von Master	Slave	126	8	0x00	0x00	0x00	Hersteller-ID 0x97	0x01	0x00	0x40	0x10
0b111	0	0b110	0 .. 31 von Master	Slave	126	8	0x01	0x00	0x00	0x67	0x00	0x00	0x40	0x10
0b111	0	0b111	0 .. 31 von Master	Slave	126	8	0x00	0x00	0x40	0x10	0x01	0x03	0x00	0x84
0b111	0	0b111	0 .. 31 von Master	Master	126	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	Start-Obj.-Hand 0x00	Start-Obj.-Hand 0x01	Bit 0-6 Knoten- nummer Bit 7 = Multi- Slave	0x00

Wird vom Master in der Quittung Stufe 7 in den Daten[7] der Wert 0x01 übertragen, geht eine MS sofort in den Programmiermodus. **Vorsicht!** Erkennt der Master anhand der übergebenen Versionen einen Update Bedarf beim Slave, wird dieser ebenfalls in den Updatemodus gesetzt.

Versionsdatum: 18.10.2009	Version 1.1
CAN_Doku_V101.doc	Seite 17 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Folgende Details der obigen Tabelle werden hier detailliert dargestellt:

- Power (Stufe 010)
maximaler Ausgangsstrom: 03 = 3.0 A, 04 = 4.0 A, 67 = 1.2 A → Systematik unklar.
- HW-Version (Stufe 010)
Vermutlich wird hier die HW-Version beschrieben, da diese in der MS1 als X.Y (4010) angegeben wurde. Evtl. gehören auch die zwei 00 davor dazu.
- SW-Version (Stufe 011 und Stufe 111)
Gehört vermutlich zur SW-Version; in der MS1 wird ja die Versionsnummer in der Form X.Y (0103) angegeben. Build-Nr. dürfte hier fehlen, die 4010 davor sind vermutlich die hier. Vermutlich gehören die zwei 00 vorne auch noch dazu.
- Versionsnummer BL (Vermutung: Boot Loader oder Board Layout oder Basic Library)

Folgende Felder sind bei der Übertragung vom Slave an den Master wichtig:

- UID in den ersten 4Stufen
- Stufe

Bei der Übertragung vom Master an den Slave sind folgende Daten wichtig:

- MID wird vom Master vergeben und ist bis zur Vergabe der Knotennummer das Identifizierungskennzeichen der aktuellen Anmeldung für dieses sich gerade anmeldende Gerät. Durch die MID kann man bis zu 31 gleichzeitige Busanmeldungen gleichzeitig und parallel laufen lassen.
- UID in den ersten 4 Stufen
Die UID-Bytes werden wieder zu den Slaves zurückgesandt. Wenn sich mehrere Slaves gleichzeitig anmelden wollen, kommt es zwangsläufig bei den 4 UID-Bytes spätestens beim letzten UID-Byte zu verschiedenen CAN-ID-Inhalten. Das führt auf der Hardware-Ebene zu einer Priorisierung, Ein Hardware-Sender muss seinen Sendevorgang abbrechen, und derjenige mit dem niedrigsten Byte-Wert hat dann überlebt. Seine CAN-ID wurde korrekt übertragen. Für diesen Fall ist es notwendig, dass keine Nutzdaten gesendet werden. Sonst würde es bei unterschiedlichen Nutzdateninhalten zu Fehler-CAN-Frames kommen. Da die Nutzdaten (UID-Bytes) aber in der ID drin stecken, kommt es zu keinem Busfehler, weil die CAN-Hardware das richtig priorisiert. Die Anwendungs-Software kann das aber nicht ohne weiteres detektieren.
Damit die Software so etwas erkennen kann, wird die empfangene CAN-Botschaft wieder mit der gleichen Stufe als Quittung zurückgesendet. Wegen der Hardware-Priorisierung bei der vorherigen Übertragung wird das UID-Byte mit dem niedrigeren Inhalt zurückgesandt.
Die Software des sich anmeldenden Slaves kann daran erkennen, ob sie noch weitermachen darf, oder ob ab jetzt ein anderes Gerät angesprochen wird und ihr eigener Anmeldevorgang abzubrechen ist.
Der Slave, der nicht durchkam, beendet dann sofort seine Anmeldung und wiederholt die Anmeldung nach einer Wartezeit noch einmal von vorne.
In einem Fall wurde eine Wartezeit von 5 Sekunden beobachtet. Theoretisch könnte man sogar sofort wieder mit Stufe 1 mit der Anmeldung neu beginnen.
- Start-Objekt-Handle (16 Bit Wert)
Dieser Handle wird unter anderem zur Anfrage weiterer Handle benutzt.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 18 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Knotennummer (nur die unteren 7 Bits, 8tes Bit wird nur bei CS1 / ECoS gesetzt)
Diese Knotennummer wird vom Master dem Slave für die nachfolgende Punkt-zu-Punkt-Kommunikation mit dem Master zugewiesen. Der Slave benützt bei allen nachfolgenden Frames diese ihm zugewiesene Knotennummer in der ID zu seiner Identifikation.

Aus den uns bisher vorliegenden Vergleichsmessungen haben wir folgende Knotennummern beobachtet:

Eine MS1-Master vergibt folgende Knotennummern:

Master	Slave
0	1

Knotennummern einer CS1 ohne Sniffer (SW-Version kleiner V2.0.4):

	Master	Slave
1. MS	2	3
2. MS	4	5
3. MS	6	7
usw.		

Knotennummern einer CS1 ab SW-Version V2.0.4 (mit Sniffer) sowie ECoS:

	Master	Slave
1. MS	42 (0x2a)	43
2. MS	44	45
3. MS	46	47
usw.		

Bei der CS1 (ab SW-Version V2.0.4) und der ECoS werden die MS1-Knotennummern erst ab 42 vergeben. Nach der Anmeldung aller MS1 an diese Zentralen wird die Anmeldung der ECoSlink-Geräte durchgeführt, z.B. der eingebaute Sniffer. Bei der ganzen ECoSlink-Kommunikation, werden ausschließlich „Knotennummern“ kleiner 42 verwendet. Daher kann davon ausgegangen werden, dass ESU das ECoSlink-Protokoll im Bereich der „Knotennummern“ 1 - 41 implementiert hat. Aus den CAN-Logfiles geht aber hervor, dass es sich bei „ECoSlink-Knotennummern“ nicht um Knotennummern im Sinne dieses Dokuments handelt, sondern um etwas ECoSlink spezifisches.

Aufgrund der Aufteilung der Knotennummern zwischen M-CAN-Protokoll und ECoSlink ist dafür gesorgt, dass diese beiden Protokolle gleichzeitig, auf einem Bus interferenzfrei benutzbar sind. Es findet lediglich eine Limitierung der Anzahl der MS1 statt, die von einer Zentrale bedient werden können. In der CS1 mit der SW-Version kleiner V2.0.4 könnten theoretisch $124 / 2 = 62$ M-CAN-Geräte eingesteckt werden, an der CS1 (ab V2.0.4) / ECoS sind damit nur noch $(124 - 42) / 2 = 41$ M-CAN-Geräte möglich.

6.3 Aufrüst- / Betriebsphase

In der Aufrüst- und Betriebsphase ändert sich die ID-Struktur des CAN-Frames. Während der Aufrüstung werden bestimmte Adressen / Referenzen vom Master an den Slave übergeben, so dass der Lokstack und der Systemstatus eindeutig zwischen Master und Slave kommuniziert werden können. Das Ende der Aufrüstphase wird mit einem speziellen CAN-Frame angezeigt.

In der Betriebsphase kann der Slave die ihm zugewiesenen Loks steuern. Der Slave tauscht zyklisch Lebenszeichen mit dem Master aus.

6.3.1 Frame-ID-Struktur nach Anmeldung

Nach der Übertragung der Knotennummer ändert sich die ID-Struktur der nachfolgenden CAN-Frames. Auf logischer Ebene wird jetzt eine Punkt-zu-Punkt-Verbindung zwischen der Kontennummer des Slave und der Slavenummer - 1 erstellt. Dadurch ist es möglich nach der Anmeldung dynamische Filter im CAN-Controller zu setzen und alle andere Kommunikation auszublenden. Das bedeutet aber auch, dass ein „logischer Broadcast“ (z.B. Stop oder Go) vom Master an jeden Slave einzeln zu senden ist.

Die Struktur der ID nach der Anmeldung wird durch folgende Tabelle beschrieben:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		

Die Bit-Struktur der ID wurde durch analytischen Vergleich vieler Messungen gewonnen. Die Spaltenüberschriften sind daher Bezeichner, die u.U. nicht exakt den Inhalt beschreiben.

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Spaltenüberschrift	Bit Bereich	Beschreibung																		
Priorität	28 .. 26	<p>Die bisher gemessenen Frames lassen sich zu folgenden Gruppen zusammenfassen (die Auflistung beginnt bei der niedrigsten Priorität)</p> <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>Bits</th> <th>Bedeutung</th> </tr> </thead> <tbody> <tr> <td>0b111</td> <td>Niedrigste Priorität: Anmeldung für die Vergabe einer Knotennummer und Flashen</td> </tr> <tr> <td>0b110</td> <td>Daten von der Zentrale abrufen</td> </tr> <tr> <td>0b101</td> <td>Fahrreglerstatus (Lok fahren erlaubt?) setzen / holen</td> </tr> <tr> <td>0b100</td> <td>Stop- / Go-Status setzen / holen</td> </tr> <tr> <td>0b011</td> <td>Master fährt Lok (das wurde beobachtet, wenn der Slave die Lok auch in der Anzeige hatte) / zyklische Überwachung</td> </tr> <tr> <td>0b010</td> <td>Slave fährt Lok</td> </tr> <tr> <td>0b001</td> <td>Noch nicht beobachtet</td> </tr> <tr> <td>0b000</td> <td>Höchste Priorität (nutzen auch einige ECoS-link Telegramme)</td> </tr> </tbody> </table> <p style="margin-left: 20px;">↓ Priorität</p>	Bits	Bedeutung	0b111	Niedrigste Priorität: Anmeldung für die Vergabe einer Knotennummer und Flashen	0b110	Daten von der Zentrale abrufen	0b101	Fahrreglerstatus (Lok fahren erlaubt?) setzen / holen	0b100	Stop- / Go-Status setzen / holen	0b011	Master fährt Lok (das wurde beobachtet, wenn der Slave die Lok auch in der Anzeige hatte) / zyklische Überwachung	0b010	Slave fährt Lok	0b001	Noch nicht beobachtet	0b000	Höchste Priorität (nutzen auch einige ECoS-link Telegramme)
Bits	Bedeutung																			
0b111	Niedrigste Priorität: Anmeldung für die Vergabe einer Knotennummer und Flashen																			
0b110	Daten von der Zentrale abrufen																			
0b101	Fahrreglerstatus (Lok fahren erlaubt?) setzen / holen																			
0b100	Stop- / Go-Status setzen / holen																			
0b011	Master fährt Lok (das wurde beobachtet, wenn der Slave die Lok auch in der Anzeige hatte) / zyklische Überwachung																			
0b010	Slave fährt Lok																			
0b001	Noch nicht beobachtet																			
0b000	Höchste Priorität (nutzen auch einige ECoS-link Telegramme)																			
Objekt-Handle ²	25 .. 10	<p>Diese Nummer wird vom Master für angemeldete Objekte dynamisch vergeben. Die Objekt-Handles sind variabel und werden vom Master auf verschiedene Anfragen hin an den Slave vergeben.</p> <p>Der Objekt-Handle 0 wurde bis jetzt nur beobachtet, wenn sich noch kein Slave am Master angemeldet hat. Alle anderen Objektnummern werden vom Master frei zugewiesen. Sie sind nicht fest vorgegeben und gelten nur während einer angemeldeten Verbindung.</p>																		
Befehl	9 .. 7	<p>Die Bedeutung dieser Bits ist bisher unklar. Es wird vermutet, dass es eine Art Befehlsidentifikation ist, was zu dem Bezeichner für dieses Feld führte.</p> <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>Befehl</th> <th>Verwendet bei</th> </tr> </thead> <tbody> <tr> <td>0b000</td> <td>Irgend ein Grundstatus (bei der Lok ist es, ob die Lok gefahren werden darf)</td> </tr> <tr> <td>0b001</td> <td>Nur für Namensabfragen, also Strings (bei MS1-Name und Lokname verwendet)</td> </tr> <tr> <td>0b010</td> <td>Nur für Listenabfragen wie z. B. die Masterstatusliste und die Lokfunktionsliste</td> </tr> <tr> <td>0b011</td> <td>Objektstrukturabfragen? (Da wir uns nicht ganz sicher)</td> </tr> </tbody> </table>	Befehl	Verwendet bei	0b000	Irgend ein Grundstatus (bei der Lok ist es, ob die Lok gefahren werden darf)	0b001	Nur für Namensabfragen, also Strings (bei MS1-Name und Lokname verwendet)	0b010	Nur für Listenabfragen wie z. B. die Masterstatusliste und die Lokfunktionsliste	0b011	Objektstrukturabfragen? (Da wir uns nicht ganz sicher)								
Befehl	Verwendet bei																			
0b000	Irgend ein Grundstatus (bei der Lok ist es, ob die Lok gefahren werden darf)																			
0b001	Nur für Namensabfragen, also Strings (bei MS1-Name und Lokname verwendet)																			
0b010	Nur für Listenabfragen wie z. B. die Masterstatusliste und die Lokfunktionsliste																			
0b011	Objektstrukturabfragen? (Da wir uns nicht ganz sicher)																			

² Ursprünglich wurde der Name Objekt-ID verwendet. Da es jedoch zu Verwechslungen mit der CAN-ID kam, wurde hier auf den Begriff Objekt-Handle geschwenkt und alle Werte, die in dieser Spalte benutzt werden sind somit Handles

		0b100	Objektstrukturabfragen? (Da sind wir uns nicht ganz sicher)
		0b101	Lok löschen (nur 1 x gesehen), kann auch sein dass damit ein Element in einer Liste gelöscht werden kann.
		0b110	Nicht verwendet! Das wird von der CS2 für das CS2-Protokoll ausgenutzt!
		0b111	Nur für zyklische Überwachung zwischen Master und Slave (da ist auch der 0x380-er Frame dabei)
Knotennummer	6 .. 0	Eindeutige Verbindungsidentifikation (bereits bei der Anmeldung beschrieben)	

6.3.2 Aufrüstungssequenz

In der folgenden Grafik wird die Sequenz der CAN-Frames während der Aufrüstung eines Slaves dargestellt. Hierbei sind nur die aufrüstungsrelevanten Frames aufgeführt.

Bei der Aufrüstung und im Betrieb ist zu beachten, dass auf jedes einleitende Telegramm eine Antwort gesendet wird, jedoch u.U. nicht sofort als direkte Antwort, sondern erst mit Verzögerung. Als Beispiel sei der Beginn der Aufrüstungssequenz aufgeführt. Hat der Master den Slave Description Handle übertragen, fragt er die Lokstackgröße an und überträgt sofort seine Namensanfrage. Erst danach antwortet der Slave. Bei der Slave-Antwort kann auch zuerst asynchron schon eine andere Meldung eingefügt werden. Aus diesem Grund kommt man in Slave und Master nicht mit einem ganz einfachen Sende- / Empfangsmodell aus, sondern benötigt einen komplexeren Protokoll-Stack.

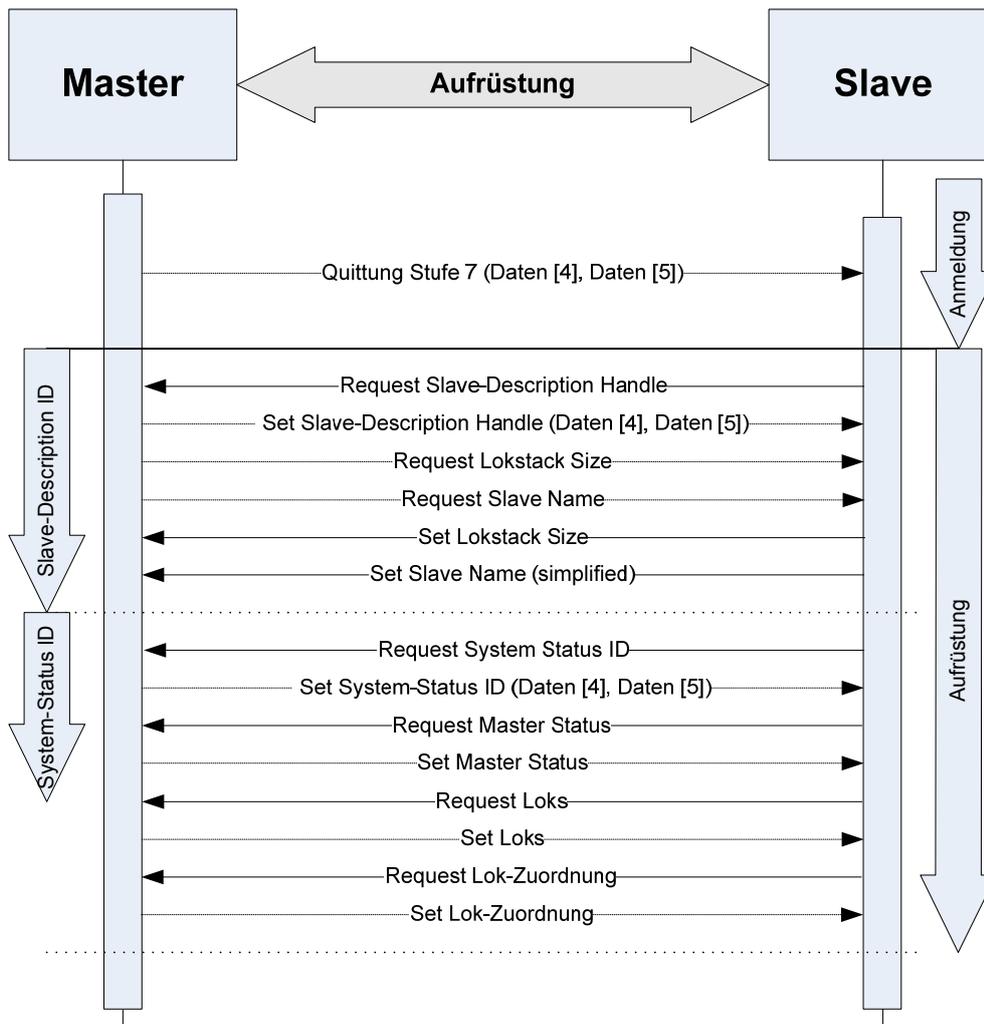


Abbildung 6-3 Sequenz einer Slave-Aufrüstung

6.4 Slave Description / Lokstack

Der Master kennt die UID des Slaves, d.h. die eindeutige Seriennummer des Geräts aus der Anmeldesequenz. Jetzt fragt der Slave den³ Handle an, unter der sich der Slave dann mit Namen vorstellt und seine Lokstackgröße bekannt gibt. Diese namentliche Vorstellung ist keine Pflicht und findet bei einer Master-MS1 auch nicht statt.

Anfrage vom Slave bzgl. Slave Description:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
					0	1	2	3	4	5	6	7
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	Aus Anmeldung	0b100	Aus Anmeldung	2	0x40	Knot						

(Knot = Knotennummer)

³ Oder „das“ konnte zwischen Nord- und Süddeutschland nicht geklärt werden, daher lieber Leser entscheiden sie selbst.

Antwortet des Masters:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	Aus Anmeldung	0b100	Aus Anmeldung	8	0x40	Knot	0x00	0x00	Obj-H	Obj-L	0x00	0x01	

Die bisher typischerweise gemessenen Werte für Obj-H und Obj-L sind:

- MS1 00 01
- CS+ECoS 00 08

Im Weiteren werden Obj-H und Obj-L als SD-Handle (Slave Description) bezeichnet und beschreiben damit diesen 16-Bit Wert.

Der SD-Handle ist bei allen Messungen bisher nur während der Aufrüstung beobachtet worden. Daher werden alle Frames, die ihn benutzen, in den folgenden Abschnitten beschrieben.

6.4.1 Request Lokstackgröße

Hier fragt der Master beim Slave die Größe des Lokstacks an. Diese Anfrage wurde bisher nur bei der CS1/CS2 und der ECoS beobachtet. Der MS1-Master nimmt an, dass der Slave nur 10 Lokstack-Plätze bietet.

Anfrage vom Master bzgl. Lokstackgröße:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SD-Handle	0b001	Aus Anmeldung	4	0x41	0x00	0x00	0x00					

Antwortet des Slaves:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SD-Handle	0b001	Aus Anmeldung	6	0x41	0x00	0x00	0x00	0x00	0x0a			

Diese Anfrage wird von einer Master-MS1 nicht gestellt.

6.4.2 Request Slave-Name

Bei diesem Telegramm wird das erste Mal die Übertragung von segmentierten Zeichenketten benutzt, die auch z.B. bei der Übertragung von Loknamen zur Anwendung kommt.

Dabei werden jeweils die Position der folgenden Zeichen im String und 4 Zeichen pro CAN-Frame übertragen. Ein Null-Byte beendet die Zeichenkette (wie bei C-Strings). Daran anschließende Null-Bytes haben keine Bedeutung.

CS1/CS2 und ECoS übertragen 4 leere Telegramme und erwarten in der Antwort den Namen des Slaves.

Anfrage vom Master bzgl. Slave-Namen:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SD-Handle	0b001	Aus Anmeldung	4	0x02	0x00	0x00	0x00					
0b110	SD-Handle	0b001	Aus Anmeldung	4	0x02	0x00	0x00	0x04					
0b110	SD-Handle	0b001	Aus Anmeldung	4	0x02	0x00	0x00	0x08					
0b110	SD-Handle	0b001	Aus Anmeldung	4	0x02	0x00	0x00	0x0C					

Antwortet des Slaves:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SD-Handle	0b001	Aus Anmeldung	8	0x02	0x00	0x00	0x00	0x4D	0x4F	0x42	0x49	
0b110	SD-Handle	0b001	Aus Anmeldung	8	0x02	0x00	0x00	0x04	0x4C	0x45	0x20	0x53	
0b110	SD-Handle	0b001	Aus Anmeldung	8	0x02	0x00	0x00	0x08	0x54	0x41	0x54	0x49	
0b110	SD-Handle	0b001	Aus Anmeldung	7	0x02	0x00	0x00	0x0C	0x4F	0x4E	0x00		

Der Name, den der Slave hier überträgt ist: "MOBILE STATION".

Hier ist sehr deutlich zu sehen, dass die Telegrammübertragung immer genau aus einer Frage und einer Antwort besteht. Der Master erwartet maximal 16 Zeichen und fragt genau diese an. Ist der String kürzer als die Anfrage, wird er Null terminiert (C-Syntax) und entsprechend übertragen. Diese Art der String-Übertragung wird auch bei Loknamen verwendet. Die Länge des letzten Antwort-Frames muss nicht entsprechend der String-Länge gekürzt sein. Das Telegramm kann auch die volle Länge (DLC=8) haben und die Daten dann mit 0x00 aufgefüllt sein.

Diese Anfrage wird von einer Master-MS1 nicht gestellt. Nur ECoS und CS1/CS2 benutzen diesen String zur namentlichen Identifizierung bei der Auswahl der am CAN-Bus angeschlossenen Geräte.

6.5 System-Status-Handle

Um den System-Status sowohl in der Aufrüstphase als auch während des Betriebs auszutauschen, wird ein spezieller Handle benötigt. Dieser wird vom Slave angefordert, vom Master beantwortet und im Weiteren dann als Objekt Handle bei der Aufrüstung und im Betrieb benutzt.

Anfrage vom Slave:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	Aus Anmeldung	0b011	Aus Anmeldung	2	0x03	0x00							

Antwortet des Masters:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	Aus Anmeldung	0b011	Aus Anmeldung	8	0x40	Knot	0x00	0x00	Obj-H	Obj-L	0x00	0x01	

Die bisher typischerweise gemessenen Werte für Obj-H und Obj-L sind:

- MS1 00 02
- CS+ECoS 00 03

Im Weiteren werden Obj-H und Obj-L als SY-Handle (System) bezeichnet und beschreiben damit diesen 16-Bit Wert.

6.6 System-Status-Austausch

Mit dem Wert des System Status Handle in der Spalte des Objekt-Handles werden die System Zustände (z.B. Stop / Go) ausgetauscht.

Folgende Status-Frames wurden gemessen:

Anfrage vom Slave

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b110	SY-Handle	0b010	Aus Anmeldung	2	0x01	0x01							

Antwortet des Masters:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b110	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x01	0x01						

Diese Statusinformationen sind in ihrer Struktur identisch. Funktional findet hier eine Zuordnung zwischen einem Index der in Slave Data[0] übertragen wird, einem Statustyp und dem zugehörigen Wert statt. Während der Aufrüstung kann der Slave immer den Statustyp und den zugehörigen Statuswert nacheinander abfragen.

Die Statusinformationen unterscheiden sich nur in den Daten und werden daher in den folgenden Tabellen zusammengefasst.

Slave Data	Bedeutung
Data[0]	fortlaufender Index beginnend bei 1
Data[1] = 0x00	Anfrage nach dem Wert
Data[1] = 0x01	Anfrage nach dem Statustyp

Master Data	Bedeutung
Data[0]	fortlaufender Index (aus Slave-Telegramm)
Data[1] = 0x00	Anfrage (aus Slave-Telegramm) nach dem Wert
Data[2]	Wert je Statustyp
Data[1] = 0x01	Anfrage nach dem Statustyp
Data[2]	Statustyp oder entfällt beim Ende der Statusliste des Masters

Um eine lose Kopplung zwischen Slave und Master bzgl. der Statusinformation zu erreichen, fragt der Slave mit aufsteigendem Index solange beim Master alle Statustypen und speichert die relevanten Statustypen ab, bis der Master auf die Statusanfrage keine Daten mehr sendet. Die Zuordnung zwischen Index und dem Statustyp muss sich auch der Master merken, da bei allen Statusänderungen im laufenden Betrieb dieser Index zum Wertaustausch benutzt wird.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 26 von 56

Statustyp	Bedeutung
0x01	Stop/Go-Status
0x02	Kurzschluss-Status
0x03	Nur bei Master MS1 (bisher unbekannt)
0x04	mfx-Anmeldestatus

Neben den Statustypen ist auch die Bedeutung der Werte relevant. Die bisher erkannten Statuswerte sind in der folgenden Tabelle zusammengefasst:

Statustyp	Wert = 1	Wert = 0
0x01	Stop	Go
0x02	Kurzschluss	Schienenspannung
0x04	Normal Betrieb	mfx-Anmeldung laufend

Während der Aufrüstung fragt der Slave die Statuswerte ab. Fragt der Slave keinen Wert ab (z.B. beim Statustyp 0x03), sendet der Master den Statuswert unaufgefordert.

6.7 Lokstack-Austausch

Beim Lokstack-Austausch werden alle Loks, die dem Lokstack des Slaves zugewiesen sind, vom Master an den Slave übertragen. Der Lokstack-Austausch besteht aus mehreren Telegrammen mit unterschiedlichen Objekt-Handles. Die Telegrammfolge bei leerem Lokstack wird am Ende dieses Kapitels behandelt.

Der Lokstack-Austausch beginnt mit der Anfrage des Slaves, ob ihm Loks zugeordnet sind.

Anfrage des Slave

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SY-Handle	0b011	Aus Anmeldung	4	0x80	0x02	0x00	0x00					

Antwort des Masters:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SY-Handle	0b011	Aus Anmeldung	8	0x80	0x02	0x00	0x00	Idx-H	Idx-L	0x00	0x01	

Bei der ersten Anfrage besetzt der Slave die Datenbytes 2 und 3 mit 0x00 vor. Bei allen folgenden Anfragen wird in diesen Bytes Idx-H / Idx-L der vorherigen Antwort eingetragen. In der folgenden Tabelle sind die Anfragen vom Slave und die Antworten vom Master gemischt. Dargestellt ist auch das Ende, d.h. alle diesem Slave zugeordneten Loks sind übertragen.

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

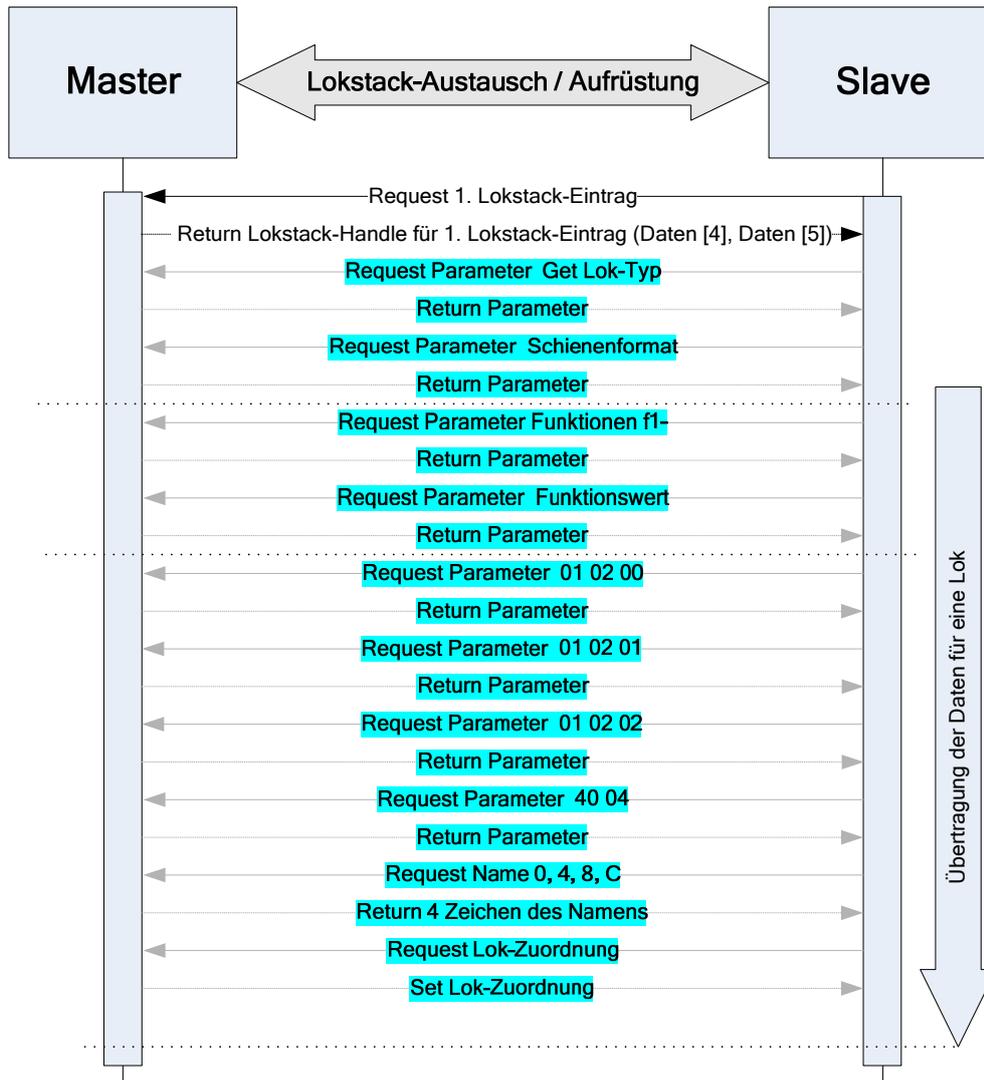
Quelle ⁴	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	SY-Handle	0b011	Aus Anmeldung	4	0x80	0x02	0x00	0x00				
M	0b110	SY-Handle	0b011	Aus Anmeldung	8	0x80	0x02	0x00	0x00	IdxH1	IdxL1	0x00	0x01
S	0b110	SY-Handle	0b011	Aus Anmeldung	4	0x80	0x02	IdxH1	IdxL1				
M	0b110	SY-Handle	0b011	Aus Anmeldung	8	0x80	0x02	IdxH1	IdxL1	IdxH2	IdxL2	0x00	0x01
S	0b110	SY-Handle	0b011	Aus Anmeldung	4	0x80	0x02	IdxH2	IdxL2				
M	0b110	SY-Handle	0b011	Aus Anmeldung	4	0x80	0x02	IdxH2	IdxL2				

Dieses Beispiel zeigt die Übertragung von 2 Loks mit den Indices (Idx-1, Idx-2). Mit dem letzten Telegramm beschreibt der Master das Ende der Übertragung.

In der folgenden Grafik wird die Telegrammkommunikation für eine Lok dargestellt. Betrachtet man die Messungen, dann ist logisch betrachtet ein Thread zuständig für die Übertragung der Lok-Indices zuständig. Für jeden belegten Index (pro Lok des Lokstacks) wird ebenfalls ein eigener Thread gestartet. (Die Implementierungsbeschreibung spiegelt am deutlichsten den Telegrammverlauf wieder. Es ist natürlich nur eine Vermutung, dass in dem Slave ein thread-orientiertes Multitasking benutzt wird.)

In der folgenden Grafik werden alle Telegramme, die zu einer Lok übertragen werden in ihrer zeitlichen Sequenz dargestellt. Die Grafik ist in Bezug auf die Anzahl der übertragenen Telegramme vereinfacht. Hier wird nur ein Telegramm dargestellt, obwohl es z.B. für die Lok-Funktionen für jede festgelegte Funktion ein Telegramm gibt. Alle Lokstack-spezifischen Telegramme benutzen als Objekt-Handle die Werte aus Idx-H / Idx-L zur Identifikation. Dieser Objekt-Handle wird im Folgenden als Lok-Handle bezeichnet.

⁴ Quelle beschreibt wer das aktuelle Telegramm sendet: S = Slave, M = Master



Wert aus Lokstack-Handle an der Position Objekt-Handle

Abbildung 6-4 Sequenz der Datenübertragung für eine Lok

6.7.1 Get Lok-Typ

Mit diesem Telegramm wird der angezeigte Lok-Typ (Dampf, Diesel, Ellok, kein Typ) übertragen.

Anfrage des Slave

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
					0	1	2	3	4	5	6	7
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	LOK-Handle	0b001	Aus Anmeldung	4	0x40	0x03	0x00	0x00				

Antwort des Masters :

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
					0	1	2	3	4	5	6	7
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	SY-Handle	0b001	Aus Anmeldung	6	0x40	0x03	0x00	0x00	0x00	Typ		

Mit dem Wert in Data[5] wird der Typ vom Master übergeben. Ob Data[4] ebenfalls zum Typ gehört ist anhand der Messungen nicht klar, jedoch drängt sich diese Vermutung auf. Die Werte in Data[5] entsprechen folgenden Typen:

Wert in Data[5]	Bedeutung
0x00	Nichts
0x01	Ellok
0x02	Diesellok
0x03	Dampflok

6.7.2 Request Schienenformat

Die Bedeutung dieses Telegramms ist erst mit der Version 3.x der ECoS Firmware klar geworden. Mit diesem Telegramm wird das Schienenformat dieses Lokstack-Eintrags bekanntgegeben. Bisher wurde nur beobachtet, dass diese Information für die reine Darstellung auf dem MS1-Display benutzt wird.

Folgende Telegrammsequenzen werden zwischen Master und Slave ausgetauscht:

Master MS1:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x40	0x01	0x00	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x40	0x01	0x00	0x00	0x00	0x03	0x00	0x03
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x40	0x01	0x01	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x40	0x01	0x01	0x00	0x00	0x4E	0x00	0x00

Master CS1 und ECoS:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x40	0x01	0x00	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x40	0x01	0x00	0x00	0x00	0x03	0x00	0x00
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x40	0x01	0x01	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x40	0x01	0x01	0x00	0x00	0x00	0x00	0x00

Ob Data[4] ebenfalls zum Typ gehört ist anhand der Messungen nicht klar, jedoch drängt sich diese Vermutung auf. Folgende Werte wurden im Feld Data[5] der ersten Anfrage beobachtet:

Wert in Data[5]	Bedeutung
0x01	Mfx
0x02	DCC / LGB
0x03	Motorola / Selectrix

Etwas verwirrend ist die Tatsache, dass der Slave zweimal in direkter Sequenz diese Anfrage fast identisch stellt. Es hat den Anschein, als wenn mit dem ersten Telegramm das Schienenformat und mit dem zweiten Telegramm seine Variante oder Detailinformation zum Schienenprotokoll abgefragt werden. Bisher wurden keine Unterschiede im zweiten Telegramm beobachtet.

6.7.3 Funktionen f1- / Funktionswerte

Diese Telegramme übermitteln die Zuordnung zwischen der Taste auf der Zentrale, dem zugeordneten Icon und einer internen Funktionsnummer. Funktionen sind hier nicht nur die Funktionsausgänge f0/fl, f1...fx, sondern auch der Motorausgang!

Bei diesen Telegrammen wird derselbe Mechanismus verwendet, der schon oben im Systemstatus beschrieben wurde. Die Übertragung erfolgt in einer offenen Liste, deren Abfrage durch den Slave initiiert und vom Master terminiert wird. Pro Funktion werden zwei Abfragen durchgeführt: Die erste beschreibt die Funktion, während mit der zweiten der Wert übertragen wird.

Es können einer Funktionstaste ein Icon und die Betätigungsart (dauerhaft / momentan) zugeordnet werden. Es ist dabei möglich Icons auszuwählen, die auf dem Display einer MS1 nicht vorgesehen sind. Diese Funktionen werden mit dem Icon-Index 0x0b übertragen. Die Icon-Informationen sowie der Zustand (Ein / Aus) werden für jede angezeigte Funktion übertragen.

Bei FKT 0x01, d.h. bei der Geschwindigkeitseinstellung wird das Telegramm zwar identisch gefüllt, jedoch die Bedeutung des Feldes ICON scheint irrelevant. Das Telegramm mit FKT 0x01 wird auch bei Funktionsdekodern identisch übertragen (sowohl in der Aufrüstphase als auch im Betrieb).

Anfrage vom Slave

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b110	LOK-Handle	0b010	Aus Anmeldung	2	FKT	0x01							

FKT ist die fortlaufende Funktionsnummer beginnen bei 0x01 für Fahrstufe und Fahrtrichtung. FKT 0x02 ist die im Schienensignal benannte Funktion „f0“ (meistens Lokbeleuchtung). FKT kann die Werte 0x01 bis 0x15 annehmen.

Antwort des Masters für belegte Funktionen:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b110	LOK-Handle	0b010	Aus Anmeldung	4	FKT	0x01	BET	ICON					

FKT ist die Funktionsnummer aus der Anfrage von oben

BET ist die Betätigungsart mit folgenden Werten:

- 00 bei FKT 0x01 (Geschwindigkeit)
- 01 Dauerfunktion
- 02 Momentanbetrieb

ICON ist der Icon-Index entsprechend der unten aufgeführten Tabelle bei Funktionen. Um kein Icon auf dem Display des Slave zu aktivieren wird der Wert 0x00 übertragen.

Beim Geschwindigkeitstelegramm wird im Feld Data[3] die Fahrstufe übertragen.

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Antwortet des Masters bei Listenende:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	LOK-Handle	0b010	Aus Anmeldung	2	FKT	0x01							

Folgende Zuordnung besteht zwischen ICON und den dargestellten Symbolen⁵:

ICON	Symbol	ICON	Symbol	ICON	Symbol	ICON	Symbol
0x01		0x05		0x09		0x0d	
0x02		0x06		0x0a		0x0e	
0x03		0x07		0x0b		0x0f	
0x04		0x08		0x0c			

Erfolgt auf die Anfrage nach einer Funktionsnummer nicht die Listenende-Antwort, fragt der Slave den Wert der aktuellen Funktion mit dem folgenden Telegramm ab. Eine Slave-MS1 ist nur in der Lage max. 8 Funktionen pro Lok zu verwalten. Sind mehr Funktionen einer Adresse (Lok) zugeordnet, erfolgt die Wertabfrage nur für die Funktionen, mit dem FKT-Wert 0x01 bis 0x09.

Anfrage vom Slave:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	LOK-Handle	0b010	Aus Anmeldung	2	FKT	0x00							

Antwortet des Masters für belegte Funktionen:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	LOK-Handle	0b010	Aus Anmeldung	3	FKT	0x00	WERT						

Die Bedeutung von WERT hängt mit der Betätigungsart der Funktion zusammen. Es gilt folgende Zuordnung:

Betätigungsart	WERT	Bedeutung
Momentan	0	Taste nicht gedrückt, Funktion nicht „aktiv“
	1	Taste gedrückt, Funktion „aktiv“
Dauer	0	Funktion aus
	1	Funktion ein
Geschwindigkeit	0x00.. 0x7f	Bei FKT 0x01 Fahrrichtung vorwärts
	0x80.. 0xff	Bei FKT 0x01 Fahrrichtung rückwärts

⁵ Die ECoS kennt die LCD-Symbole, die im Display der MS implementiert sind und ordnet ggf. ein passendes Symbol zu, so dass nicht alle ICON-Nummern übertragen werden.

Das Antworttelegramm wird im Betrieb auch spontan vom Slave als Wertänderung an den Master gesendet.

Die komplette Abfrage der letzten belegten Funktion inkl. Listenende wird in der folgenden Tabelle beschrieben:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	LOK-Handle	0x10	Aus Anmeldung	4	0x01	0x02	0x00	0x00				
M	0b110	LOK-Handle	0x10	Aus Anmeldung	5	0x01	0x02	0x00	0x00	0x00			
S	0b110	LOK-Handle	0x10	Aus Anmeldung	4	0x01	0x02	0x01	0x00				
M	0b110	LOK-Handle	0x10	Aus Anmeldung	5	0x01	0x02	0x01	0x00	0x00			
S	0b110	LOK-Handle	0x10	Aus Anmeldung	4	0x01	0x02	0x02	0x00				
M	0b110	LOK-Handle	0x10	Aus Anmeldung	5	0x01	0x02	0x02	0x00	0x00			

6.7.4 Übertragung des Loknamens

Bei der Übertragung des Loknamens wird der gleiche Mechanismus verwendet, wie bei dem „Request des Slave-Namens“. Hier geht die Initiative vom Slave aus.

Anfrage vom Slave bzgl. des Loknamens und Antwort des Masters:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x02	0x00	0x00	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x02	0x02	0x00	0x00	LN[0]	LN[1]	LN[2]	LN[3]
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x02	0x00	0x00	0x04				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x02	0x00	0x00	0x04	LN[4]	LN[5]	LN[6]	LN[7]
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x02	0x00	0x00	0x08				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x02	0x00	0x00	0x08	LN[8]	LN[9]	LN[10]	LN[11]
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x02	0x00	0x00	0x0C				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	8	0x02	0x00	0x00	0x0C	LN[12]	LN[13]	LN[14]	LN[15]

Pro Anfrage werden max. 4 Zeichen des Loknamens übertragen (LN[x]). Der Slave beendet die Anfrage entweder nach 16 Zeichen (bzw. 4 Anfragetelegrammen) oder wenn in einem Telegramm vier mal der Wert 0x00 übertragen wird. Wenn also z.B. ein Lokname „312“ übertragen werden soll, dann werden vom Master für die Felder LN[0] bis LN[3] folgende Hex-Werte übertragen: 0x33, 0x31, 0x32, 0x00. Der Slave fragt danach nach weiteren Teilen des Loknamens und erhält danach in den Feldern LN[4] bis LN[7] vier mal den Wert 0x00. Danach bricht der Slave die Anfrage nach weiteren Namensteilen ab.

6.7.5 Request / Set Lok-Zuordnung

Nach dem Restart einer MS1 wählt diese den ersten Eintrag im Lokstack zum Steuern aus, sofern ein Eintrag im Lokstack vorhanden ist. Eine Speicherung des vorherigen „Spielstandes“ findet nicht statt. Der Slave versucht den ersten Lokstack-Eintrag zu belegen und der Master antwortet entsprechend. Der Slave fragt mit folgendem Telegramm die Lok-Zuordnung beim Master an:

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 33 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7	
0b110	LOK-Handle	0b000	Aus Anmeldung	2	0x03	0x00	0x00	0x00	0x00	0x00	0x01		

Die Antwort des Masters wird bestimmt durch die Zuordnung vor dem Restart des Slaves und der aktuellen Zuordnung von Loks zu den Fahrtreglern am Master.

Antwort des Masters, wenn Zuordnung möglich:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	LOK-Handle	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		

Antwort des Masters bei Zuordnung aktuell am Master:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	LOK-Handle	0b000	Aus Anmeldung	6	0x03	0x00	0xff	0xff	0x00	0x01		

Bisher wurden in Data[5] auch größerer Werte als 0x01 beobachtet. Welche logische Bedeutung dieser Wert hat ist momentan unklar. (siehe auch nächstes Kapitel)

6.7.6 Bisher ungeklärte Telegramme

Wie im obigen angedeutet sind bisher nicht alle Telegramme interpretiert. Die Analyse eines Telegramms setzt eine gewisse Dynamik in den Daten des Telegramms voraus, die zu funktionalen Änderungen im Slave führen. Bei den folgenden Telegrammen findet z. Zt. (ECoS Firmware 3.0.1) keine Änderung in den Daten statt! Zur Vollständigkeit der Beschreibung sind diese Daten hier aufgeführt. Sollte der Leser sachdienliche Hinweise zur Funktion dieser Daten machen können, kontaktieren Sie bitte den Autor.

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b110	LOK-Handle	0x01	Aus Anmeldung	4	0x40	0x04	0x00	0x00				
M	0b110	LOK-Handle	0x01	Aus Anmeldung	6	0x40	0x04	0x00	0x00	0x00	0x00		
S	0b110	LOK-Handle	0x00	Aus Anmeldung	2	0x03	0x00						
M	0b110	LOK-Handle	0x00	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		

Bei dem letzten Telegrammpaar wurde schon eine gewisse Dynamik in den Daten beobachtet. Der Wert in Data[5] ist identisch mit dem Wert von Data[5], der bei der Zuordnung des Reglers eingetragen! Dieser Wert ist variabel, allerdings ist der Grund für die Änderung bisher nicht analysiert!

7 Betrieb

7.1 Allgemeines

Der bisherige Text beschreibt das CAN-Protokoll zwischen einer Zentrale und einer MS1 als Slave. Aufgrund der Bedienelemente der MS1 ist ein Schalten bzw. die Anzeige von Rückmeldungen nicht möglich. Daher ist es nicht möglich zu erkennen, ob für diese Aufgaben CAN-Telegramme definiert sind, die in ihrem Aufbau in das M-CAN Schema hineinpassen würden.

Ähnlich wie mit den fehlenden Bedienelementen verhält es sich auch mit Verriegelungen. Es gibt funktionale Verriegelungen, die das Dialogsystem des Masters erbringt (z.B. Ausschluss des gleichzeitigen Steuern einer Lok von Master und Slave). Hier ist es nicht möglich zu sagen, welche Telegramme solche verriegelten Bedienungen auf dem CAN-Bus hervorbringen würden.

7.2 Zyklische Überwachung

Mit Abschluss der Aufrüstphase beginnt der Slave mit der angeschlossenen Zentrale zyklisch Lebenszeichen auszutauschen. Es besteht kein Unterschied ob innerhalb des Zyklus ein Telegramm zwischen Master und Slave ausgetauscht wird oder nicht. Obwohl bereits nach der Zuweisung der Knotennummer alle Daten zum Durchführen der zyklischen Überwachung vorhanden sind, wird anscheinend aus Optimierungsgründen auf die zyklische Überwachung während der Aufrüstphase verzichtet.

7.2.1 Slave angeschlossen

Die zyklische Überwachung findet mit dem folgenden Telegrammaustausch statt:

Anfrage vom Slave:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b011	Aus Anmeldung	0b111	Slave	4	0x00	0x00	0x00	Knot					

Knot = Knotennummer des Masters

Antwort des Masters:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
					0	1	2	3	4	5	6	7
0b011	Aus Anmeldung	0b111	Master	8	0x03	0x00	0x00	Knot	UID[3]	UID[2]	UID[1]	UID[0]

Knot = Knotennummer des Masters

Data[4...7] sind bei der normalen Anfrage mit der UID[3...0] des Slaves aus der Anmeldung gefüllt. Kommt es zu einer Verzögerung im Pollzyklus (wurde beim Anmelden einer mfx-Lok im Betrieb beobachtet), wird erst mehrfach ohne Antwort des Masters gepollt und dann die Anfrage „Verzögerung im Zyklus“ gestellt.

Die normale Anfrage wird zyklisch ca. alle 250ms vom Slave an den Master gesendet.

7.2.2 Slave entfernt / Verzögerung im Zyklus

Bei einer ECoS mit Firmware 3.0.1 wurde beim Ausbleiben der Slave Antwort für ca. 7,5sec folgendes Telegramm gemessen:

Anfrage vom Master:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b011	Aus Anmeldung	0b111	Master	4	0x00	0x00	0x00	Knot					

Knot = Knotennummer des Masters

Antwort des Slaves / neuer Zyklus:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b011	Aus Anmeldung	0b111	Slave	8	0x03	0x00	0x00	0x00	UID[3]	UID[2]	UID[1]	UID[0]	

Data[4...7] sind bei dieser Anfrage mit der UID[3...0] des Slaves aus der Anmeldung gefüllt. Der Wiederholungszyklus für dieses Telegramm beträgt ca. 1sec.

Dieses Telegramm wird max.30 mal wiederholt. Danach wird die virtuelle Verbindung zwischen einem Master und dem gerade angefragten Slave vom Master terminiert und die zyklische Anfrage beendet. Es ist davon auszugehen, dass alle dynamischen Belegungen im Master für diese Verbindung zurückgesetzt werden.

7.3 Steuern

Nur Adressen (Loks, ...), die im Lokstack des Slaves eingetragen sind, können gesteuert werden, d.h. es können die Geschwindigkeit, die Fahrtrichtung und die Funktionen (F0, ...) geändert werden. Steuerungsmeldungen werden nur bei Änderungen durch den Bediener übertragen. Befindet sich eine Lok im Lokstack des Slaves, wird aber aktiv vom Master gesteuert und nicht auf dem Slave angezeigt, werden keine Steuerungstelegramme an den Slave übertragen. Es gibt folgende Ausnahme: Die aktuelle Firmware der ECoS (V3.0.x) erlaubt keine gleichzeitige Bedienung derselben Lok vom Drehknopf der ECoS und vom Drehknopf einer angeschlossenen MS1 (Werkseinstellung). Jedoch kann dieselbe Lok über das LAN-Interface der ECoS gesteuert werden.

Die Telegrammstruktur und -bedeutung wurde bereits in der Aufrüstphase im Kapitel „6.7.3 Funktionen f1- / Funktionswerte“ ausführlich beschrieben.

Beispiel einer Geschwindigkeitssteuerung durch den Slave:

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 36 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b110	LOK-Handle	0b010	Aus Anmeldung	3	0x01	0x00	WERT						

Für die Übertragung von Funktionen gilt der oben beschriebene Telegrammaufbau.

Da die fehlerfreie Übertragung durch die CAN-Bus Hardware sichergestellt ist, wird auf eine Quittung durch den Master verzichtet. Es wird nur vom Slave an den Master gesendet ohne dass der Master mit einem Quittungstelegramm die Übertragung bestätigt.

7.4 Status Änderung Stopp / Go / Überstrom

Die verwendeten Protokollelemente sind bereits in der Anmeldung im Kapitel „6.6 System-Status-Austausch“ beschrieben. Sie werden hier nur zur besseren Lesbarkeit erneut aufgeführt.

Stopp-Button an Master gedrückt:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x00	0x01						

Löschen der Überstromindikatoren durch Master:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x02	0x00	0x00						

Übertragen des Spannungszustands am Boosterausgang:

CAN ID	DLC	Data [0 .. 7]							
		0	1	2	3	4	5	6	7
Xxx	7	0x00	0x04	0x00	0x01	0x03	0x00	0x00	

Da dieser CAN-Frame nicht vom MS1-Master übertragen wird, ist er hier nur zur Vollständigkeit aufgenommen. Es scheint sich um ein ECOSlink Telegramm zu handeln. In Data[6] zeigt ein 0x00 an, dass keine Spannung an der Schiene anliegt, bei dem Wert 0x01 wird die Schiene spannungsversorgt. Dieses letzte Telegramm wird auch dann auf dem CAN-Bus gesendet, wenn keine MS1 angeschlossen ist. Alle anderen Telegramme dieses Kapitels werden ohne angeschlossenen Slave / MS1 unterdrückt!

Wird der Stopp-Button an der MS1 gedrückt kommt es zu folgender Sequenz:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7	
S	0b0010	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x00	0x01						
M	0b100	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x00	0x01						
M	0b100	SY-Handle	0b010	Aus Anmeldung	3	0x02	0x00	0x00						

Danach folgt wie oben beschrieben die Übertragung des Spannungszustands am Boosterausgang.

Die Betrachtung der Kurzschluss-Übertragung ist aufgrund eines fehlenden ECoS-Boosters vereinfacht. Da der MS1-Booster nicht benutzt wird, kann die Kurzschlussmeldung nicht von einer MS1 initiiert werden. Für die folgenden Messungen stand kein ECoS-Booster zur Verfügung. Daher sind hier nur die Telegramme betrachtet, die durch einen Kurzschluss des eingebauten Boosters erzeugt werden.

Es tritt kein Telegramm-Unterschied zwischen einem Kurzschluss am Programmiergleis bzw. am „MainGleis“ auf. Kommt es zu einem Kurzschluss am Boosterausgang, wird eine ECoS mit der Firmware 3.0.1 etwas „panisch“, was sich in folgender Übertragungssequenz äußert (alle Telegramme von der ECoS gesendet):

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
Bit28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7	
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x00	0x01						
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x02	0x00	0x00						
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x01	0x00	0x01						
0b100	SY-Handle	0b010	Aus Anmeldung	3	0x02	0x00	0x01						

Danach erfolgt 3-mal wie oben beschrieben die Übertragung des Spannungszustands am Boosterausgang.

Die Analyse der Sequenz ist:

- Setze Stopp-Indikator auf MS1-Display
- Lösche Überstrom-Indikator auf MS1-Display
- Setze Stopp-Indikator auf MS1-Display
- Setze Überstrom-Indikator auf MS1-Display

Diese Sequenz wird durch keine zyklische Überwachung unterbrochen. Die MS1 sendet auch keine Quittung für irgendeines der vom Master gesendeten Telegramme.

7.5 Umtaufen eines Slaves

Mit dieser Funktion kann man den Namen eines Slaves ändern. Dieser Name wird sowohl im Master als auch im Slave angezeigt. Mit Änderung des Namens eines Slaves im Master werden Telegramme ähnlich wie im Kapitel „6.4.2 Request Slave-Name“. Der Unterschied liegt in der Art und Weise wie der Master den Slave-Namen anfragt.

Setzen des Slave-Namens durch den Master:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]								
					0	1	2	3	4	5	6	7	
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0										
0b110	SD-Handle	0b001	Aus Anmeldung	8	0x02	0x00	0x00	0x00	0x4B	0x53	0x20	0x4D	
0b110	SD-Handle	0b001	Aus Anmeldung	7	0x02	0x00	0x00	0x04	0x53	0x31	0x00		

Hier wurde der Name einer MS1 auf „KS MS1“ umgeändert. Der Slave antwortet zweimal mit derselben Sequenz, nur unter Benutzung seiner Knotennummer.

7.6 Zuordnung /Abmelden einer Lok

Wie bereits im Kapitel „7.1 Allgemeines“ beschrieben, kann eine Lok nur vom Master oder alternativ von einem Slave gesteuert werden. Eine gleichzeitige Bedienung über den Fahrtregler am Master und an dem MS1-Slave ist ausgeschlossen (Werkseinstellung). Betrachtungen zur Übergabe von Loks die über den Sniffer-Eingang gesteuert werden, sind kein Bestandteil dieser Phase der CAN-Bus Analyse.

Um eine Lok zu steuern, muss der Slave diese Lok beim Master (sofern frei) belegen. Dieses Belegen erfolgt mit folgender Telegramm-Sequenz:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
						0	1	2	3	4	5	6	7
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0									
S	0b101	LOK-Handle	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		

Lok-Handle ist dabei die Referenz aus der Übergabe des Lokstacks.

Für den nächsten Fall wurde parallel zum Slave auf dem Master dieselbe Lok einem Fahrtregler zugewiesen. Dabei werden keine CAN-Telegramme zwischen Master und Slave ausgetauscht – logisch, da der Master die komplette Lokverwaltung durchführt.

Wird jetzt auf dem Slave ein Lokauswahl-Dialog begonnen, dann wird sofort mit Anzeige einer anderen Lok folgender Telegramm-Dialog durchgeführt:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
						0	1	2	3	4	5	6	7
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0									
S	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x01	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0xFF	0xFF	0x00	0x01		
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		

Wie bereits im Kapitel „6.7.5 Request / Set Lok-Zuordnung“ kann Data[5] größere Werte als 0x01 annehmen. Tritt dieser Fall auf, dann antwortet der Master in einer Situation auch mit dem gleichen Wert, mit Ausnahme der Zuordnungsfreigabe (letzte Zeile in vorheriger Tabelle). Diese wurde immer mit Data[5] = 0x01 beobachtet. Die folgende Zuordnung der Lok wird dann aber wieder mit dem größeren Wert durchgeführt.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 39 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b101	LOK-Handle2	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle2	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		

Da der Master diese Lok (LOK-Handle1) sofort belegt, gibt es für den Slave keine Möglichkeit die Lok weiter zu steuern, bis der Master LOK-Handle1 freigibt. In der Zwischenzeit zwischen der Abmeldung von LOK-Handle1 und dem Anmelden der zweiten Lok findet normaler Telegrammverkehr statt, d.h. minimal Poll-Telegramme. Als nächstes wurde versucht den LOK-Handle1 wieder vom Slave zu steuern. Da in der Zwischenzeit LOK-Handle2 bereits selektiert war, ergibt sich folgende Sequenz:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b101	LOK-Handle2	0b000	Aus Anmeldung	6	0x03	0x01	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle2	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		
M	0b101	LOK-Handle2	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		

Mit dieser Sequenz wird eine Lok (LOK-Handle2) durch den Slave wieder freigegeben.

Mit Auswahl von LOK-Handle1, die noch vom Master gesteuert wird, wird folgende Telegramm-Sequenz ausgetauscht:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
S	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x01	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0xFF	0xFF	0x00	0x01		

Auf der MS1 blinkt jetzt ein durchgekreuztes MS1-Icon unterhalb des Geschwindigkeitsbalkens als Indikator, dass die aktuelle Lok nicht durch den Slave steuerbar ist. Geschwindigkeitsänderungen am Master werden auf dem Slave angezeigt.

Als nächstes wurde LOK-Handle1 auf dem Master deselektiert, d.h. auf beiden Fahrtreglern der ECoS waren andere Loks ausgewählt. Daraus ergibt sich folgende Telegrammsequenz, da der Slave diese Lok noch angewählt hatte:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x00	0x00	0x01		
S	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x01	0x00	0x08	0x00	0x01		
M	0b101	LOK-Handle1	0b000	Aus Anmeldung	6	0x03	0x00	0x00	0x08	0x00	0x01		

Wird auf dem Master eine Lok aus dem Slave-Lokstack ausgewählt, die nicht vom Slave selektiert ist, erfolgt folgende Telegrammsequenz:

Quelle	Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
	Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
M	0b101	LOK-HandleX	0b000	Aus Anmeldung	6	0x03	0x00	0xFF	0xFF	0x00	0x01		

Wird nun diese Lok (LOK-HandleX) zwischen dem rechten und linken Fahrtregler am Master ausgetauscht, werden zuerst die Abmeldung und dann die Neuankündigung gesendet. Anhand der übertragenen Daten ist nicht erkennbar, ob LOK-HandleX vom linken oder rechten Fahrtregler gesteuert wird.

Aus den beschriebenen Telegrammsequenzen lässt sich die Funktion der Datenelemente nur erraten:

Data[1] könnte die Reglernummer am Slave sein (hier MS1 mit einem Regler). Bei Data[1] = 0 ist kein Regler selektiert.

Data[2] und Data[3] beschreiben folgende Zustände:

- 0x0008 – Anmeldung am Slave
- 0x0000 – Abmelden am Master oder am Slave (je nach Absender-Knotennummer)
- 0xffff – Lok-Handle durch Master belegt.

Weitere Werte von Data[2]/[3] wurden bisher nicht beobachtet.

7.7 Hinzufügen / Löschen einer Adresse aus dem Lokstack

Unabhängig davon, ob eine Lok von einem Slave gesteuert wird oder nicht, kann sie auf dem Master aus dem Lokstack herausgelöscht werden. Genauso ist es möglich, unabhängig von der Steuerung durch einen Master eine Lok einem Slave-Lokstack hinzuzufügen.

7.7.1 Löschen einer Adresse aus dem Lokstack

Diese Funktion wird auf dem Master initiiert und zwar durch Betätigung des Bestätigungs-Buttons im Slave Dialog. Folgendes Telegramm wird dabei vom Master an den Slave gesendet:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
					0	1	2	3	4	5	6	7
0b110	SY-Handle	0b101	Aus Anmeldung	8	0x80	0x00	0x00	0x00	LokstackH	LokstackL	0x00	0x01

Data[4] / [5] beinhalten den Index des Lokstack-Eintrags, der gelöscht werden soll. Ein MS1-Slave sucht jetzt in seinem Lokstack nach einem weiteren Eintrag und meldet diesen mit den Telegrammen wie im Kapitel „7.6 Zuordnung /Abmelden einer Lok“ beschrieben sofort an. Steht kein weiterer Eintrag im Lokstack zur Verfügung, unterbleibt diese Anmeldung.

7.7.2 Hinzufügen einer Adresse zum Lokstack

Das Hinzufügen einer Adresse/Lok zum Lokstack erfolgt ähnlich wie in der Aufrüstung im Kapitel „6.7 Lokstack-Austausch“ beschrieben. Allerdings wird in der Aufrüstung diese Abfrage durch den Slave initiiert, jetzt initiiert der Master.

Initiierung der Lokstack-Anfrage durch den Master:

Priorität	Objekt-Handle	Befehl	Knotennummer	DLC	Data [0 .. 7]							
Bit 28 .. 26	Bit 25 .. 10	Bit 9 .. 7	Bit 6 .. 0		0	1	2	3	4	5	6	7
0b110	SY-Handle	0b100	Aus Anmeldung	8	0x80	0x00	0xFF	0xFF	Idx-H	Idx-L	0x00	0x03

Nach diesem Telegramm beginnt der Slave sofort, d.h. ohne eine weitere Quittierung mit der Abfrage der Parameter für diesen Lokstack-Eintrag wie in den 6.7.x beschrieben. Mit Übertragung des letzten Telegramms des Loknamens ist die Übertragung eines Lokstack-Eintrags abgeschlossen. Ein Aufrüstungsende Telegramm wird nicht übertragen. Während dieser Übertragung findet auch keine zyklische Überwachung statt.

8 Herunterfahren des Masters

Wird der Master ausgeschaltet, versucht der Slave noch 30mal den Master zyklisch zu überwachen und stellt danach die Überwachung ein. Parallel dazu zeigt eine MS1 im Display „KEIN MASTER“ an. Danach, d.h. nach den vergeblichen Überwachungsversuchen, versucht der Slave sich bei einem Master anzumelden und sendet zyklisch das Stufe 0 Telegramm der Anmeldung.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 43 von 56

9 Flashen einer MS1

9.1 Überblick

Die MS1-Geräte sind in der Lage, sich selbst neu zu programmieren. Dies kann erforderlich werden, wenn z. B. das Protokoll, wie in den vorherigen Kapiteln beschrieben, geändert oder erweitert werden soll. Zudem wird es erforderlich, wenn in der Lokdatenbank der MS1 neue Einträge dazu kommen. In so einem Fall hält dann die Zentrale die neue Firmware für die MS1 in einer Datei bereit. Jede MS1-Firmware hat eine Kennung, welche SW-, HW- und Herstellerkennung (Vendor-ID) sie selber hat. In den dafür vorgesehenen Firmware-Update-Dateien (BCI-Files) der Zentralen sind diese Angaben auch enthalten. Beim Anmelden am CAN-Bus, wie in den vorherigen Kapiteln beschrieben, werden diese Kennungen schon ganz am Anfang in den ersten CAN-Frames ausgetauscht. Dabei prüft die Zentrale, ob es notwendig und möglich ist, eine Aktualisierung der Firmware auf der MS1 durchzuführen. Ist dies der Fall, wird das der MS1 angeboten.

9.2 Setzen des Flashbits im Anmelde-Frame

Prio.	UID	Stufe	MID	Richt.	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 16	Bit 15 .. 13	Bit 13 .. 8	Bit 7	Bit 6 .. 0									
0b111	0	0b111	0 .. 31 von Master	Slave	126	8	0x00	0x00	0x40	0x10	0x01	0x03	0x00	0x84
0b111	0	0b111	0 .. 31 von Master	Master	126	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	Start-Obj.-Hand 0x00	Start-Obj.-Hand 0x01	Bit 0-6 Knotennummer	0x01

Wenn bei der Knotennummervergabe in der Stufe 7 beim Zuweisen der Knotennummer im 8. Datenbyte (Data[7]) das Bit Nr. 0 von der Zentrale gesetzt wird, signalisiert die Zentrale dem Slave, dass er in den Flash-Mode gehen soll. In diesem Fall wird davon ausgegangen, dass die Übergabe der Slave-Kennungen wie SW-Version, HW-Version und Vendor-ID beim Anmeldevorgang dazu geführt haben, dass die Zentrale meint, ein Update für den Slave bereit zu haben. Bei einer MS1 als Slave führt das dazu, dass der Bootloader resetfest angesprochen wird. Die MS1 merkt sich das dauerhaft und beharrt von nun an auf ein Update. Wird sie in diesem Zustand ausgesteckt und dann wieder eingesteckt, macht sie genau an diesem Punkt wieder weiter. Sie will sich flashen lassen. Dieser Zustand kann nur durch einen erfolgreichen Flashvorgang beendet werden. Auf dem Display der MS1 ist in diesem Zustand nichts zu sehen. Offensichtlich ist der Bootloader nicht in der Lage, etwas auf dem Display auszugeben.

Wenn das Flash-Indikationsbit in Data[7].0 von der Zentrale gesetzt wurde, kann das angesprochene Gerät vermutlich selbst entscheiden, ob es in den Flashmodus geht, oder nicht. Die bisher ausgelieferten MS1 machen das jedenfalls immer, in dem sie dann mit dem Flashen fortfahren, bzw. die Zentrale auffordern, den Flashprozess anzustoßen.

9.3 ID-Aufteilung beim Flashen

Priorität	Flash-Handle	Request	Flash-Stufe	Flash-Flags	Knotennummer	DLC	Data [0 .. 7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14 .. 10	Bit 9 ...7	Bit 6 .. 0 = 127		

Die Bezeichnungen Flash-Handle, Request, Flash-Stufe und Flash-Flags sind willkürlich festgelegte Bezeichnungen, die zu den von den Programmierern verwendeten Bezeichnungen keinen Bezug haben. Sie dienen hier lediglich als Beschreibungshilfsmittel. Das gleiche gilt für die hier angegebenen Bitgrenzen.

9.4 Signalisierung der Flashbereitschaft an die Zentrale

Wurde ein Gerät in den Flashmodus versetzt, wird es daraufhin die Zentrale auffordern, mit dem Flashen zu beginnen. Dies erfolgt mit dem nachfolgend beschriebenen CAN-Frame. Nennen wir ihn einmal Flashbereitschaft des flashwilligen Slaves.

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x0	0b101	0b00000	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Sollte eine MS1 nach der Busanmeldung von der Zentrale in den Flashmodus versetzt worden sein und dann ausgesteckt werden, dann muss nach dem Wiedereinstecken auf jeden Fall noch einmal geprüft werden, welche Kennungen das flashwillige Gerät hat. Ebenso muss der Zentrale die Geräte-UID bekanntgemacht werden. Alle diese Kennungen sind in so einem Fall der Zentrale nicht mehr bekannt.

Da es auch vorkommen kann, dass an einer Zentrale zur gleichen Zeit mehrere flashwilligen Geräte angesteckt sein können, muss man die einzelnen Geräte beim Flashen auch unterscheiden können. Zu diesem Zweck wird nach der Anzeige der Flashbereitschaft eines Slaves diesem eine eindeutige Kennung zugewiesen, unter der das Flashen für genau diese Geräte-UID durchgeführt wird. Dies wird hier auch in Anlehnung einer Anmeldung ähnlich wie die MID bei der Busanmeldung gehandhabt. Nach der Zusendung der Flashbereitschaft vergibt die Zentrale für diese nun folgende Flash-Session eine Kennung, das Flash-Handle. So wie es aussieht, wird dafür ein Byte in der CAN-ID vorgehalten. Es sind die Bits 25 bis 18. Dieses Flash-Handle wird bei jeder neu erkannten Flashbereitschaft eines Slaves erhöht. In dem Fall einer ECoS und einer CS1 wird das Handle immer um 2 erhöht. Nimmt man jedoch an dass nur die Bits 25 – 19 dieses Handle beinhalten, wird immer nur um eins erhöht. Begonnen wird mit dem Handle Nr. 2. Bit 19 ist in diesem Fall gesetzt. Das zu flashende Gerät übernimmt bei den nachfolgenden Flash-Frames dieses ihm zugewiesene Flash-Handle für alle nachfolgenden Flash-CAN-Frames. Da bei der Flash-Handle-Vergabe in den ersten 4 Datenbytes die Geräte-UID enthalten ist, kann davon ausgegangen werden, dass in diesem Fall auch nur genau dieses eine Gerät angesprochen wird.

9.5 ID-Abfragen

Gleichzeitig mit der Vergabe des Flash-Handles wird auch noch die 4-Byte lange Vendor-ID abgefragt. Dies erfolgt mit folgendem CAN-Frame:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b00001	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Bei den Abfragen der Vendor-ID, der SW/HW-ID und einer weiteren Kennung, die sich uns bisher nicht genau erschlossen hat (vielleicht die Build-Version), wird immer auch die Geräte-UID in den Datenbytes [0 ... 3] mit übergeben. Dadurch kann eine Verwechslung von Geräten ausgeschlossen werden.

Antwort vom Slave auf die Vendor-ID-Abfrage:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02 von Zentrale vergeben	0b000	0b00001	0b000	127	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	0x00	0x00	0x00	0x97

Das 0x97 = 151 im Datenbyte 7 kennzeichnet ein Gerät der Firma ESU entsprechend NMRA-Festlegung.

SW/HW-ID-Abfrage:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b00010	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Antwort vom Slave auf die SW/HW-ID-Abfrage:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b00010	0b000	127	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	HW-ID 0x01	HW-ID 0x00	SW-ID 0x40	SW-ID 0x10

Build-Versionsabfrage:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0		Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
0b111	0x02	0b000	0b00011	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Antwort vom Slave auf die Build-Versionsabfrage:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0		Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
0b111	0x02	0b000	0b00011	0b000	127	8	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave	HW-ID 0x01 ?	HW-ID 0x00 ?	Buildv. 0x00	Buildv. 0x67

Sind diese Abfragen gemacht, kennt die Zentrale die Parameter, anhand deren entschieden wird, ob das Gerät geflashed wird oder nicht.

9.6 Flash-Abbruch

Wir haben in einem Fall beobachtet, dass an dieser Stelle der Flashvorgang von der Zentrale abgebrochen worden ist.

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0		Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
0b111	0x00	0b000	0b00000	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Wird dieser Frame einer flashwilligen MS1 gesendet, hört sie auf, weitere CAN-Frames zu senden.

9.7 Flash-Freigabe:

Steht dem Flashvorgang nichts mehr im Weg, wird mit dem nachfolgenden Frame die Flashfreigabe an das flashwillige Gerät gegeben.

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0		Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
0b111	0x02	0b000	0b00000	0b000	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Der Flashvorgang selber wird vom flashwilligen Gerät gesteuert. Es fordert die Zentrale auf, ihm X Bytes ab Daten-Offset Y zuzusenden. Dabei werden die Flags bedient. Das Flash-Handle ist dasjenige von der ersten Vergabe bei der Vendor-ID-Abfrage. Wenn die MS1 die Daten von der Zentrale anfordert, setzt sie das Bit 8 der CAN-ID Liefert die Zentrale die Daten, dann setzt sie das Bit 9 der CAN-ID. Die Zent-

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

rale sendet pro CAN-Flash-Frame immer 4 Bytes. Ab welchem Daten-Offset die Zentrale senden soll, teilt das geflashte Gerät der Zentrale in den Datenbytes mit (Data[0 ...3]). Somit stehen also 2^{32} Bytes (2GB) als max. flashbare Firmware zur Verfügung. Das sollte auch für künftige Geräte noch eine zeitlang reichen.

Flashdaten-Abfrage vom geflashten Gerät:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b00000	0b001	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x00	Offset [0] 0x00	Anzahl [3] 0x00	Anzahl [2] 0x00	Anzahl [1] 0x02	Anzahl [0] 0x26

Hier fordert die MS1 0x226 = 550 Bytes ab Offset 0x00 an.

Antwort der Zentrale auf die Flash-Datenabfrage des Slaves mit Flashdaten:

Prio.	Handle	Req.	Stufe	Flags	Knotennummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b00000	0b010	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x00	Offset [0] 0x00	Flash-Data 0x35	Flash-Data 0x09	Flash-Data 0xF8	Flash-Data 0xA9
0b111	0x02	0b000	0b00000	0b010	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x00	Offset [0] 0x04	Flash-Data 0xD0	Flash-Data 0x53	Flash-Data 0x04	Flash-Data 0xAE
0b111	0x02	0b000	0b00000	0b010	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x00	Offset [0] 0x08	Flash-Data 0xE4	Flash-Data 0xC3	Flash-Data 0x19	Flash-Data 0x51
0b111	0x02	0b000	0b00000	0b010	127	8	wei-	tere	134	CAN	Frames	der	Zen-	Trale
0b111	0x02	0b000	0b00000	0b010	127	6	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x02	Offset [0] 0x24	Flash-Data 0x36	Flash-Data 0x74		

In unserem Fall sendet dann die Zentrale auch 0x0226 = 550 Bytes ab Offset 0 an die MS1. Offensichtlich passen 550 Bytes in den Flash-Buffer der geflashten MS1, welche dann in das Flash der MS1 übertragen (programmiert) werden. Da 550 Bytes nicht geradzahlig in 4-Byte-CAN-Frames übertragen werden können, enthält der letzte Frame nur 2 Flash-Daten-Bytes. Um die Buslast zu reduzieren, werden nicht jedesmal 4 weitere Flash-Daten-Bytes vom dem zu flashenden Gerät abgefragt, obwohl das Protokoll das auch hergeben würde. Das würde die Kommunikation nur unnötig verlängern. Vielmehr werden die 550 Bytes als aufeinanderfolgende CAN-Frames incl. dem Hochzählen des Offsets in den Datenbytes Data[0 ...3] einfach als Block übertragen. Dies ergibt dann 138 CAN-Frames am Stück, der letzte Frame mit nur 2 Nutzdatenbytes.

Wir gehen davon aus, dass in den 550 Bytes 512 Bytes Nutzdaten enthalten sind, welche dann durch Checksummen und sonstigem Overhead 550 Bytes ergeben.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 48 von 56

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Sind die 138 CAN-Frames an das zu flashende Gerät übertragen, dann fordert das geflashte Gerät, wenn es wieder empfangsbereit ist, den nächsten Block mit Flashdaten an. Das geht dann so lange, bis alle Daten vom BCI-File übertragen sind.

Ein weitere Abfrage vom Slave mit einem Byteoffset zwischendrin:

Prio.	Hand- le	Req.	Stufe	Flags	Kno- ten- num- mer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0b0x02	0b000	0b000000	0b001	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x06	Offset [0] 0x72	Anzahl [3] 0xXX	Anzahl [2] 0xXX	Anzahl [1] 0xXX	Anzahl [0] 0xXX

Flashdaten von der Zentrale mit Byteoffset zwischendrin:

Prio.	Hand- le	Req.	Stufe	Flags	Kno- ten- num- mer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b000000	0b010	127	8	Offset [3] 0x00	Offset [2] 0x00	Offset [1] 0x06	Offset [0] 0x72	Anzahl [3] 0xXX	Anzahl [2] 0xXX	Anzahl [1] 0xXX	Anzahl [0] 0xXX
0b111	0x02	0b000	0b000000	0b010	127	8	0x00	0x00	0x06	0x76	0xXX	0xXX	0xXX	0xXX
0b111	0x02	0b000	0b000000	0b010	127	6	0x00	0x00	0x08	0x92
0b111	0x02	0b000	0b000000	0b010	127	6	0x00	0x00	0x08	0x96	0xXX	0xXX		

9.8 Flashen-Ende-Frame

Sind alle Bytes übertragen, dann wird ein „Ende-Frame“ gesendet. Dieser Frame zeigt dem zu flashenden Gerät an, dass keine weitere Flashdaten mehr in der Zentrale vorhanden sind.

Prio.	Hand- le	Req.	Stufe	Flags	Kno- ten- num- mer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]
Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14.. 10	Bit 9 .. 7	Bit 6 .. 0									
0b111	0x02	0b000	0b000000	0b011	127	4	UID[3] Slave	UID[2] Slave	UID[1] Slave	UID[0] Slave				

Ist das komplette BCI-File übertragen und das Flash-Ende der MS1 bekannt gemacht worden, dann macht die MS1 von sich aus einen Reset und startet neu. Ist beim Flashen alles gut gegangen, wird sie sich wieder neu am CAN-Bus, wie in den vorherigen Kapiteln beschrieben, anmelden.

9.9 Beispiel eines Flashvorgangs

Hier noch ein Beispiel vom Beginn eines Flashvorganges (rechts außen etwas kommentiert). Anmeldung mit anschließendem Flashen

Quelle	Priorität	UID	Stufe	MID = Veränderungs- zähler	Richtung gesendet von	Knoten- nummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]	Kommentar	
	Bit 28 .. 26	Bit 25 .. 16	Bit 15 .. 13	Bit 12 .. 8	Bit 7	Bit 6 .. 0											
S	0b111	0b00000000	0b000	0b00000000	Slave	126	0									Frames von MS1	
M	0b111	0b00000000	0b000	0b1011000	Master	126	8	00	03	0xC6	0x7F	00	01	00	01		
S	0b111	0b00000010	0b001	0b1011000	Slave	126	0										
M	0b111	0b00000010	0b001	0b1011000	Master	126	8	00	00	00	0x97	00	00	01	01		
S	0b111	0b10011100	0b010	0b1011000	Slave	126	0										
M	0b111	0b10011100	0b010	0b1011000	Master	126	8	01	0x13	00	03	00	00	00	00		
S	0b111	0b00000011	0b011	0b1011000	Slave	126	0										
M	0b111	0b00000011	0b011	0b1011000	Master	126	8	00	00	00	01	01	06	00	0x2D		
S	0b111	0b00000000	0b100	0b1011000	Slave	126	8	00	02	0x9C	03	00	01	00	03		UID MS1
S	0b111	0b00000000	0b101	0b1011000	Slave	126	8	00	00	00	0x97	01	00	0x40	0x10		
S	0b111	0b00000000	0b110	0b1011000	Slave	126	8	01	00	00	0x67	00	00	0x40	10		
S	0b111	0b00000000	0b111	0b1011000	Slave	126	8	00	00	0x40	0x10	01	01	00	1E		SW-Version MS
M	0b111	0b00000000	0b111	0b1011000	Master	126	8	00	02	0x9C	03	00	02	00	01	Aufforderung zum Flashen	

Quelle	Priorität	Handle	Req.	Stufe	Flags	Knoten- nummer	DLC	Data [0]	Data [1]	Data [2]	Data [3]	Data [4]	Data [5]	Data [6]	Data [7]	Kommentar
	Bit 28 .. 26	Bit 25 .. 18	Bit 17 .. 15	Bit 14 .. 10	Bit 9 .. 7	Bit 6 .. 0										
S	0b111	0b00000000	0b101	0b000000	0b000	127	4	0x00	0x02	0x9C	0x03					Flash-Startwunsch der MS
M	0b111	0b00000010	0b000	0b00001	0b000	127	4	0x00	0x02	0x9C	0x03					Gib mir mal die Vendor-ID? Stufe oder Offset 1
S	0b111	0b00000010	0b000	0b00001	0b000	127	8	0x00	0x02	0x9C	0x03	0x00	0x00	0x00	0x97	Vendor-ID von MS
M	0b111	0b00000010	0b000	0b00010	0b000	127	4	0x00	0x02	0x9C	0x03					Gib SW/HW-Version
S	0b111	0b00000010	0b000	0b00010	0b000	127	8	0x00	0x02	0x9C	0x03	0x01	0x00	0x40	0x10	SW/HW-Version von MS
M	0b111	0b00000010	0b000	0b00011	0b000	127	4	0x00	0x02	0x9C	0x03					Abfrage 3 ??
S	0b111	0b00000010	0b000	0b00011	0b000	127	8	0x00	0x02	0x9C	0x03	0x01	0x00	0x00	0x67	Hier Antwort auf Abfrage 3
M	0b111	0b00000010	0b000	0b00000	0b000	127	4	0x00	0x02	0x9C	0x03					ACK, du kannst beginnen
S	0b111	0b00000010	0b000	0b00000	0b001	127	8	0x00	0x00	0x00	0x00	0x00	0x00	0x02	0x26	MS ⁶ benötigt ab Offset 3 550 Bytes
M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x00	0x00	0x00	0x35	0x09	0xF8	0xA9	Blockübergabe mit Flash-Flag
M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x00	0x00	0x04	0xD0	0x53	0x04	0xAE	= 550 / 4 = 137,5 Blk.
M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x00	0x00	0x08	0xE3	0xC3	0x19	0x51	

M	0b111	0b00000010	0b000	0b00000	0b010	127	6	0x00	0x00	0x02	0x24	0x36	0x74			Ende 1. Block (nur 2 Nutzdatenbytes)
M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x00	0x02	0x26	0x00	0x00	0x02	0x26	Zweiter Block mit Offset 550 Bytes und
M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x00	0x02	0x26	0xBB	0x26	0xc4	0xc6	Länge 550 Bytes Nutzdaten

M	0b111	0b00000010	0b000	0b00000	0b010	127	8	0x00	0x01	0x12	0xfa	0xec	0x6f	0x54	0xc6	Letzter Frame von letztem Block
M	0b111	0b00000010	0b000	0b00000	0b010	127	6	0x00	0x01	0x12	0xfe	0xec	0x6f			Die letzten 2 Nutzdatenbytes
S	0b111	0b00000010	0b000	0b00011	0b000	127	4	0x00	0x02	0x9C	0x03					Flashen beendet

⁶ Der Kommentar ist hier über die folgenden 2 Zeilen umgebrochen

9.10 Das BCI-File-Format

Die Firmware-Updates werden in Dateien mit der Endung BCI von ESU bereitgestellt. Diesem Umstand verdankt dies Kapitel seinen Namen. Updates für die MS1 (ggf. auch anderer Slaves) werden im Master zwischengespeichert, wobei der Nutzdatenanteil unverändert über den CAN-Bus an den Slave gesendet wird. Aus diesem Grund wurde auch dieses Datenformat (ansatzweise) analysiert.

Leider konnten wir über das verwendete BCI-Fileformat im Web nichts Passendes finden. Wir nehmen an, dass BCI so was wie **B**inary **C**ode **I**mage bedeutet.

Durch Probieren und Vergleich mit mehreren unterschiedlichen BCI-Files (von verschiedenen CS1-Ständen) konnten wir folgenden Aufbau ermitteln:

File-Header: (192 Bytes)

Nutzdaten: (70400 Bytes)

Die Größe des Headers steht wiederum selbst im Header. Es gilt zu beachten, dass die Werte im Motorola-Format (Big Endian) abgelegt sind.

Hier ein paar Offsets im Header:

Offset	Länge (Byte)	Bedeutung	Häufiger Wert
0x18	4	Offset für Nutzdaten	0xC0 -> ab 192 Nutzdaten
0x1C	4	Nutzdatengröße	0x00 0x01 0x13 0x00 -> 70400
0x64	2	SW-Version	0x01 0x03 -> V 1.3
0x6E	2	HW-Version	0x40 0x10 -> V 40.10
0x90	32 (max)	Gerätenamen	„Mobile Station MFx Motorola“
0xB0	4	Vendor-ID	0x97 = 151 = ESU

SW- und HW-Version kommen wiederholt im Header vor!

Ab Headergröße (oft 192 Bytes = Offset 0xC0) kommen dann die Nutzdaten (in unserem Fall für die MS bisher immer genau 70400 Bytes. Da dieser Wert wesentlich größer als die Flash-Größe der MS von 64kByte ist, drängt sich da die Vermutung auf, dass in diesem BCI-File vermutlich noch irgendwelche Checksummen oder ähnliches enthalten sind. Die Übertragung auf dem CAN-Bus legt zudem nahe, dass die Nutzdaten in Gruppen zu 550 Bytes gegliedert sind, die wiederum 512 Bytes der Firmware enthalten. Wie haben diese internen Strukturen bisher nicht weiter untersucht und planen das auch nicht durchzuführen.

Die Nutzdaten in dem BCI-File werden beim Flashen 1:1 in 4-Byte-Paketen auf dem CAN übertragen.

10 Offene Punkte

Allen Beteiligten ist klar, dass bisher nicht alle Geheimnisse des ECoS / MS1 / CS1-CAN-Protokolls gelüftet sind und sich auch noch neu Geheimnisse auftun können. Folgende Punkte sind heute noch weiter zu analysieren.

- ECoSlink, d.h. Protokolle die nicht von der MS1 benutzt werden
- Zyklische Überwachung bei Wiederkehr des Slaves, wenn der Master eine Zyklusverzögerung detektiert hat.
- Telegramme, die im Kapitel „6.7.6 Bisher ungeklärte Telegramme“ aufgeführt sind.

Versionsdatum: 18.10.2009		Version 1.1
CAN_Doku_V101.doc		Seite 52 von 56

11 Telegramm Trace

In diesem Kapitel wird der Telegramm Trace einer MS1 an eine ECoS Zentrale dargestellt. Hierbei wurde an eine laufende Zentrale (ECoS mit Firmware 3.0.1) eine MS1 angesteckt. Im Lokstack befand sich nur die Lok 003 131-0. Die Daten dieser Lok wurden direkt aus dem Lok-Katalog ohne Anpassung benutzt.

Dieser Trace ist ungefiltert, d.h. auch Telegramme die bisher nicht analysiert sind, werden hier dargestellt.

Die Spalten links der Spalte DLC stellen in binärer Form (0b...) die CAN-ID dar. Data[0 .. 7] zeigt die Nutzdaten in hexadezimaler Darstellung. ASCII [0 .. 7] versucht die Nutzdaten als Zeichen zu interpretieren.

Priorität Bit28 .. 26	Objekt-Handle		Befehl Bit 9 .. 7	Knotennummer Bit 6 .. 0	DLC	Data [0 .. 7] hex								ASCII [0 .. 7]								
	Bit 25 .. 10	Bit 11 .. 0				0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
000	00000000	00000000	100	0000001	1	00									-							
100	00000000	00000000	101	0000001	2	00	00								-	-						
111	00000000	00000000	000	1111110	0																	
111	00000000	00001100	001	1111110	8	00	04	84	D3	00	01	00	01	-	-	-	-	-	-	-	-	-
111	00000010	00101100	000	1111110	0																	
111	00000010	00101100	001	1111110	8	00	00	00	97	00	00	01	03	-	-	-	-	-	-	-	-	-
111	00000100	01001100	000	1111110	0																	
111	00000100	01001100	001	1111110	8	01	13	00	05	00	00	00	00	-	-	-	-	-	-	-	-	-
111	10110011	01101100	000	1111110	0																	
111	10110011	01101100	001	1111110	8	00	00	00	01	03	00	00	01	-	-	-	-	-	-	-	-	-
111	00000000	10001100	000	1111110	8	00	02	04	B3	00	02	00	03	-	-	-	-	-	-	-	-	-
111	00000000	10101100	000	1111110	8	00	00	00	97	01	00	40	10	-	-	-	-	-	-	-	@	-
111	00000000	11001100	000	1111110	8	01	00	00	67	00	00	40	10	-	-	-	g	-	-	-	@	-
111	00000000	11101100	000	1111110	8	00	00	40	10	01	08	00	10	-	-	@	-	-	-	-	-	-
011	00000000	00000001	111	0101010	8	00	00	00	2B	00	02	04	B3	-	-	-	+	-	-	-	-	-
111	00000000	11101100	001	1111110	8	00	02	04	B3	00	02	AB	00	-	-	-	-	-	-	-	-	-
110	00000000	00000010	100	0101011	2	40	2B							@	+							
110	00000000	00000010	100	0101010	8	40	2B	00	00	00	08	00	01	@	+	-	-	-	-	-	-	-
110	00000000	00001000	001	0101010	4	41	00	00	00					A	-	-	-					

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	Objekt-Handle		Befehl	Knotennummer	DLC	Data [0 .. 7] hex								ASCII [0 .. 7]							
	Bit28 .. 26	Bit 25 .. 10				Bit 9 .. 7	Bit 6 .. 0	0	1	2	3	4	5	6	7	0	1	2	3	4	5
110	00000000	00001000	001	0101010	4	02	00	00	00					-	-	-	-				
110	00000000	00001000	001	0101010	4	02	00	00	04					-	-	-	-				
110	00000000	00001000	001	0101010	4	02	00	00	08					-	-	-	-				
110	00000000	00001000	001	0101011	6	41	00	00	00	00	0A			A	-	-	-	-			
110	00000000	00001000	001	0101011	8	02	00	00	00	4B	53	20	4D	-	-	-	-	K	S		M
110	00000000	00001000	001	0101011	7	02	00	00	04	53	31	00		-	-	-	-	S	1	-	
110	00000000	00001000	001	0101011	5	02	00	00	08	00				-	-	-	-	-			
110	00000000	00000010	011	0101011	2	03	00							-	-						
110	00000000	00001000	001	0101010	4	02	00	00	0C					-	-	-	-				
110	00000000	00001000	001	0101011	5	02	00	00	0C	00				-	-	-	-	-			
110	00000000	00000010	011	0101010	8	03	00	00	00	00	03	00	01	-	-	-	-	-	-	-	-
110	00000000	00000011	010	0101011	2	01	01							-	-						
110	00000000	00000011	010	0101010	3	01	01	01						-	-	-					
110	00000000	00000011	010	0101011	2	01	00							-	-						
100	00000000	00000011	010	0101010	3	01	00	00						-	-	-					
110	00000000	00000011	010	0101011	2	02	01							-	-						
110	00000000	00000011	010	0101010	3	02	01	02						-	-	-					
110	00000000	00000011	010	0101011	2	02	00							-	-						
100	00000000	00000011	010	0101010	3	02	00	00						-	-	-					
110	00000000	00000011	010	0101011	2	03	01							-	-						
110	00000000	00000011	010	0101010	3	03	01	04						-	-	-					
110	00000000	00000011	010	0101011	2	03	00							-	-						
100	00000000	00000011	010	0101010	3	03	00	00						-	-	-					
110	00000000	00000011	010	0101011	2	04	01							-	-						
110	00000000	00000011	010	0101010	2	04	01							-	-						
110	00000000	00000010	011	0101011	4	80	02	00	00					-	-	-	-				
110	00000000	00000010	011	0101010	8	80	02	00	00	00	09	00	01	-	-	-	-	-	-	-	-
110	00000000	00000010	011	0101011	4	80	02	00	09					-	-	-	-				
110	00000000	00001001	001	0101011	4	40	03	00	00					@	-	-	-				

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	Objekt-Handle		Befehl	Knotennummer	DLC	Data [0 .. 7] hex								ASCII [0 .. 7]							
	Bit28 .. 26	Bit 25 .. 10				Bit 9 .. 7	Bit 6 .. 0	0	1	2	3	4	5	6	7	0	1	2	3	4	5
110	00000000	00000010	011	0101010	4	80	02	00	09					-	-	-	-				
110	00000000	00001001	001	0101010	6	40	03	00	00	00	03			@	-	-	-	-	-		
110	00000000	00000010	011	0101011	4	81	02	00	00					-	-	-	-				
110	00000000	00001001	001	0101011	4	40	01	00	00					@	-	-	-				
110	00000000	00000010	011	0101010	4	81	02	00	00					-	-	-	-				
110	00000000	00001001	001	0101010	8	40	01	00	00	00	03	00	00	@	-	-	-	-	-	-	-
110	00000000	00000010	011	0101011	4	82	02	00	00					-	-	-	-				
110	00000000	00001001	001	0101011	4	40	01	01	00					@	-	-	-				
110	00000000	00000010	011	0101010	4	82	02	00	00					-	-	-	-				
000	01000000	00000000	010	0000110	4	03	06	71	66					-	-	q	f				
110	00000000	00001001	001	0101010	8	40	01	01	00	00	00	00	00	@	-	-	-	-	-	-	-
110	00000000	00001001	010	0101011	2	01	01							-	-						
110	00000000	00001001	010	0101010	4	01	01	01	01					-	-	-	-				
110	00000000	00001001	010	0101011	2	01	00							-	-						
110	00000000	00001001	010	0101010	3	01	00	00						-	-	-					
110	00000000	00001001	010	0101011	2	02	01							-	-						
110	00000000	00001001	010	0101010	2	02	01							-	-						
110	00000000	00001001	010	0101011	4	01	02	00	00					-	-	-	-				
110	00000000	00001001	010	0101010	5	01	02	00	00	00				-	-	-	-	-			
110	00000000	00001001	010	0101011	4	01	02	01	00					-	-	-	-				
110	00000000	00001001	010	0101010	5	01	02	01	00	00				-	-	-	-	-			
110	00000000	00001001	010	0101011	4	01	02	02	00					-	-	-	-				
110	00000000	00001001	010	0101010	5	01	02	02	00	00				-	-	-	-	-			
110	00000000	00001001	001	0101011	4	40	04	00	00					@	-	-	-				
110	00000000	00001001	001	0101010	6	40	04	00	00	00	00			@	-	-	-	-	-		
110	00000000	00001001	000	0101011	2	03	00							-	-						
110	00000000	00001001	000	0101010	6	03	00	00	00	00	01			-	-	-	-	-	-		
110	00000000	00001001	001	0101011	4	02	00	00	00					-	-	-	-				
110	00000000	00001001	001	0101010	8	02	00	00	00	30	30	33	20	-	-	-	-	0	0	3	

Beschreibung des ECoS / MS1 / CS1 CAN-Protokolls für Entwickler

Priorität	Objekt-Handle		Befehl	Knotennummer	DLC	Data [0 .. 7] hex								ASCII [0 .. 7]							
	Bit28 .. 26	Bit 25 .. 10				Bit 9 .. 7	Bit 6 .. 0	0	1	2	3	4	5	6	7	0	1	2	3	4	5
110	00000000	00001001	001	0101011	4	02	00	00	04					-	-	-	-				
110	00000000	00001001	001	0101010	8	02	00	00	04	31	33	31	2D	-	-	-	-	1	3	1	-
110	00000000	00001001	001	0101011	4	02	00	00	08					-	-	-	-				
110	00000000	00001001	001	0101010	8	02	00	00	08	30	00	00	00	-	-	-	-	0	-	-	-
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				
011	00000000	00000001	111	0101010	8	00	00	00	2A	00	04	84	D3	-	-	-	*	-	-	-	-
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				
011	00000000	00000001	111	0101010	8	00	00	00	2A	00	04	84	D3	-	-	-	*	-	-	-	-
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				
011	00000000	00000001	111	0101010	8	00	00	00	2A	00	04	84	D3	-	-	-	*	-	-	-	-
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				
011	00000000	00000001	111	0101010	8	00	00	00	2A	00	04	84	D3	-	-	-	*	-	-	-	-
101	00000000	00001001	000	0101011	6	03	00	00	08	00	01			-	-	-	-	-	-		
101	00000000	00001001	000	0101010	6	03	00	00	08	00	01			-	-	-	-	-	-		
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				
011	00000000	00000001	111	0101010	8	00	00	00	2A	00	04	84	D3	-	-	-	*	-	-	-	-
000	01000000	00000000	010	0000110	4	03	07	79	37					-	-	y	7				
000	00000000	00000000	100	0000001	1	00								-							
100	00000000	00000000	101	0000001	2	00	00							-	-						
011	00000000	00000001	111	0101011	4	00	00	00	2A					-	-	-	*				