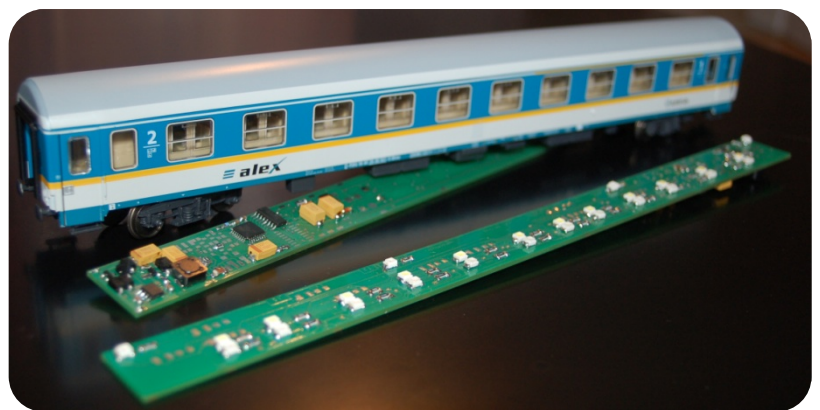


DCCrail V4.x

**Universal Waggondecoder für Abteil-, Schlaf- Doppelstock- und
Mittelgangwaggons mit zahlreichen Funktionen**



Version 1.6

Einleitung

Diese Anleitung bezieht sich auf die DCC-Wagendecoder, (DCCrail V4) von **Christoph Schörner und Toralf Wilhelm**. Der Decoder versteht sich nicht als kommerzielles Fertigprodukt, sondern ist eine Entwicklungshilfe für technisch interessierte Modellbahner zum Eigenbau von DCC Wagendecoder.

Der Decoder basiert auf den Mikrocontroller der Firma Atmel und ist nach Typ mit unterschiedlicher Peripherie und Funktionen ausgestattet.

Hier noch einmal ein klarer Hinweis:

Der Decoder und diese Anleitung wurden sorgfältig geprüft und nach bestem Wissen erstellt. Für die hier dargebotenen Informationen wird kein Anspruch auf Vollständigkeit, Aktualität, Qualität und Richtigkeit erhoben. Es kann keine Verantwortung für Schäden übernommen werden, die durch das Vertrauen auf die Inhalte dieser Anleitung, der Decoder oder deren Gebrauch entstehen.

Die Software der Decoder steht als Quelltext auf meiner Internetseite zur Verfügung und darf von jedem benutzt, erweitert und verbessert werden.

Eine kommerzielle Nutzung der Software oder Teile daraus erfordert zwingend mein Einverständnis!

Die Decoder sind mit einem Bootloader zum einfachen Softwareupdate ausgestattet, welcher von Hagen aus dem Mikrocontrollerboard stammt.

(<http://www.mikrocontroller.net/topic/95839>)

Sicherheitshinweise

Unsachgemäßer Gebrauch und Nichtbeachtung der Anleitung können zu unkalkulierbaren Gefährdungen führen.

Gewährleistung

Die Verwendung dieser Betriebsanleitung ist nur für den Nachbau und den Eigenbedarf des beschriebenen Bausteins erlaubt. Eine anderweitige Nutzung bedarf der schriftlichen Einwilligung des Verfassers.

Für den Nachbau und dessen Funktionen des beschriebenen Bausteins übernimmt der Verfasser keinerlei Haftung. Für die Einhaltung bestehender Vorschriften und dem vorschriftsmäßigen Einsatz des Produkts ist der Betreiber alleine verantwortlich.

Inhaltsverzeichnis

1. DCCrail V4.....	4
a. elektrische Parameter	
b. verfügbare Funktionen	
c. Bauteilliste	
2. Aufbaubeschreibung	6
3. nachträgliche Verbesserungen / Erweiterungen.....	11
a. Softwareversionen	
b. Änderungsanweisungen	
4. Bestückungsplan der verschiedenen Varianten.....	13
a. Variante A	
b. Variante B	
c. Variante C	
d. Variante D	
e. Sonderfunktionen	
5. Softwareupdate.....	16
6. Schaltung / Layout.....	17
7. CV – Konfigurationsvariablen.....	19

1. DCCrail V4:

Der Dccrail Decoder ist ein universal Waggondecoder für Abteil-, Schlaf-, Doppelstock- und Mittelgangwaggons (2.Klasse). Die Platine hat ungekürzt 28cm und kann auf 26cm, 20cm, 13cm und 10cm gekürzt werden. Alle Bauteile sind in Elektronikwarenhäusern z.B. Reichelt erhältlich. Platine kann bei **fichtelbahn.de** bezogen werden. Ein Nachbau ist auch für den nicht geübten Hobbyisten möglich. Der Decoder passiert auf den Atmega8 Mikrocontroller der 2,7V Version.



a. elektrische Parameter:

DCC Eingangsspannung:	27V
Funktionsausgänge:	16 Stück je maximal 500mA
maximal zulässiger Gesamtstrom:	500mA
Analogbetrieb:	ab 7V Gleisspannung möglich
Pufferung:	möglich, mit Tantal 1F o. mit LiPo Zelle 3,7V ... wir empfehlen mit einer LiPo-Zelle mit Tiefentladungsschutz sonst besteht bei extremer Entladung „Brandgefahr“ ... eine LiPo-Trennung erhöht die Lebensdauer Ihrer Zelle (siehe Änderungsanweisung)

b. Funktionen:

Jeder Ausgang kann wahlweise mit verschiedenen Funktionen belegt werden.
Demo – Video auf der Homepage www.fichtelbahn.de vorhanden!

- **normal Beleuchtung**
- **Neonlicht** (Flackern beim einschalten)
- **defekte Lampe**
(simuliert ein erneutes Zünden, defektes Leuchtmittel im Zufallmodus)
- **Abteil - Simulation**
(Zufall Abteilbeleuchtung an/aus, einzelne Abteile werden per Zufall ein und ausgeschalten)
- **Nachtlicht - Simulation**
(durch auslösen einer weiteren Funktionstaste werden alle Abteile per Zufall in den Nachtmodus geschalten.
Durch deaktivieren der Funktionstaste wird per Zufall zurück geschalten.

c. Bauteilliste:

1x	A78L05	IC2
4x	B140F	D1, D2, D9, D10
2x	BC849C	T1, T3
1x	BCV26/BCV28	T2
1x	BCX51	T2 (Alternative für T2)
1x	BC858	Q5 (muss zusätzlich bestückt werden bei Alternative)
1x	B240F	D5
3x	LL103A	D4, D6*, D11
1x	ZD27V	D7
1x	ZF4V3	D3
1x	Atmega8L8AU [TQFP-32]	IC4
1x	L-PIS2816 47µH	L1
1x	L-1206AS 1,0µH	L3
4x	100nF / 0805	C8, C11, C15, C21
2x	100nF / 0603	C12, C23
1x	10µF/35V Tantal Typ D	C10
1x	0 Ohm / 1206	R53
1x	20 kOhm / 1206	R62
1x	4,7 kOhm / 1206	R61
1x	47 kOhm / axial	R65
2x	1 Ohm / 0805	R54, R58
1x	1 kOhm / 0805	R60
2x	10 kOhm / 0805	R9, R59
2x	100 kOhm / 0805	R56, R57
1x	220 Ohm / 0805	R63
1x	18kOhm / 0603	R68
1x	47 kOhm / 0402	R2
1x	100Ohm / 1206	R8, R66
2-4x	220µF/10V Typ D oder auch 100µF / 6,3V, entscheiden ist die Tantal-Bauform Typ D und mindestens 6,3V	C1, C2, C9, C13
1x	DCCrail V4 Platine	
1-2x	ULN2003	IC1, IC3

für Variante B, C und D sind 2 Stück notwendig

Beleuchtung:

LED1206 / PLCC-2	Anzahl von Bestückung abhängig
100 Ohm / 1206	Anzahl von Bestückung abhängig

Schlusslicht-, Spitzenlicht-, Kupplungsfunktion:

4x	BC817	Q1, Q2, Q3, Q4
4x	2,2 kOhm / 0805	R4, R5, R51, R52
2x	270 Ohm / 1206	R6, R7
2x	150 Ohm / 0603	R1, R3*

optionale Pufferung mit LiPo-Trennung:

1x	3,7V LiPo Akku mit Entladeschutz oder GoldCap 5,5V (mind. 1F)	
1x	BC807	T4
1x	470 Ohm / 0603	R3*
1x	LL103A	D8*

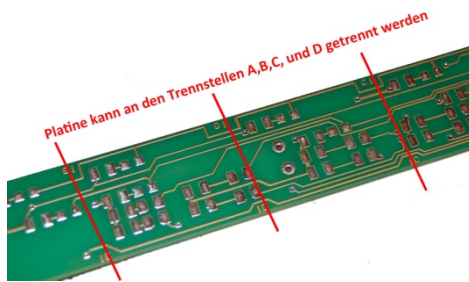
*der Widerstandwert ist von der gewählten Option abhängig

2. Aufbaubeschreibung:

Für den Aufbau ist nicht viel nötig. Ein LötKolben, mit feiner Lötspitze, Lötzinn maximal 1 mm stark, Flussmittel, eine feine Pinzette, ein Seitenschneider, ein Ohmmeter und eventuell etwas Entlötlitze. Zum Programmieren ist ein Programmer und die passende Software dafür nötig. Ich verwende Atmels AVR-Studio und einen „AVRICE mkII“.

Alle nötigen Bauteile können bei **Reichelt Elektronik** bezogen werden und die Wagenplatine gibt es bei **Fichtelbahn.de** (support@fichtelbahn.de).

1.Schritt: Platine teilen



Die Platine kann an Ihren Marken A, B, C und D mit einer kleinen Säge getrennt werden und somit in allen Waggons wunschgemäß eingebaut werden.

Wichtig: Diese Trennung sollte sorgfältig und nur an den gekennzeichneten Marken passieren, dass keine weiteren Leiterbahnen beschädigt oder herausgerissen werden. Nach der Trennung an der Schnittkante nach eventuellen Leiterbahnkurzschlüsse kontrollieren und diese gegeben falls entfernen, ein Ohmmeter kann hierbei behilflich sein.

Folgende Platinenlängen sind möglich:

A: 10cm

B: 13cm

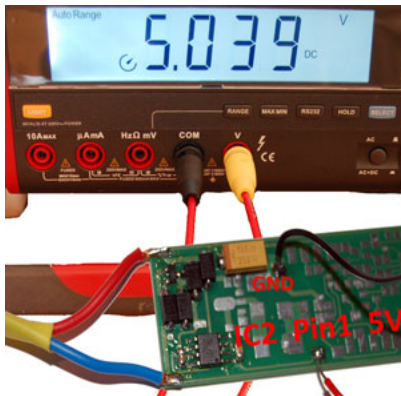
C: 20cm

D: 26cm

E (original Länge): 28cm

2.Schritt: Spannungsversorgung μ C

Als erstes bestücken wir die Platine mit den Gleichrichterdiode **D1, D2, D9 und D10**. Folgend die Schutzdiode **D7, C10** den Glättungskondensator und Spannungsregler 78L05SMD **IC2** mit den beiden Abblockkondensatoren 100nF **C8** und **C11**.

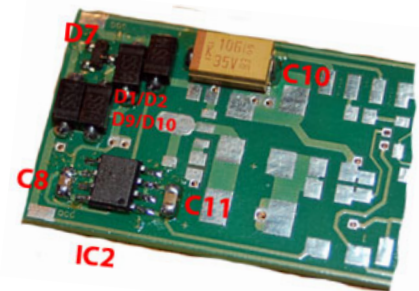


Kontrolle:

Zum testen, wird eine 9-18 Volt AC / DC Spannung oder das DCC-Signal angelegt.

An dem Spulen-PAD **L2** sollte jetzt eine Spannung von **5 Volt** messbar sein.

Folgen Sie den nächsten Abschnitt nur bei erfolgreicher Spannungsmessung.



Weiter geht es mit der Beschaltung u. Versorgung um den μ C:

Abblockkondensatoren: **C12, C21, C23**

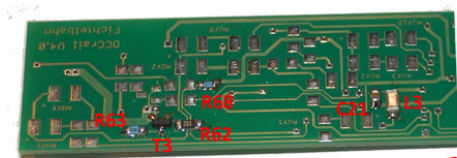
SMD-Spulen: **L3**

SMD-Diode: **D4**

SMD-Widerstände: **R2, R9, R62, R63, R65, R68**

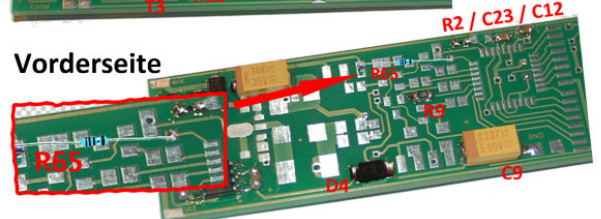
SMD-Transistor: **T3**

Tantal-Kondensator: **C9**



Rückseite

Vorderseite



Besonderheit: Der R65 wurde nachträglich für eine bessere VCC-Regelung eingefügt, deshalb ein „fliegender Aufbau“. Dieser Axial-Widerstand ist für den Betrieb notwendig. Um einen eventuellen Kurzschluss zu vermeiden können die Drähte mit einem Strumpfschlauch überzogen werden.

Hinweis: Lieber etwas länger die Beine, weil die darunter liegende Schaltung wird evtl. auch noch bestückt!!

3.Schritt: Spannungsversorgung Verbraucher

Weiter geht es mit den Bauteilen für den Schaltregler
(Spannungsversorgung für die Verbraucher):

Abblockkondensator: C15
SMD-Diode: D5, (D6), D3, D11
SMD-Spulen: L1
SMD-Widerstände: R8, R56, R57, R58, R59,
R60, R61, R66
SMD-Transistor: T1, T2 (alternative Q5)
Tantal-Kondensator: C1, C2, C13

Wichtig:

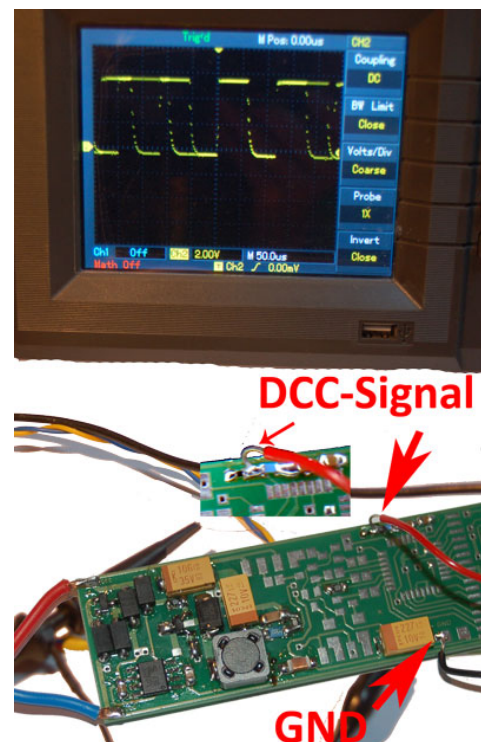
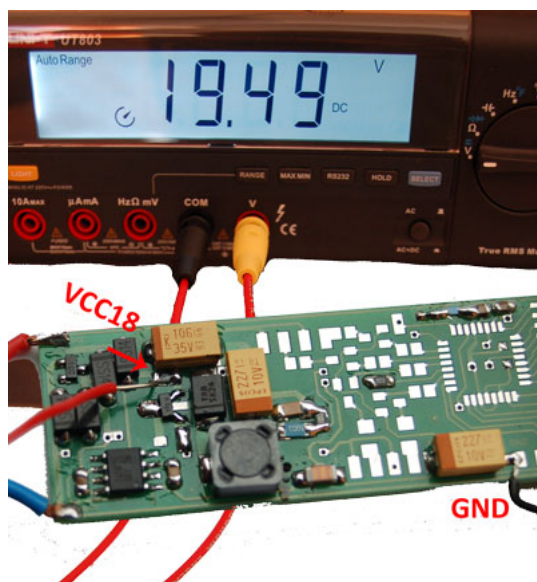
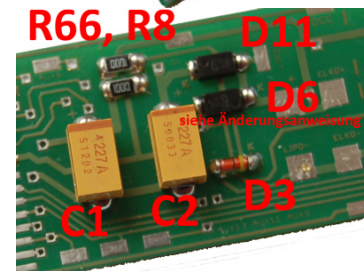
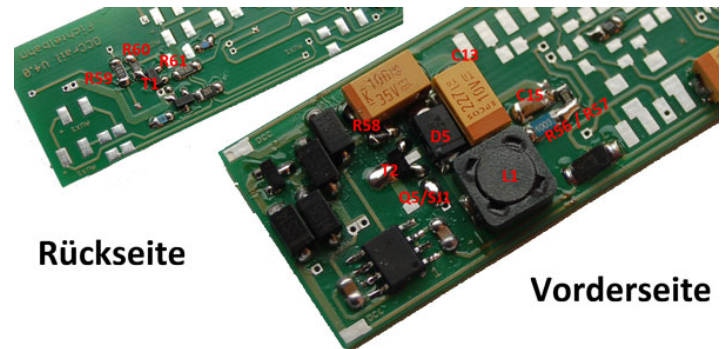
Bei der Alternativbestückung von T2 mit einen BCX52 und Q5 mit
einem BC858 ist Jumper SJ1 offen.
Bei der Standardbestückung mit einem Darlington Transistor BCV26 oder
BCV28 für T2 ist SJ1 geschlossen und Q5 nicht bestückt.

Bemerkung:

D6 sollte nur bestückt werden, wenn keine LiPo-Trennung erwünscht ist.
Wir empfehlen mit LiPo-Trennung
(siehe Änderungsanweisung)

Kontrolle:

Zum Testen wird an den Decoder ein DCC Signal angelegt und
mit einem Multimeter und falls zur Hand einem Scope das DCC-
Signal am Widerstand R2 gemessen. Eine Spannung um die 18V
(je nach Betriebsspannung des Boosters) sollte am Widerstand
R58 messbar sein.



Folgen Sie den nächsten Bauabschnitt nur bei erfolgreicher Spannungsmessung 18V.

4.Schritt: Programmierung und Einbau des Prozessor

Programmierung

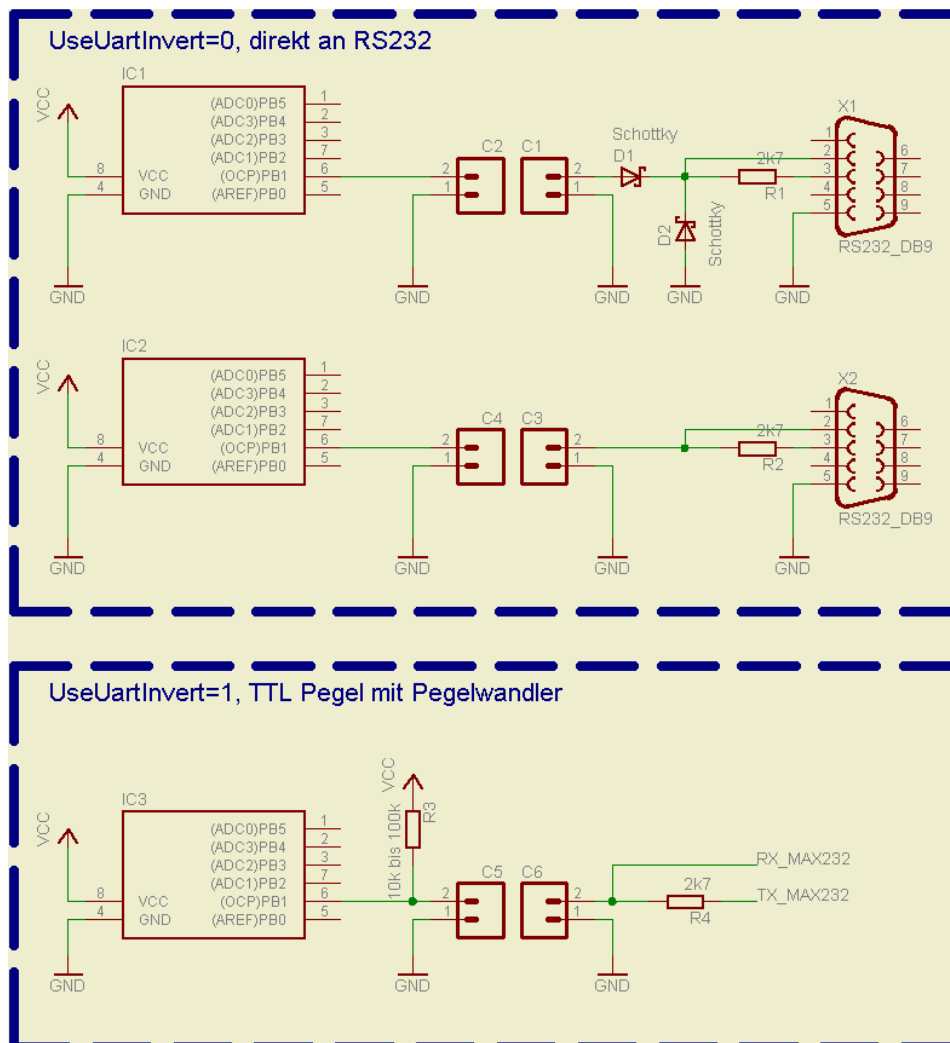
Die Software für den Decoder basiert auf folgendem Konzept:

Im AVR wird zuerst ein Bootloader (AVRootloader.hex) installiert. Das ist ein kleines Programm, welches uns später erlaubt, die eigentliche Software für unseren DCC Decoder einfach und unkompliziert über eine 2 Draht Verbindung (Masse + Signal) auszutauschen. So ist es dann jeder Zeit möglich, den Decoder im eingebauten Zustand mit einer neuen Firmware auszustatten und das ohne speziellen AVR-Programmer.

Die eigentliche DCC Software besteht aus zwei Teilen, dem Programm und den CV – Daten. Das **Programm (*.hex)** und die passende **Variablendatei (*.eep)**, diese enthält die CV's des Decoders. Diese werden im EEPROM des AVR's abgelegt und sind so im Betrieb veränderbar.

Vorbereitung:

- Für den Bootloaderbetrieb ist noch ein 1-wire Kabel nötig, dieses lässt sich leicht selbst anfertigen:

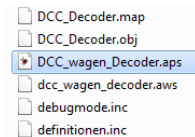


- Als nächstes müssen wir den internen RC-Oszillator für den Atmega8 kalibrieren (OSCAL). Das klingt schlimm ist aber nicht aufwendig:

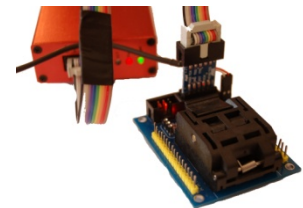
WICHTIG:

Dieser Vorgang muss bei jedem neuen μC ausgeführt werden und in die Firmware eingetragen werden.

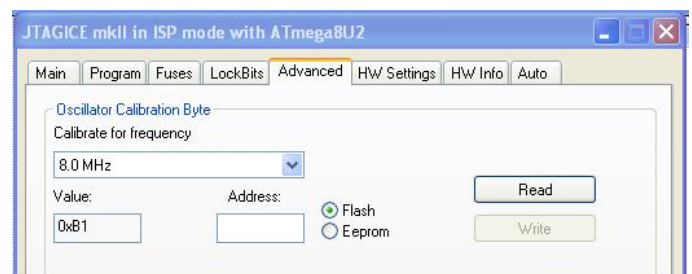
- 1) Die Firmware öffnen mit dem AVR-Studio
(Datei: DCC_wagen_decoder.aps)



- 2) Den Atmega8 in einen passenden Sockel legen und diesen über ISP mit dem Programmer verbinden.

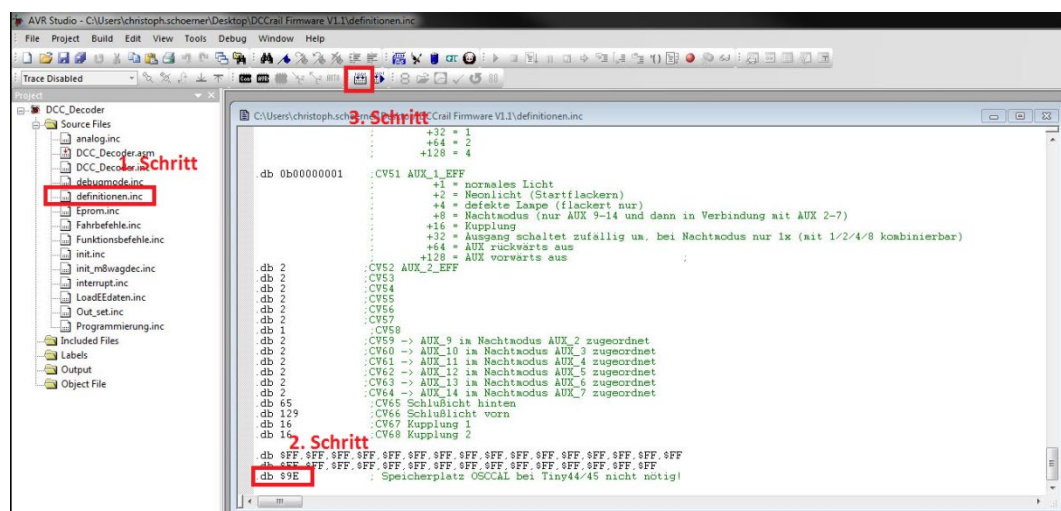


- 3) Aus dem AVR-Studio den Programmer aufrufen, den passenden μC wählen und im Register „Advanced“ den Oscal-Wert „Value“ auslesen.
(in diesem Beispiel ist es „B1“).



Diesen Wert sollte auf der Platine vermerkt werden, weil der Oscal-Wert für jedes Update benötigt wird.

- 4) Als nächsten Schritt muss der ermittelte Wert in die Firmware eingetragen werden.



Dazu muss die Datei „definitionen“ geöffnet werden und am Ende der Datei findet man den Eintrag „ Speicherplatz OSCAL“

Hier wird bestehende Wert (9E) durch Ihren ermittelten Wert (in unserem Beispiel B1) ersetzt.

Als letzter Schritt muss das komplette Projekt kompiliert werden. In diesem Zuge werden die beiden Datei *.hex und *.eep in dem Projektordner erstellt.

Vorbereitung fertig, jetzt wird programmiert!

Wir müssen jetzt dem Decoder die richtigen Fuses setzen und den Bootloader flashen:

- Nun eine Verbindung zum AVR Programmer herstellen
- Im Main Ordner, unter Device den Atmega8 auswählen. Nun kann die Signatur des AVR's ausgelesen werden. Diese sollte richtig angezeigt werden. Wenn das so ist, besteht eine korrekte Verbindung zum Atmega8.
- Jetzt wechseln wir zum Reiter Fuses und tragen für den Atmega8 folgende zwei Werte ein:

HIGH: 0xDA

LOW: 0x84

dann auf **Program** klicken.

```
*****
* FUSE info: für 8MHz int. RC *
* MIT Bootloader *
*
* RSTDISBL:   nein(1) *      ;RESET Pin mit RESET Funktion
* WDTON:      nein(1) *      ;WDT beim Start aus
* SPIEN:      ja(0) *       ;serial programmieren erlaubt
* CKOPT:      nein(1) *      ;
* EESAVE:     nein(1) *      ;eeprom wird bei chiperase auch gelöscht
* BOOTSZ:     ja(0) *       ;BOOTSZ legt Bootloaderbereich fest hier 512 words
* BOOTRST:    ja(0) *       ;Reset Vektor auf Bootloader
* BODLEVEL:   ja(0) *       ;BOD bei ca 2.7V VCC
* BODEN:      ja(0) *       ;BOD ein
* SUT/CKSEL:  *            ;8MHz interner RC Oosc SUT 6CK+0ms
*
* High Byte: 0xDA *
* Low Byte: 0x84 *
*****
```

- Nun in den **Program** Reiter wechseln und bei Flash die AVRroolader.hex auswählen.
- dann auf **Program** klicken

Jetzt kann der Atmega8 auf die Platine gelötet werden ... richtige Einbaurichtung beachten.

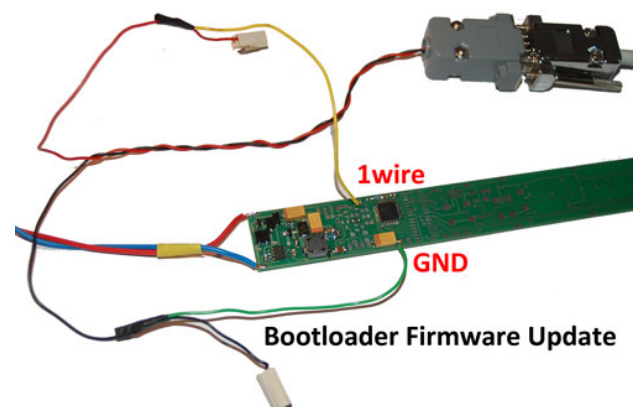
Tipp: Fixieren Sie die beiden quer überliegenden Pins auf der Platine, mit Zugabe von Flussmittel lassen sich die restlichen Pins problemlos verlöten.



Was fehlt jetzt noch?

Unser eigentliches Decoderprogramm natürlich. Bis jetzt versucht der Atmega8 nur ständig dieses am 1-wire Eingang einzulesen und genau das machen wir jetzt.

Nun müssen wir unseren Decoder mit der 1-wire Verbindung verkabeln (siehe Vorbereitung) und am Eingang eine DC Spannung anlegen.



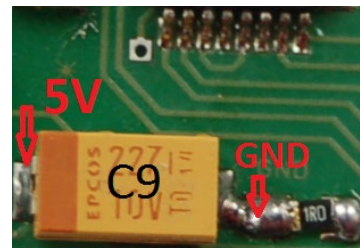
Firmware aufspielen:

- Wir starten jetzt aus dem Ordner:
/AVRooloader/Windows die
AVRrootloader.exe
wählen den Port und die Baudrate aus.
(Baudrate auf 9600)
- Bei den Programming Files wählen wir
unser Decodersoftware (*.hex) und unsere
Daten (*.eep) aus.
- Danach verbinden wir uns mit unserem
Atmega8 . **Connect to device.**
- Nun 1x auf Program klicken, kurz warten
und fertig ist der Decoder programmiert!
Wer mag, kann sich den EEPROM Inhalt
(das ist die DCC_Decoder.eep) ansehen.
Ansonsten . **Disconnect device**



Kontrolle:

Nach Anlegen eines DCC-Signals und einer aktiven Funktion (Funktionstasten F1-F7) sollte an dem Tantal-Kondensator C9 die Spannung 4,8V messbar sein.



Hilfe:

Falls keine Versorgungsspannung messbar ist sollten Sie nochmals den Abschnitt "Versorgungsspannung Verbraucher - Schaltregler" überprüfen.

Mit einem Scope sollten Sie an dem SMD-Transistor T1, an der Basis ein StepDown-Signal messen können, wenn eine Funktion (Funktionstasten F0-F7) aktiv ist.

Der StepDown arbeitet nur bei aktiver Funktion.

3. nachträgliche Verbesserungen / Erweiterungen

a) Softwareversionen:

- aktuelle Version 4.1 [Basisversion mit LiPo-Trennung]

b) Änderungsanweisung:

Wir empfehlen sehr, den **LiPo-Betrieb** mit **LiPo-Trennung** als Pufferung zu verwenden. Bei einer DCC oder DC Unterbrechung ist die Platine noch für 60sec. gepuffert und führt ihre Funktion fort. Kommt innerhalb dieser Zeitspanne kein neues DCC oder DC Signal schaltet der μC alle Ports ab und geht schlafen.

Ohne einer LiPo-Trennung wird die komplette Schaltung weiterhin mit der LiPo-Spannung versorgt bis die Zelle von Ihrer eigenen Tiefentladungsschutzschaltung getrennt wird. Dies führt zur Reduzierung der Lebensdauer. Nach jedem Neustart der Platine wird die LiPo-Zelle auf Ihre Zellenspannung von 3,7V geladen. Das den Nachteil mit sich bringt, dass die angeschlossenen LED's mit steigender Ladespannung heller werden. Die volle Leuchtkraft der LED's ist erst mit Erreichen der Zellen Spannung von 3,7V gegeben. Ladezeit ca. 5 Minuten.

Der Clou an der LiPo-Trennung ist, dass der LiPo über einen Ausgang (AUX_17) geschalten wird. Legt der Atmega8 sich nach den 60sec schlafen, schaltet er alle Ausgänge ab und koppelt die Lipo-Zelle automatisch mit von der Schaltung.

Die LiPo-Zellen Spannung bleibt erhalten und bei erneutem Einschalten ist fast keine Ladung nötig, somit kommt es zu keinen heller werdenden LED's.

Hinweis: Die LiPo-Zelle hat auch eine Eigenentladung so wird bei langer Betriebspause trotzdem eine Ladevorgang nötig sein.

Diese Funktion ist auch als Standard in der Firmware ausgewählt.

Um diese Funktion zu nutzen müssen folgende Teile als Luftverdrahtung auf die Platine gebaut werden.

SMD-Transistor: **T4**

SMD-Diode: **D6**

Fädeldrahtverbindung zum Aux_17 GND

Mit Verwendung dieser Funktion ist der AUX_17 belegt, sollte diese Funktion nicht erwünscht sein kann durch ändern des CV32 der AUX_17 zu einen normalen Ausgang konfiguriert werden.



Bemerkung: Sie benötigen für die LiPo-Trennung noch einen vollwertigen Bestückten AUX_17 Ausgang (*siehe Bestückungsvarianten*).

Dieses Bild kann von Ihrer Bestückung abweichen - AUX_17 ist noch nicht bestückt!!

4. Bestückungsplan der verschiedenen Varianten

Variante A:

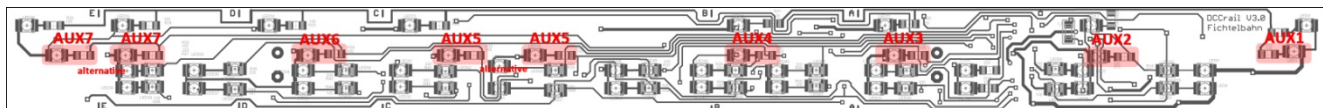
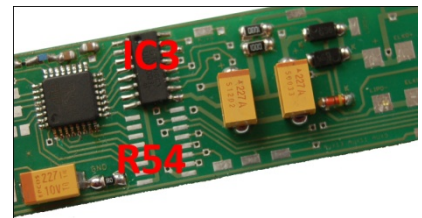
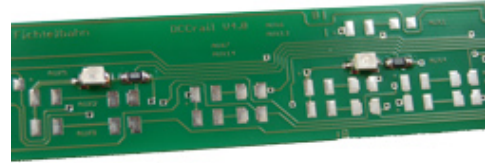
Mittelgangbestückung (2.Klasse)

Variante A ist ein Aufbau für den Mittelgang eines Waggons und kann je nach Länge der Platine mit 3-7 LED's der Farben gelb, warmweiß oder kaltweiß bestückt werden.

Sie benötigen:

IC3 als LED-Driver und den Sicherungswiderstand **R54**, auf der LED-Seite muss für **AUX_2** noch eine Widerstandsbrücke **R53** eingelötet werden.

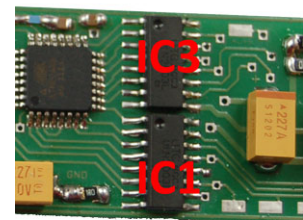
LED **1,2,3,18,19**(alternative **23**), **20,21**(alternative **32**)
AUX_1 bis AUX_7



Variante B:

Doppelstockwagen-Option (2.Klasse)

Variante B ist ein identischer Aufbau der Variante A, nur dass über den zweiten ULN2003 **IC1** weitere Ausgänge (PÄDs) für eine externe Platine oder Leuchtdioden zur Verfügung stehen. (Vorwiderstände nicht vergessen)



Variante C:

Gang- und Abteilwagenbestückung

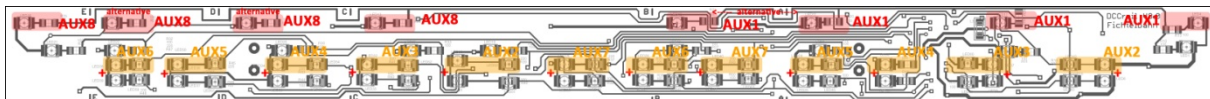
Variante C ist ein Aufbau mit Ganglicht und der ersten Abteilreihe für eine Abteil-Lichtsimation. Sie können je nach Länge der Platine 4-12 Abteile realisieren. Durch vertauschen der Widerstände mit den Leuchtdioden sollte fast jedes Abteil beleuchtet werden. Abhängig der Wagen-Epoche kann hier gelb, warmweiß oder kaltweiß bestückt werden. Es werden immer zwei Abteile gemeinsam geschaltet!



Sie benötigen:

IC3 und **IC1** als LED-Driver und den Sicherungswiderstand **R54**, auf der LED-Seite muss für **AUX_2** noch eine Widerstandsbrücke **R53** eingelötet werden.

Ganglicht:	LED 4,11,12,17	-> AUX_1	
	LED 13,14,15,16	-> AUX_8	... zwei Leuchtdioden pro AUX genügen!
Abteilbeleuchtung:	LED 5, (42), 30, (50)	-> AUX_2	
	LED 7, (60), 32 (52)	-> AUX_3	
	LED 9, 34, (54)	-> AUX_4	
	LED 24, (44), 36, (56)	-> AUX_5	
	LED 26, (46), 38, (58)	-> AUX_6	
	LED 28, (48) 40, (62)	-> AUX_7	... (??) ist die Alternative beim tauschen



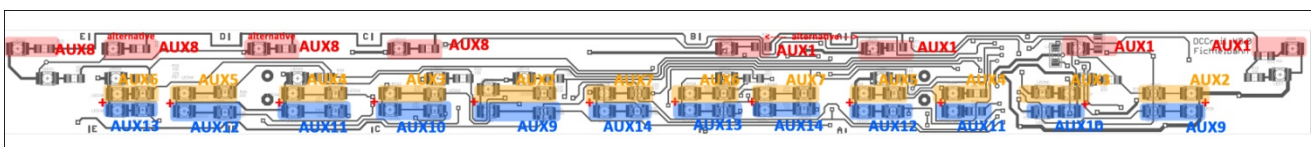
Variante D:

Nachtlicht-Option



Variante D ist ein identischer Aufbau der Variante C, nur dass die zweite Reihe der LED's mit bestückt werden für die Nachtlicht-Simulation.

Nachtlicht:	LED 6, (43), 31, (51)	-> AUX_9	
	LED 8, (61), 33 (53)	-> AUX_10	
	LED 10, 35, (55)	-> AUX_11	
	LED 25, (45), 37, (57)	-> AUX_12	
	LED 27, (47), 39, (59)	-> AUX_13	
	LED 29, (49) 41, (63)	-> AUX_14	... (??) ist die Alternative beim tauschen



Bemerkung:

Für alle Leuchtdioden ist ein Vorwiderstand notwendig, dieser Vorwiderstand ist von der LED-Spannung abhängig, ein guter Richtwert ist **100 Ohm**.

Sonderfunktionen:

Spitzlicht, Schlusslicht:

Mit den CV-Werten kann jedem Ausgang (AUX_15 oder AUX_16) individuell als Spitzlicht oder Schlusslicht konfiguriert werden.

Für diese Funktion müssen folgende Teile bestückt werden:

AUX_15:

SMD-Widerstand: **R4,R6** (R54 falls noch nicht vorhanden)
SMD-Transistor: **Q1**

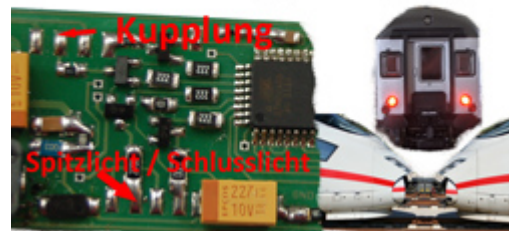
AUX_16:

SMD-Widerstand: **R5,R5** (R54 falls noch nicht vorhanden)
SMD-Transistor: **Q2**

Bemerkung:

Zur besseren Erkennung wurde der Axialwiderstand R65 auf dem Bild weggelassen!

Schlusslicht u. Spitzlicht sowie zwei Kuppelausgänge



Kupplungsfunktion:

Mit den CV-Werten kann den Ausgängen (AUX_17 oder AUX_18) eine Kupplungsfunktion zugewiesen werden. Es steht individuell jedem Ausgang ein Zeitglied von 1-7 Sekunden zur Verfügung.

Für diese Funktion müssen folgende Teile bestückt werden:

AUX_18:

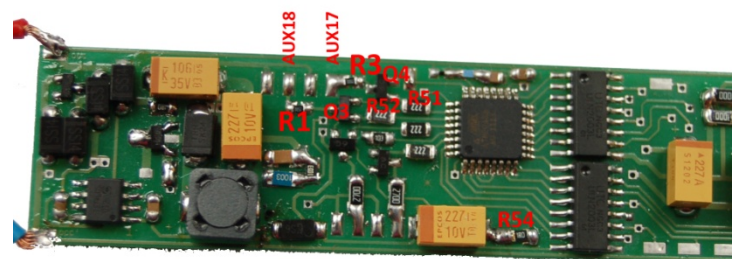
SMD-Widerstand: **R1,R51** (R54 falls noch nicht vorhanden)
SMD-Transistor: **Q3**

AUX_17:

SMD-Widerstand: **R3,R52** (R54 falls noch nicht vorhanden)
SMD-Transistor: **Q4**

Bemerkung:

Zur besseren Erkennung wurde der Axialwiderstand R65 auf dem Bild weggelassen!



Wichtiger Hinweis:

Die Funktion LiPo-Trennung ist als Standard auf den AUX_17 programmiert. Für diese Funktion muss AUX_17 vollständig bestückt werden mit **R3 als 470Ohm** und die Bestückung der „Änderungsanweisung“ beachten.

Falls keine LiPo-Trennung erwünscht ist, kann der AUX_17 zu einem vollwertigen weiteren Ausgang umgestellt werden mit der Änderung der CV32 auf 0. In diesem Fall ist für **R3 der Wert 150Ohm** zu bestücken.

5. Softwareupdate

Der Decoder ist mit einem Bootloader versehen, der es ermöglicht die Betriebssoftware mit Hilfe einer 2 Drahtverbindung (GND und Signalleitung) auszutauschen. Dazu muss der Decoder in den Update Modus gebracht werden. Dann kann mit Hilfe des AVR Bootloaders ein Softwareupdate durchgeführt werden.

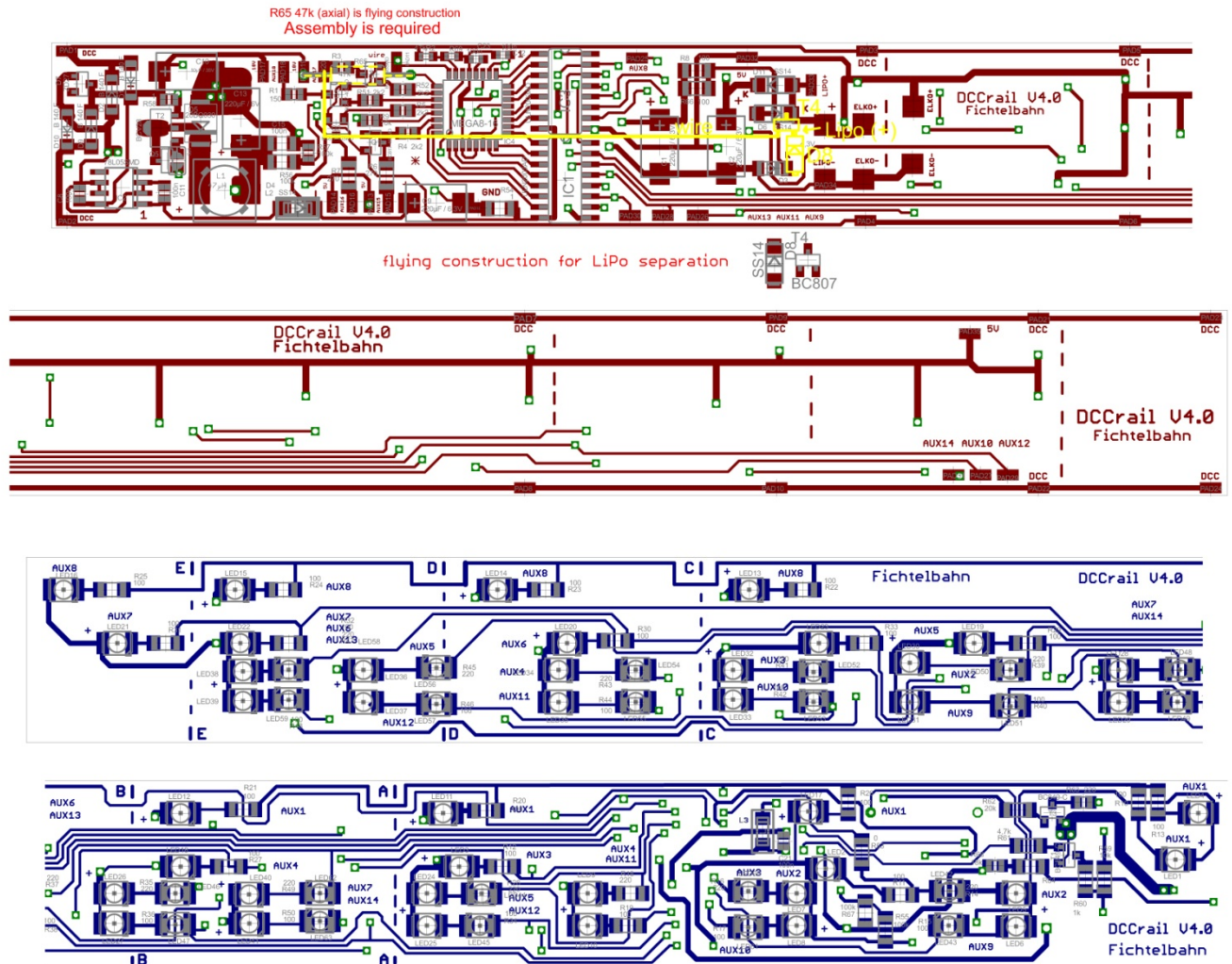
Vorgehensweise:

1. in CV30 eine 1 schreiben
2. Decoder jetzt mit Gleichspannung betreiben
3. 1-wire Kabel anschließen
4. AVR Bootloader starten, Com Port einstellen, Baudrate auf 9600 einstellen
5. *.hex und *.eep File auswählen

(Wichtig: zuvor den Oscal-Wert ermitteln und diesen in der Definitionsdatei ablegen und das Projekt neu kompilieren ... weitere Informationen im Punkt "Programmieren")

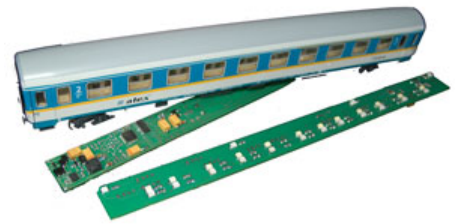
6. mit Decoder verbinden
7. Update starten
8. fertig

Darauf achten, dass auch das EEPROM File mit ausgetauscht wird!



Ein hochauflösendes pdf - Layout für den Nachbau ist in den Downloadunterlagen enthalten.

7. CV – Konfigurationsvariablen



<u>CV-Adresse</u>	<u>Default</u>	<u>Wert</u>	<u>Bezeichnung / Funktion</u>
CV1	3		Decoderadresse
CV7	„auslesen“		Software-Version
CV13	7		im Analogmodus aktive Funktionen +1: F0/FL ist an +2: F1 ist an +4: F2 ist an +8: F3 ist an +16: F4 ist an +32: F5 ist an +64: F6 ist an +128: F7 ist an
CV17	192		Erweiterte Adresse höherwertiges Byte der langen Adresse plus 192
CV18	100		Erweiterte Adresse niederwertiges Byte der langen Adresse
CV29	6		Konfiguration nach DCC-Norm +1: Fahr-Richtung umgekehrt +2: 28/128 Fahrstufen, sonst 14 +4: automatisch Analog/Digital-Erkennung, sonst nur Digitalbetrieb +32: Erweiterte (lange) Adresse (CV17 und CV18), sonst CV1
CV30	0		1 gleich Softwareupdate ein
CV31	205		U_LED 4,2V → ca. 205 (je nach AVR da die int. rev. unterschiedlich)
CV32	1		0 = AUX_17 als normaler Port 1 = LiPoetrieb (LiPo-Trennung mit dem Port AUX_17)

CV33	33	<p>AUX_1 Zuordnung zu einem Funktionsausgang +0 = F0/FL +1 = F1 +2 = F2 +3 = F3 +4 = F4 +5 = F5 +6 = F6 +7 = F7 +0 = Helligkeitsstufe 1 +32 = Helligkeitsstufe 2 +64 = Helligkeitsstufe 3 +96 = Helligkeitsstufe 4 +128 = Helligkeitsstufe 5 +160 = Helligkeitsstufe 6 +192 = Helligkeitsstufe 7 +224 = Helligkeitsstufe 8</p> <p>Bemerkung: Eine Funktion kann mit einer Helligkeitsstufe kombiniert werden!</p>
CV34	34	AUX_2 Zuordnung zu einem Funktionsausgang
CV35	34	AUX_3 Zuordnung zu einem Funktionsausgang
CV36	34	AUX_4 Zuordnung zu einem Funktionsausgang
CV37	34	AUX_5 Zuordnung zu einem Funktionsausgang
CV38	34	AUX_6 Zuordnung zu einem Funktionsausgang
CV39	34	AUX_7 Zuordnung zu einem Funktionsausgang
CV40	33	AUX_8 Zuordnung zu einem Funktionsausgang
CV41	227	AUX_9 Zuordnung zu einem Funktionsausgang [AUX_9 im Nachtmodus AUX_2 zugeordnet]
CV42	227	AUX_10 Zuordnung zu einem Funktionsausgang [AUX_10 im Nachtmodus AUX_3 zugeordnet]
CV43	227	AUX_11 Zuordnung zu einem Funktionsausgang [AUX_11 im Nachtmodus AUX_4 zugeordnet]
CV44	227	AUX_12 Zuordnung zu einem Funktionsausgang [AUX_12 im Nachtmodus AUX_5 zugeordnet]
CV45	227	AUX_13 Zuordnung zu einem Funktionsausgang [AUX_13 im Nachtmodus AUX_6 zugeordnet]
CV46	227	AUX_14 Zuordnung zu einem Funktionsausgang [AUX_14 im Nachtmodus AUX_7 zugeordnet]
CV47	224	<p>AUX_15 Schlusslicht hinten +0 = Helligkeitsstufe 1 +32 = Helligkeitsstufe 2 +64 = Helligkeitsstufe 3 +96 = Helligkeitsstufe 4 +128 = Helligkeitsstufe 5 +160 = Helligkeitsstufe 6 +192 = Helligkeitsstufe 7 +224 = Helligkeitsstufe 8 >> feste Zuordnung zur Funktionstaste F0/FL</p>
CV48	224	AUX_16 Schlusslicht vorne

CV49	39	AUX 17 Kupplung I +0 = F0/FL +1 = F1 +2 = F2 +3 = F3 +4 = F4 +5 = F5 +6 = F6 +7 = F7 +32 = Zeitintervall 1sec. +64 = Zeitintervall 2sec. +96 = Zeitintervall 3sec. +128 = Zeitintervall 4sec. +160 = Zeitintervall 5sec. +192 = Zeitintervall 6sec. +224 = Zeitintervall 7sec.
CV50	231	AUX 18 Kupplung II
CV51	1	Effekte von AUX_1 +1 : normales Licht +2 : Neonlicht (Startflackern) +4 : defekte Lampe (flackert in zeitlichen Abständen) +8 : Nachtmodus (nur AUX 9-14 und in Verbindung mit AUX 2-7) +16: Kupplung +32: Ausgang schaltet zufällig um, bei Nachtmodus nur 1x (mit 1/2/4 und 8 kombinierbar) +64: AUX rückwärts aus +128: AUX vorwärts aus
CV52	2	Effekte von AUX_2
CV53	2	Effekte von AUX_3
CV54	2	Effekte von AUX_4
CV55	2	Effekte von AUX_5
CV56	2	Effekte von AUX_6
CV57	2	Effekte von AUX_7
CV58	1	Effekte von AUX_8
CV59	2	Effekte von AUX_9 AUX_9 im Nachtmodus AUX_2 zugeordnet
CV60	2	Effekte von AUX_10 AUX_10 im Nachtmodus AUX_3 zugeordnet
CV61	2	Effekte von AUX_11 AUX_11 im Nachtmodus AUX_4 zugeordnet
CV62	2	Effekte von AUX_12 AUX_12im Nachtmodus AUX_5 zugeordnet
CV63	2	Effekte von AUX_13 AUX_13 im Nachtmodus AUX_6 zugeordnet
CV64	2	Effekte von AUX_14 AUX_14 im Nachtmodus AUX_7 zugeordnet
CV65	65	Schlusslicht hinten
CV66	67	Schlusslicht vorne
CV67	16	Kupplung 1
CV68	16	Kupplung 2

Für Verbesserungsvorschläge und Hinweise auf Fehler bin ich sehr dankbar.

Auf die Bauanleitung bzw. irgendwelcher Software gibt es keine Haftung für irgendwelche Schäden oder Funktionsgarantie. Ich hafter nicht für Schäden, die der Anwender oder Dritte durch die Verwendung der Software oder Hardware verursachen oder erleiden. In keinem Fall hafter ich für entgangenen Umsatz oder Gewinn oder sonstige Vermögensschäden die bei der Verwendung oder durch die Verwendung dieser Programme oder Anleitungen entstehen können.

Bei Rückfragen steht Ihnen unser Support-Forum gerne zur Verfügung!

Kontakt:

fichtelbahn.de

Christoph Schörner

Ahornstraße 7

D-91245 Simmelsdorf

support@fichtelbahn.de



© 2012 Fichtelbahn

Alle Rechte, insbesondere das Recht der Vervielfältigung und Verbreitung sowie der Übersetzung vorbehalten.
Vervielfältigungen und Reproduktionen in jeglicher Form bedürfen der schriftlichen Genehmigung durch Fichtelbahn.
Technische Änderungen vorbehalten.