# DCC-Dekoder

**In Bearbeitung**

## DCC-Funktions-Dekoder mit ATtiny85

### Funktions-Dekoder mit Digispark-Board

Hier wurde ein Funktionsdekoder mit einem ATtiny85-Digispark-Board vorgestellt. Ich habe dafür die Hardware ergänzt, damit auch CVs in Wagen, die mit diesem Decoder ausgerüstet sind, auch auf dem Programmiergleis programmiert und zurückgelesen werden können. Der damit ausgestattete Dekoder hat dann nur noch 3 Ausgänge.

Ein ergänzter Arduino-Sketch für die 3-Port-Variante mit ACK ist dort abgelegt, ebenso ein Sketch für 4 Ports ohne CV-Lesemöglichkeit. Das Programmieren kann aber „blind“ auf dem Programmiergleis erfolgen.
**Ich empfehle in jedem Fall das dort erwähnte Upgrade auf die 300ms-Bootzeit.** Oder es wird mit der ISP-Programmierung komplett auf einen Bootloader verzichtet.

Nach ersten positiven Versuchen mit einem fliegendem Aufbau habe ich eine kleine Zusatzplatine entworfen, die rückseitig mit Stiften aufgelötet wird. Hier der überarbeitete Entwurf mit der zusätzlichen Diode.

Schaltplan
Bestückungsplan
Stückliste
Bestellmöglichkeit der Leiterplatte für Eigenbau bei Aisler
Eine kommerzielle Nutzung ist untersagt.

Hier ein Einbaubeispiel der ersten Version der Platine im Pwgs/Daa-Wagen. Nachträglich musste eine zusätzliche Diode eingefügt werden, damit auch mit dem Stützkondensator ein CV-Schreiben und -Lesen mit dem ACK-Signal zuverlässig funktionieren.

Die Software-Variante mit den 3 Ports wird von mir weiterentwickelt. Da ich die USB-Funktion im endgültigen Einbau nicht benötige, verwende ich das Board ohne den Micronucleus-Bootloader sondern mit ISP-Programmierung ohne Bootloader mit dem STK500-Programmer.

Damit hatte ich den notwendigen Programmspeicherplatz für zusätzliche Funktionalität:

- Funktionsmapping für zwei Ports.
- Dimmung der Ports

Aktuelle Projektdaten sind auf Github abgelegt.

Folgende Funktionen sind aktuell konfiguriert:

- F0 Taste schaltet richtungsabhängig beide Schluss-LEDs an PB0 (LV) und PB4 (LR)
- F1 Taste schaltet Wagenbeleuchtung an PB1 (AUX1), die beiden Onboard-LEDs des Digispark-Boards wurden deaktiviert.

Wenn auf das Lesen der Konfigurationsvariablen verzichtet werden kann oder ein optionaler Schalter/Jumper möglich ist, dann kann auch Port PB3 als Ausgang genutzt werden. Mit PB3 (AUX2) ist aber kein Dimmen möglich, da an diesem Port kein PWM unterstützt wird.

Schaltplan

Diese Variante ist auch bei Github beschrieben. Eine Zusatzplatine für das Digispark-Board folgt.

---

### Funktions-Dekoder auf Lichtleiste

Auf der Basis der oben genannten Hard- und Software wurde eine Lichtleiste mit einem ATtiny85 aufgebaut. Diese benötigt ebenfalls keinen Bootloader, wie das Digispark-Board. Die Programierung erfolgt über einen 6-poligen ISP-Stecker mit dem STK500-Programmer. Es wird der gleiche Sketch wie oben verwendet. Es werden drei Ausgänge genutzt und das ACK-Signal wird verwendet.

Die Lichtleiste wird in SCHICHT-Rekowagen eingesetzt. Dort sind auch die Leiterplattendaten abgelegt.

---

## DCC-Zubehör-Dekoder mit ATtiny85

### Entwicklungsboard

Nachdem ich den Funktionsdekoder auf Basis des ATtiny85 für die Beleuchtung meiner Wagen erfolgreich in Betrieb genommen hatte (s.o.), habe ich auf gleicher Basis die Entwicklung eines Sketches für einen Zubehördekoders begonnen, um damit Signale zu steuern.
Als ersten Ansatz dafür war ein Arduino-Beispielsketch der NmraDcc-Bibliothek von MRRWA. Dieser wurde dann um die Funktionen zur Ansteuerung der Ausgänge ausgebaut.

Mit einer Zusatzplatine wird aus dem DCC-Signal die Spanung für das ATtiny85-Board generiert. An die Ausgänge des ATtiny sind Transistoren angeschlossen, um Verbraucher mit der gleichgerichteten DCC_Spannung versorgen zu können. Es kann optional ein Stützkondensator angeschlossen werden, was für stationäre Verbraucher aber eher nicht notwendig ist. Mit einem weiteren Anschluss kann auch ein Lastwiderstand zugeschaltet werden, mit dem einen Ausgang zur Erzeugung des DCC-ACK-Signal genutzt wird. Damit ist dann auch Lesen der CV-Werte mit dem Programmiergleis-Anschluss möglich.

Der Dekoder hat 4 Ausgänge (Ports), die mit DCC-Befehlen aktiviert und deaktiviert werden können. Mit der Adresse+1 kann der Port als blinkend eingeschaltet werden. Die Blinkperiode (für alle Ausgänge gleich) ist konfigurierbar von 0,25 … 3,75s.
PORT1 und PORT2 können als alternierend verbunden werden (z.B. Weichen, Signale) und es besteht auch die Möglichkeit, bei PORT1 und PORT2 einen Impuls definierter Länge (0,25 … 8s) zu erzeugen, z.B. bei Weichen ohne Endabschaltung.

CV1/CV9 = Dekoder-Adresse (Modul-Adresse nach NMRA mit 4 Ports und zwei Gates)
CV1 = 6 bit LSB

```
x x x x x x x x
- - +-+-+-+-+-+ - - - - - LSB 0 … 63
```

CV9 = 3 Bit MSB

```
x x x x x x x x
- - - - - +-+-+ - - - - - MSB (0 … 7) * 64
```

CV3 und CV4 Time On Function für PORT1 und PORT2

```
x x x x x x x x
- - - +-+-+-+-+ - - - - - 5 bit Impulslänge für PORT1 und PORT2, number * 256 ms (256 ms
… 7.93 s)
```

CV33 Configuration PORT1-PORT2-Behavior
```
x x x x x x x x
- - - - - - +-+ - - - - - „0-0“ = 4 unabhängige Ports
- - - - - - +-+ - - - - - „0-1“ = PORT1/PORT2 als alternierende Ausgänge mit PORT1-
On/Off-Kommando
- - - - - - +-+ - - - - - „1-0“ = PORT1/PORT2 als alternierende Ausgänge mit PORT1- bzw.
PORT2-On-Kommando
```

CV34 Blink Periode
```
x x x x x x x x
- - - - +-+-+-+ - - - - - 4 bit Blinkperiode in s (0.25 … 3.75 s)
```

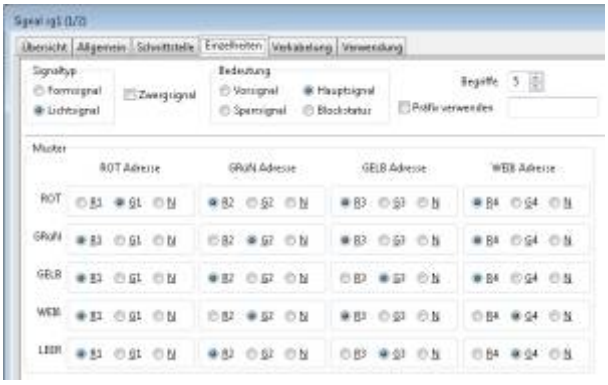
Für die Adressierung gibt es unterschiedliche Varianten. Ich habe mit Rocrail und der DCC-Commandstation von DCC-Ex die MADA-Adressierung ohne Probleme verwenden können. Andere Adressierungsarten kann ich mit dieser Commandstation nicht testen.
An einer Z21 sollte es mit der PADA-Adressierung funktionieren, wahrscheinlich jedoch mit einem Offset von +4.

Im Video ein Beispiel für eine Signalsteuerung durch Rocrail und dem LocoIO-Keypad.

Für meine Modellbahn habe ich erstmal folgende Signalbilder mit statischer Anzeige vorgesehen:

- Hl 1 - ein grünes Licht - Fahrt mit Strecken-Höchstgeschwindigkeit
- Hl 3a - ein gelbes Licht unten, darüber ein grünes Licht - Fahrt mit 40 km/h, dann mit Strecken-Höchstgeschwindigkeit
- Hl 10 - ein gelbes Licht oben - „Halt“ erwarten
- Hl 12a - ein gelbes Licht unten, darüber ein gelbes Licht - Geschwindigkeit auf 40 km/h ermäßigen, „Halt“ erwarten
- Hl 13 - ein rotes Licht - „Halt“

Die Signalbilder sind im Rocrail mit den Einstellungen für Mustern für die 4 Ausgänge erstellt worden. Es sind 5 Muster möglich, wenn man Muster „Leer“ auch belegt.



Mit Aktionen können auch weitere Signalbilder erzeugt werden, wie Hl 4 und Hl 7. Ich habe mal testweise diese zwei erzeugt:

- Hl 6a - ein gelbes Licht, darüber ein grünes Blinklicht
- Hl 9a - ein gelbes Licht, darüber ein gelbes Blinklicht

attiny_signal_k.mp4