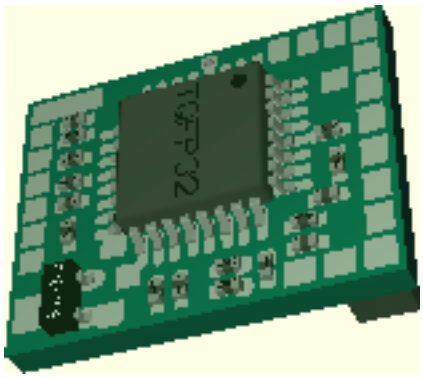
**Bau und Programmieranleitung für den OpenSourceCarDecoder**

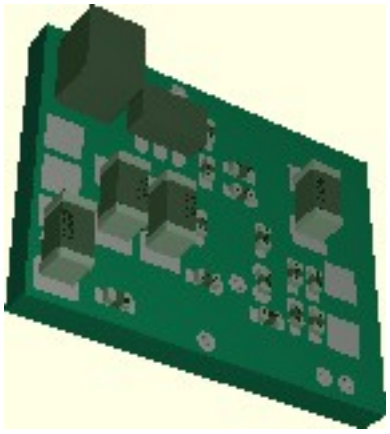
Vorbemerkungen

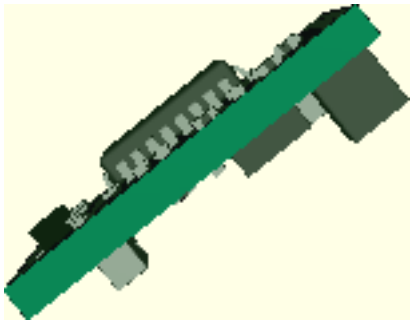
Die folgende Anleitung beschreibt den Aufbau und die Inbetriebnahme des „OpenSourceCarDecoder“. Ich möchte ausdrücklich darauf hinweisen, das es sich hierbei nicht um ein kommerzielles Produkt handelt, sondern um eine Anregung für den interessierten Modellbauer oder Elektroniker. Ich habe die Schaltung und die Software nach besten Gewissen entworfen und entwickelt, kann aber Fehler nicht ausschließen und übernehme auch keinerlei Haftung für Schäden, die durch die Nutzung meines Decoders oder der Software entstehen. Jeder darf den Decoder nachbauen und die Software verwenden oder weiterentwickeln (bitte mit Quellenangabe). Eine kurze Mail Eurer Tests, mit Fehlern, Anregungen, Tipps oder Weiterentwickelungen würde mich sehr freuen. Eine kommerzielle Nutzung der Software erfordert allerdings zwingend mein Einverständnis! Alle im folgenden genanten Namen und Marken sind Eigentum ihrer jeweiligen Unternehmen und werden hiermit von mir anerkannt.

Birkenwerder im Februar 2013

Toralf Wilhelm
mail@toralfwilhelm.de



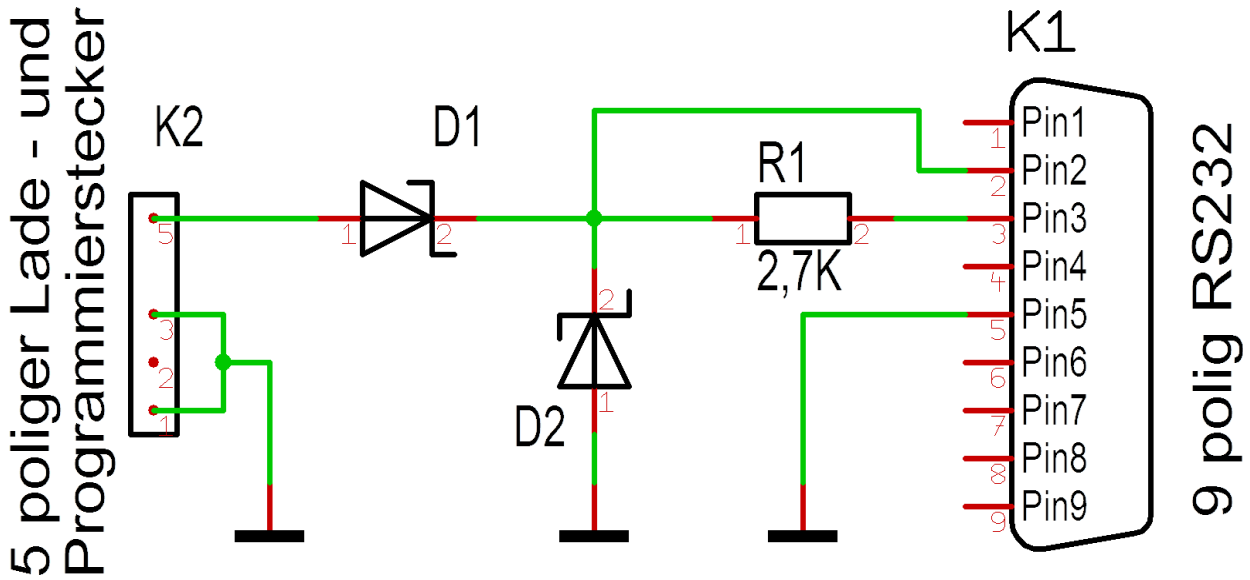




# Inhalt

**Vorbereitungen und Voraussetzungen**

Für den Eigenbau der Car Decoder ist Erfahrung im Umgang mit sehr kleinen SMD Bauteilen nötig! Wer diese nicht hat, dem rate ich dringend davon ab, seine Decoder selbst aufzubauen. Die Bauteile haben äußere Abmessungen von 0,5 x 1 mm (Bauform 0402). Um das erfolgreich aufzubauen sind ruhige Hände, gute Augen, viel Löterfahrung, eine sehr feine Lötspitze, Lötzinn maximal 0,5 mm stark, eine SMD Pinzette, Flussmittel und etwas Entlötlitze nötig. Für alle die hier und jetzt aufgeben müssen, wird es bei Projektfortschritt fertig bestückte Platinen geben, so das dann jeder mit normalen Modellbaufähigkeiten an diesem Projekt mitmachen kann.
Zum programmieren ist ein AVR ISP Programmer und die passende Software dafür nötig. Ich verwende Atmels AVR-Studio in der Version 4 und einen „usbprog“ von Benedikt Sauter. Es gibt hierfür noch viele Alternativen, eine gute Anlaufstelle für Informationen zum AVR ist: http://www.mikrocontroller.net/articles/AVR-Tutorial . Da dieses Projekt sich ständig weiterentwickeln soll, ein Open Source Projekt ist und somit auch schnell und unkompliziert auf Anregungen und Wünsche der Nutzer eingegangen werden kann, wird sich die Software öfter einmal ändern. Abgesehen davon, bin ich kein professioneller Programmierer und werde mit Sicherheit den ein oder anderen Fehler fabrizieren. Aus diesem Grund habe ich auch bei diesem Projekt wieder den Betrieb mit einem Bootloader zum einfachen wechseln der Decoder Software vorgesehen. Hierfür ist noch ein 1-wire Kabel nötig, welches sich aber leicht selbst anfertigen lässt. Es werden zwei beliebige Schottkydioden und ein 2,7K Widerstand benötigt. K2 ist eine polige Stiftleiste, bei der Pin 4 entfernt wird (abkneifen). Die Pins 1 bis 3 sind hier für die original Faller Ladebuchse gedacht, Pin 4 gibt es nicht und Pin 5 ist der Programmierpin. Dieser muss natürlich an den Fahrzeugen auch noch angebracht werden. Durch den fehlenden Pin 4 ist der Programmierstecker dann auch verpolsicher. Der Betrieb an USB – seriell Adaptern ist auch hier wieder möglich.




Die passende Software AVRootloader.exe stellt Hagen aus dem Mikrocontrollerboard zur Verfügung (ist im Softwarepaket zum Car-Decoder enthalten). Zur Inbetriebnahme ist ein Multimeter und ein IR - Booster nötig. Das aktuelle Softwarepaket findet Ihr immer auf „opcar.de“ unter folgenden Link: http://www.opcar.de/download/index.html .

**Bauteile**

Dank Oliver gibt es eine Excel Bauteilliste, in der Links zu drei Elektronikversendern enthalten sind. Sie ist auch unter den Downloads auf „opcar.de“ zu finden. Hier noch einmal alle Positionen für die 18x14mm große TQFP Version:

| Pos | Name | Stück | Wert | Gehäuse |
|---|---|---|---|---|
| 1 | C1, C9, C10 | 3 | 4,7µF | 1206 |
| 2 | C2 | 1 | 10uF | 1206 |
| 3 | C3 - C6 | 4 | 100nf | 0402 |
| 4 | C7 | 1 | 100pF | 0402 |
| 5 | C8 | 1 | 220pF | 0402 |
| 6 | IC1 | 1 | MCP1640D | SOT23/6 |
| 7 | IC2 | 1 | ATMEGA8 | TQFP32(7X7) |
| 8 | L1 | 1 | LQH3C4,7uH | 1210 |
| 9 | R1 | 1 | 120k | 0402 |
| 10 | R2 | 1 | 47k | 0402 |
| 11 | R3, R6, R15 | 3 | 10k | 0402 |
| 12 | R4 | 1 | 47 | 0402 |
| 13 | R5, R11, R21, R22, R23, R24, R25,R27 | 8 | 220 | 0402 |
| 14 | R7 | 1 | 100k | 0402 |
| 15 | R8, R9 | 2 | 3k9 | 0402 |
| 15 | R10 | 1 | 33 | 0402 |
| 16 | R12, R18, R19 | 3 | 470 | 0402 |
| 17 | R14 | 1 | 33k | 0402 |
| 18 | R20,R26 | 2 | 1k | 0402 |
| 19 | T1 | 1 | IRLML2803 | SOT23/3 |
| 20 | IR-Empfänger | 1 | 455kHz | ca.8x6x6 |
| 21 | D1, D2 SMD IR Dioden | 2 | ELIR11-21C | 1206 |
| 22 | T2, T3 SMD Fototransistoren | 2 | ELPT11-21C | 1206 |

Die Positionen 20 – 22 sind externe Fotobauteile, welche nicht auf der Platine bestückt werden, sondern direkt am Fahrzeug zu montieren sind. Die Platinen sind auf Anfrage bei mir, für momentan (Stand 02/2013) 1,50€/Stück erhältlich. Es gibt sie aktuell in zwei unterschiedlichen Größen (18x14 mm und 14x10 mm). Sie Unterscheiden sich nur in der Bauform des IC2 ATMEGA8. Auf der kleinen Platine ist die QFN Bauform des ATMEGA vorgesehen, welche schwerer zu bekommen ist, teurer ist und sich Bauform bedingt schlechter von Hand löten lässt. Alle anderen Bauteile sind identisch.

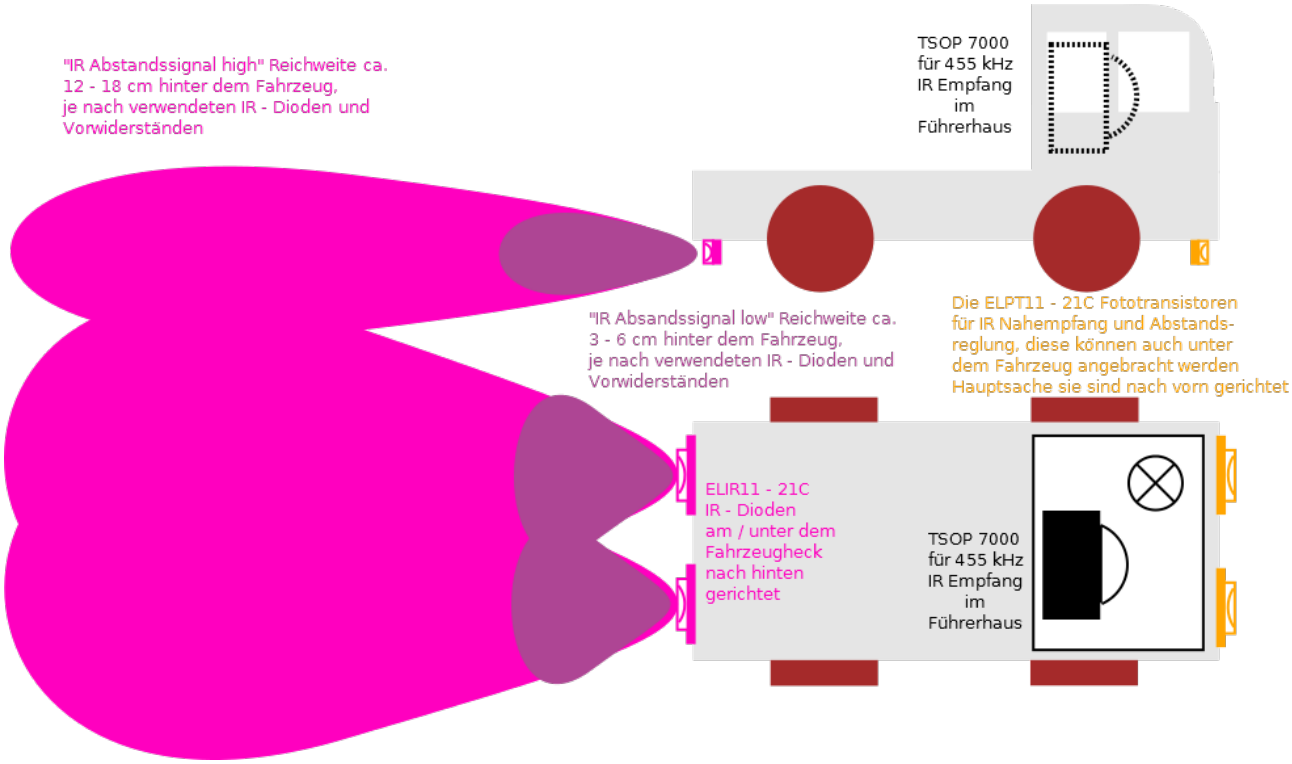
**Externe Fotobauteile**

Die Fotobauteile zur Abstandssteuerung der Positionen 21 und 22 sind in der SMD Bauform 1206 und müssen mit freier Sicht nach vorn bzw. hinten am Fahrzeug montiert werden. Das kann bei kleinen Fahrzeugen (Spur N bzw. auch PKWs in H0) nicht immer „optisch schön“ möglich sein. Dann kann alternativ, bei etwas Reichweitenverlust, auf Bauteile in der Bauform 0805 zurückgegriffen werden:

| | | | | |
|---|---|---|---|---|
| 21 | D1, D2 SMD IR Dioden | 2 | ELIR17-21C | 0805 |
| 22 | T2, T3 SMD Fototransistoren | 2 | KP2012P3C | 0805 |

Wobei R27 dann mit 100R und R26 mit 680R abweichend zu bestücken sind. Dies erhöht den Strom durch die IR Dioden, was den Reichweitenverlust etwas ausgleicht, allerdings auf Kosten des

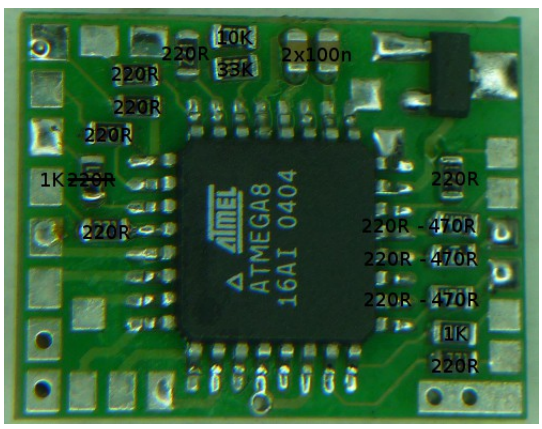
Stromverbrauchs. Folgend ein Bild, zur Montage der Fotobauteile am Fahrzeug:



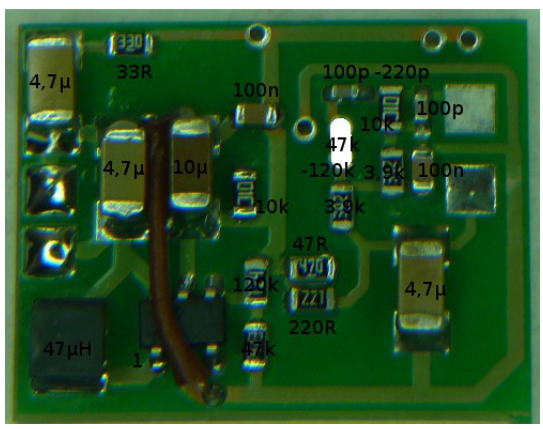

**Bestückung**

Je nach eigenen Fähigkeiten kann die Reihenfolge unterschiedlich sein. Elektrisch gesehen, macht es Sinn zuerst den Spannungswandler zu bestücken, ihn dann zu testen und danach den Rest der Bauteile zu bestücken. Wer etwas weniger Erfahrung hat, kann aber auch erst mit dem Mikrocontroller anfangen, meine Empfehlung ist, die Lötpads auf der Platine dazu leicht zu verzinnen, so das auf jedem Pad ein kleiner Zinnhügel ist, dann den Mikrocontroller exakt positionieren und zuerst diagonal je ein Pin anzulöten. Jetzt noch einmal die exakte Position über den Lötpads überprüfen, Notfalls noch einmal korrigieren und dann einfach Pin für Pin durch Erwärmen ohne weiteres Lötzinn auflöten. Sollten einmal zwei Pads verbunden sein, bekommt man das überschüssige Lötzinn mit Flussmittel und Kuferlötsauglitze ( normale feine Litzen geht auch) wieder entfernt. Alle Pins noch einmal überprüfen! Danach werden die restlichen SMD Bauteile bestückt. Folgend zwei Bilder von der Platine mit den bestückten Bauteilen.

Oberseite:




Unterseite:




Auf der Unterseite dieser ersten Platinenversion ist eine Drahtbrücke zu sehen. Diese muss hier

noch nachträglich von der GND Seite C1/C2 nach Pin2 vom IC1 eingebaut werden und verhindert ein schwingen des IC1. Das Design der Platine ist an diesem Punkt noch Fehlerhaft und wird bei Version 2 dahingehend noch einmal verbessert. Die LED Vorwiderstände können und müssen an die verwendeten LEDs angepasst werden. Als Richtwert können die Werte aus der Bauteilliste verwendet werden.

### Inbetriebnahme

Wenn vorhanden, macht sich ein ein regelbares Netzteil sehr gut beim überprüfen der Decoderhardware. Wenn nicht vorhanden, können auch zwei 1,5V Batterien verwendet werden. Zum Schutz bei Fehlerhafter Hardware sollte dann allerdings ein Widerstand mit ca. 10R in die Pluszuleitung zur Decoderplatine eingefügt werden. Mit Netzteil bitte auch so ca. 3V einstellen und den Strom auf ca. 200mA begrenzen.
Jetzt den Decoder an die 3V anschließen. Als erstes die +UB (z.B. über C2) messen. Wenn alles richtig bestückt ist sollten dort ca. 4,3V anstehen. Wenn nicht, müssen die Bauteile C1, L1, IC1, R1, R2 und C2 bzw. die +UB auf Kurzschlüsse (Lötbrücken) überprüft werden.
Wenn die Spannung in Ordnung ist, kann mit der Erstprogrammierung weiter gemacht werden.

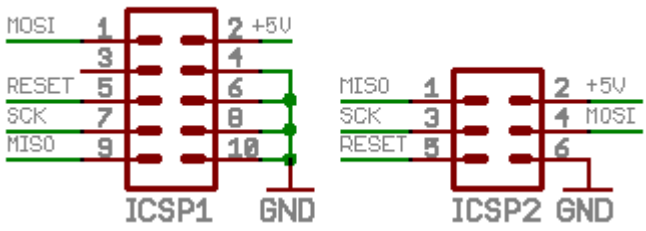
### Programmierung

#### Bootloader

Die Software für den Car 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 Car Decoder einfach und unkompliziert über eine 2 Draht Verbindung (Masse + Signal) auszutauschen (z.B. über einen weiteren Pin an der Ladebuchse). So wird es dann jeder Zeit möglich sein, den Decoder eingebaut in einem Fahrzeug mit einer neuen Software auszustatten und das ohne speziellen AVR - Programmer. Die eigentliche CarDecoderSoftware besteht aus zwei Teilen, dem Programm und den CV – Daten. Das Programm (m8_car.hex) ist für alle Decoderversionen mit der selben Hardware identisch. Die passende Variablendatei (m8_car.eep) enthält die CV des Decoders. Diese werden im EEPROM des AVRs abgelegt und sind so im Betrieb veränderbar. Sie ist also bei jedem Decoder unterschiedlich. Es gibt also drei verschiedene Dateien für den Decoder:

| | |
|---|---|
| AVRootloader.hex | den Bootloader |
| m8_car.hex | das Decoderprogramm |
| m8_car.eep | die CV Variablen des Decoder |

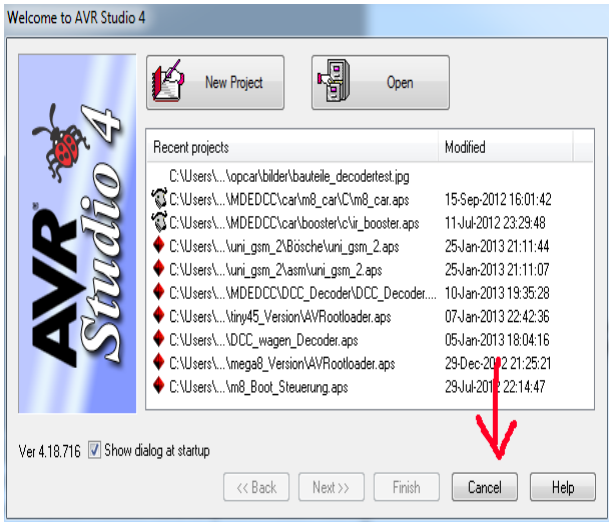
In dem AVR muss mindestens der Bootloader mit Hilfe eines AVR Programmers geschrieben werden. Hier möchte ich jetzt erläutern, wie mit Hilfe des AVR Studios von Atmel und einem USBPROG (original AVR mkII kompatibel) der Bootloader in den AVR geschrieben wird. Ich habe dafür ein ISP Kabel (In System Programmer Kabel) mit folgender Belegung (siehe: http://www.mikrocontroller.net/articles/AVR_In_System_Programmer )



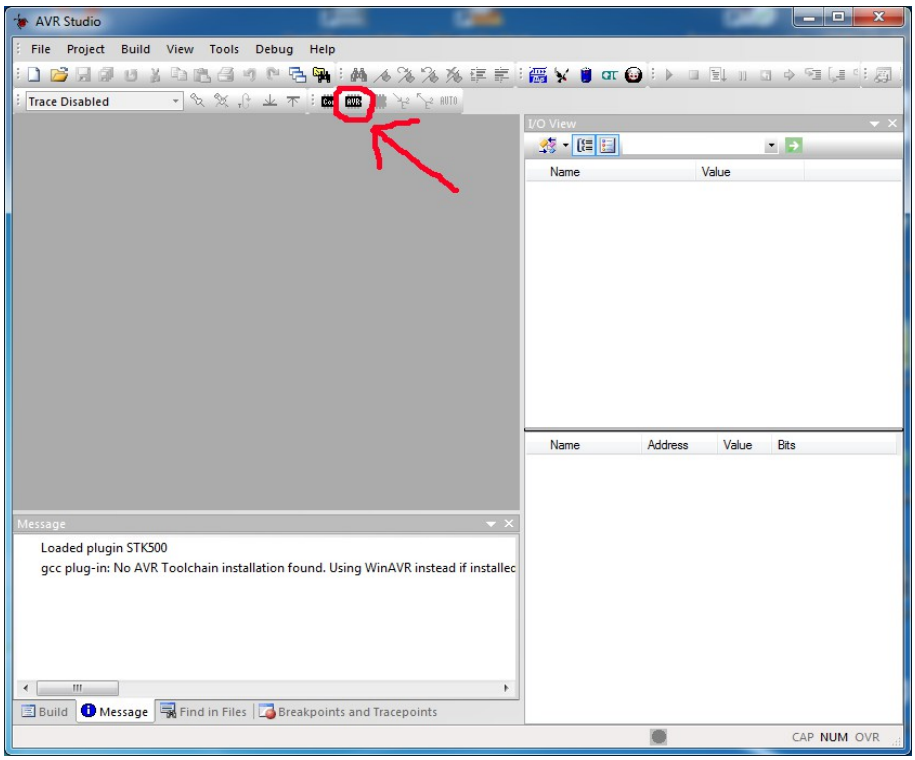
Es müssen folgende sechs Verbindungen mit dem Decoder hergestellt werden (siehe Anschlussbelegung vom Decoder):

| | | |
|---|---|---|
| ISP / MOSI | → | Decoder / MOSI |
| ISP / MISO | → | Decoder / MISO |
| ISP / SCK | → | Decoder / SCK |
| ISP / RESET | → | Decoder / RESET |
| ISP / GND | → | Decoder / GND TSOP |
| ISP / +5V | → | Decoder / +UB TSOP |

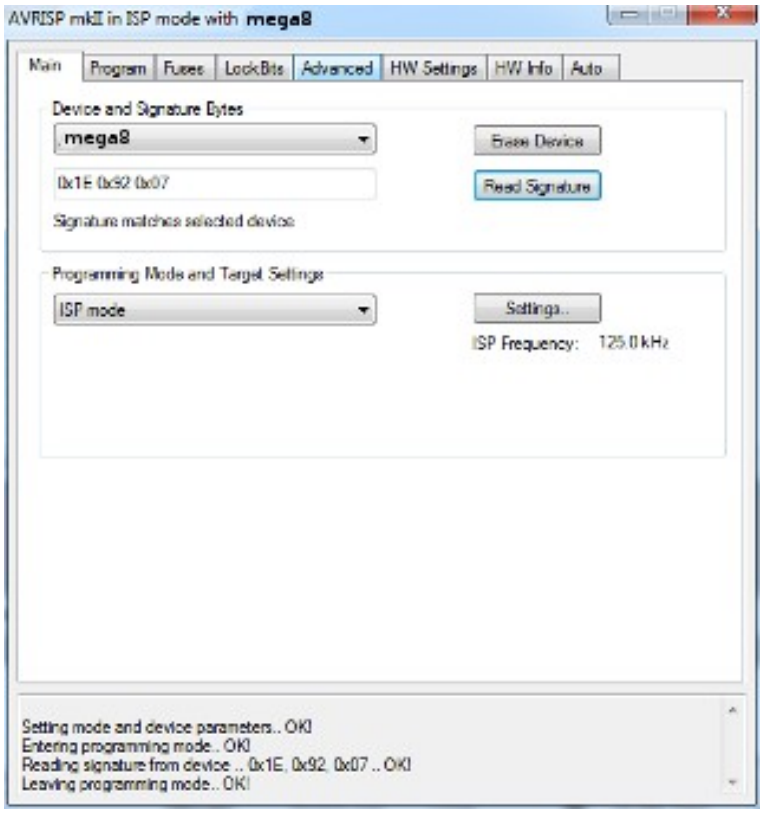
Das Atmel AVR Studio starten (hier Version 4). Im „Welcome“ Bildschirm muss kein Projekt geöffnet werden, einfach mit „Cancel“ abbrechen.




Man erhält dann ein leeres AVR Studio Fenster nun den Programmerdialog öffnen.



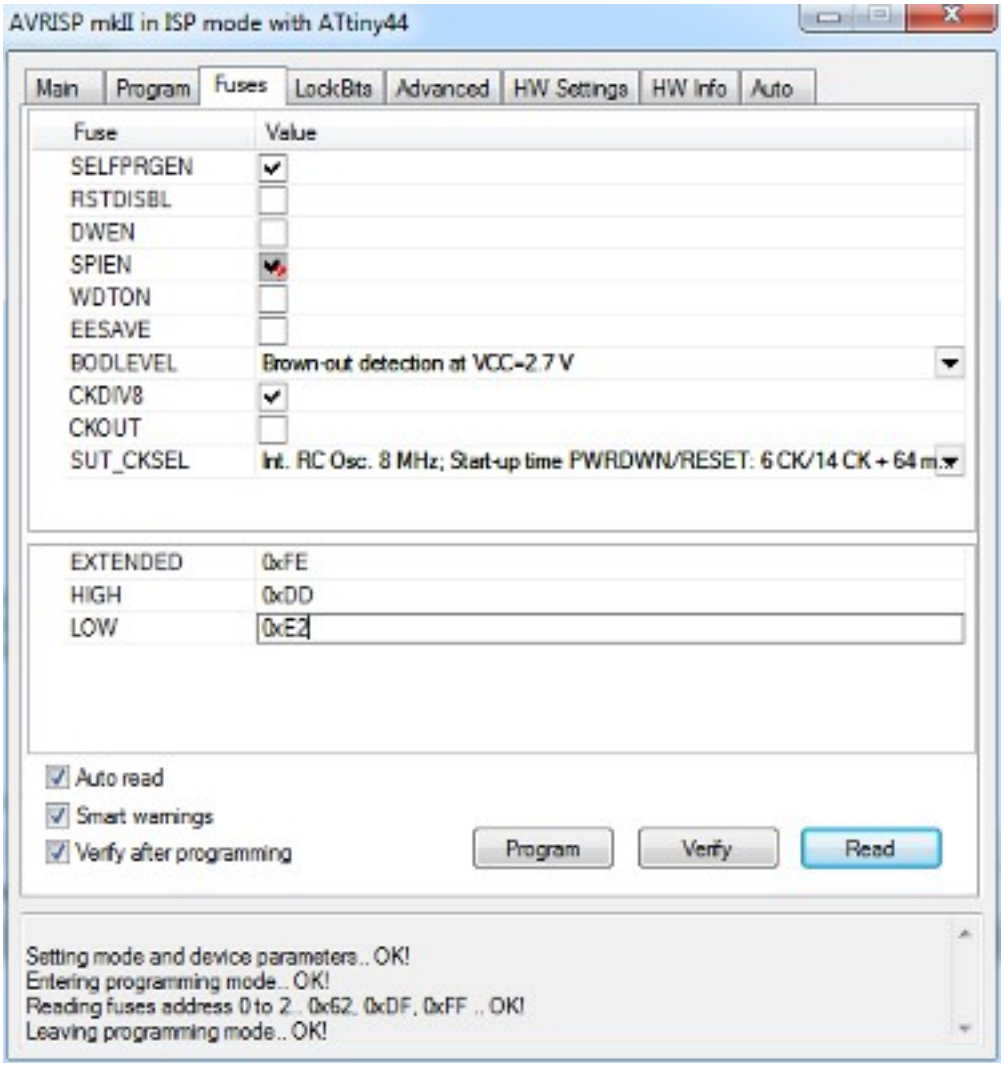
Im Programmerdialog den ***„Main“*** Reiter auswählen.




Zuerst bei ***„Device and SignatureBytes“*** den verbauten AVR Typen auswählen (wenn mega8 bestückt ist, mega8 auswählen beim mega88 diesen auswählen).
Unter ***„Settings“*** 125kHz auswählen und abspeichern. Dann ***„Read Signature“*** auswählen. Es sollte jetzt eine gültige Signatur aus dem AVR ausgelesen werden, es darf an dieser Stelle keine Fehlermeldung auftreten. Wenn es eine gibt, folgendes überprüfen:

- Treiber für den AVR Programmer installiert?
- ISP Verbindung alle sechs Leitungen richtig angeschlossen?
- Decoderplatine mit 3-4V versorgt?
- +UB TSOP auf der Decoderplatine muss ca. 4,3V haben

Wenn die Signatur erfolgreich ausgelesen wurde, zum Reiter „***Fuses“*** wechseln.
(Achtung die folgende Ansicht stimmt nicht mit dem CarDecoder mega8 überein!)




Wir tragen jetzt für die Fuse folgende Werte ein:

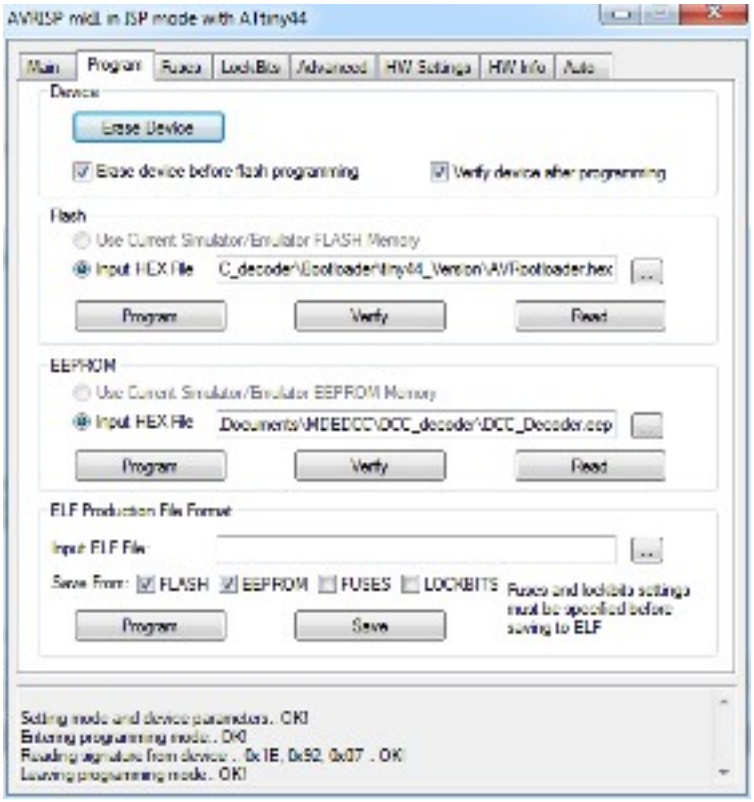
mega8 Bootloader Version:

HIGH: 0xDC
LOW: 0x84

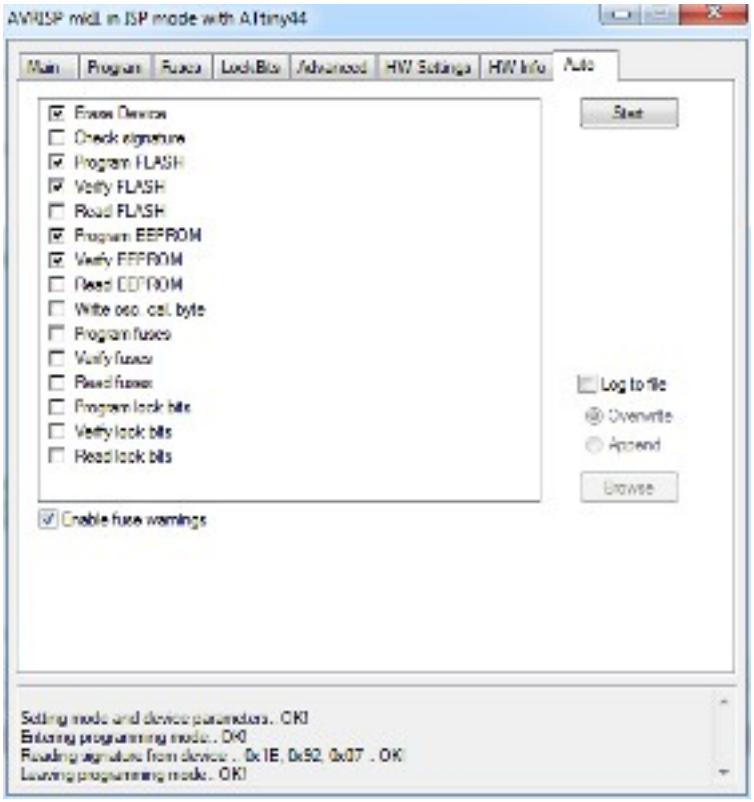
und speichern diese mit ***„Program“*** ab.

Danach gehen wir zurück auf die ***„Main“*** Seite und stellen die ISP Frequenz auf 1MHz um. Das ist notwendig, da wir durch das ändern der Fuses den internen Takt des AVR von 1Mhz auf 8Mhz umgeschaltet haben. Zur Kontrolle wieder die Device Signatur auslesen. Es darf wieder keine Fehlermeldung geben.

Nun in den ***„Program“*** Reiter wechseln und bei ***„Flash“*** die passende AVRooloader.hex auswählen (für mega8 bzw. mega88). Bei ***„EEPROM“*** und ***„ELF..“*** muss nichts eingegangen werden.
(auch hier passt das Bild nicht zum mega8 CarDecoder, es dient nur zur Symbolansicht)




Zum Schluss in den ***„Auto“*** Reiter wechseln , es muss mindestens bei ***„Erase Device“***, ***„Program Flash“*** und ***„Verify Flash“*** ein Haken sein, der Rest braucht nicht aktiv sein. Dann auf ***„Start“*** klicken und den Bootloader in den AVR schreiben.




Damit ist der erste Teil der Decoderprogrammierung fertig. Es können jetzt alle ISP Leitungen wieder vom Decoder entfernt werden.

**Firmware und Konfigurationsvariablen**

Im zweiten Schritt, wird mit Hilfe des Bootloaders und einer 1-wire (1-Draht) Verbindung die OpenCarDecoder Firmware und die Konfigurationsvariablen (CV Werte) in den Decoder geschrieben. Zuerst einmal etwas Grundsätzliches zur Firmware bzw. zum Firmware wechseln. Das OpenCarSystem nutzt zur Übertragung der Steuerungsbefehle eine normale serielle RS232 Datenübertragung (IR moduliert). Bei einer RS232 Datenübertragung ist es wichtig, das Sender und Empfänger gleich schnell arbeiten. Es ist nur eine minimale Abweichung der Übertragungsfrequenz möglich, ohne die Übertragung gestört ist. Da Hauptsächlich aus Platzgründen der Decoder den internen RC Oszillator des AVR verwendet und dieser nicht sehr genau und Betriebs-spannungsabhängig ist, muss jeder AVR einzeln abgeglichen werden. Dies macht die OpenCarDecoder Firmware automatisch. Sie verwendet dafür das modulierte IR Signal des IR – Booster. Der Booster arbeitet Quarzstabil, so das ein abgeglichener Decoder auch an anderen Boostern betrieben werden kann. Wichtig ist nur, das dieser automatische Abgleich unmittelbar nach dem aufspielen der Firmware auf den Decoder stattfindet. Aus diesem Grund muss beim Firmware aufspielen ein IR – Booster im Betrieb sein (es muss kein DCC Signal anliegen) und der TSOP7000 muss angeschlossen sein. Der Decoder zeigt diesen Abgleich durch einschalten beider Blinklichter an. Diese gehen nach ca. einer Sekunde nacheinander aus. Danach ist der Decoder abgeglichen und betriebsbereit.
Zum 1-wire programmieren müssen folgende Verbindungen hergestellt werden:

| **5 poliger Lade und Programmieranschluss** | | **OpenCarDecoder** |
|---|---|---|
| K2 Pin5 / 1-wire | → | Prog |
| K2 Pin1 oder 3 / GND | → | GND TSOP |

Der Decoder muss wieder mit 3 - 4V versorgt werden. Wir starten jetzt aus dem Ordner: /AVRooloader/Windows die AVRootloader.exe :

und wählen den ***Port*** und die ***Baudrate*** (9600) aus. Bei den ***Programmingfiles*** wählen wir unser Decodersoftware (m8_car.hex) und unsere CV Daten (m8_car.eep) aus.
Danach verbinden wir uns mit unserem AVR → ***Connect to device***. Wir sollten dann folgendes Bild erhalten (im Fensterrahmen muss „connected“ stehen):


Wenn dem so ist, einmal auf ***„Program“*** klicken und kurz warten bis die Programmierung fertig ist, das sieht dann so aus:

Jetzt ist der Decoder fertig programmiert. Bevor die Verbindung zum Decoder getrennt wird (***„Disconnected device“***) überprüfen ob er IR Booster eingeschaltet ist (zum automatischem Abgleich). Wenn ja, dann ***„Disconnected device“.*** Am Decoder sollten jetzt alle Blinklichter einschalten und nach ca. einer Sekunde nacheinander wieder ausgehen. Fertig, der Decoder ist nun betriebsbereit.

An dieser Stelle sei bemerkt, der Firmware Update des Decoders läuft auch später genau so ab, wie hier beschrieben. So ist es problemlos möglich diese unkompliziert auszutauschen. Dazu muss in die CV55 eine 1 programmiert werden und der Decoder neugestartet werden. Er springt dann (bei CV55 = 1) direkt den Bootloader an und wartet auf eine neue Firmware.

**Videoanleitungen**

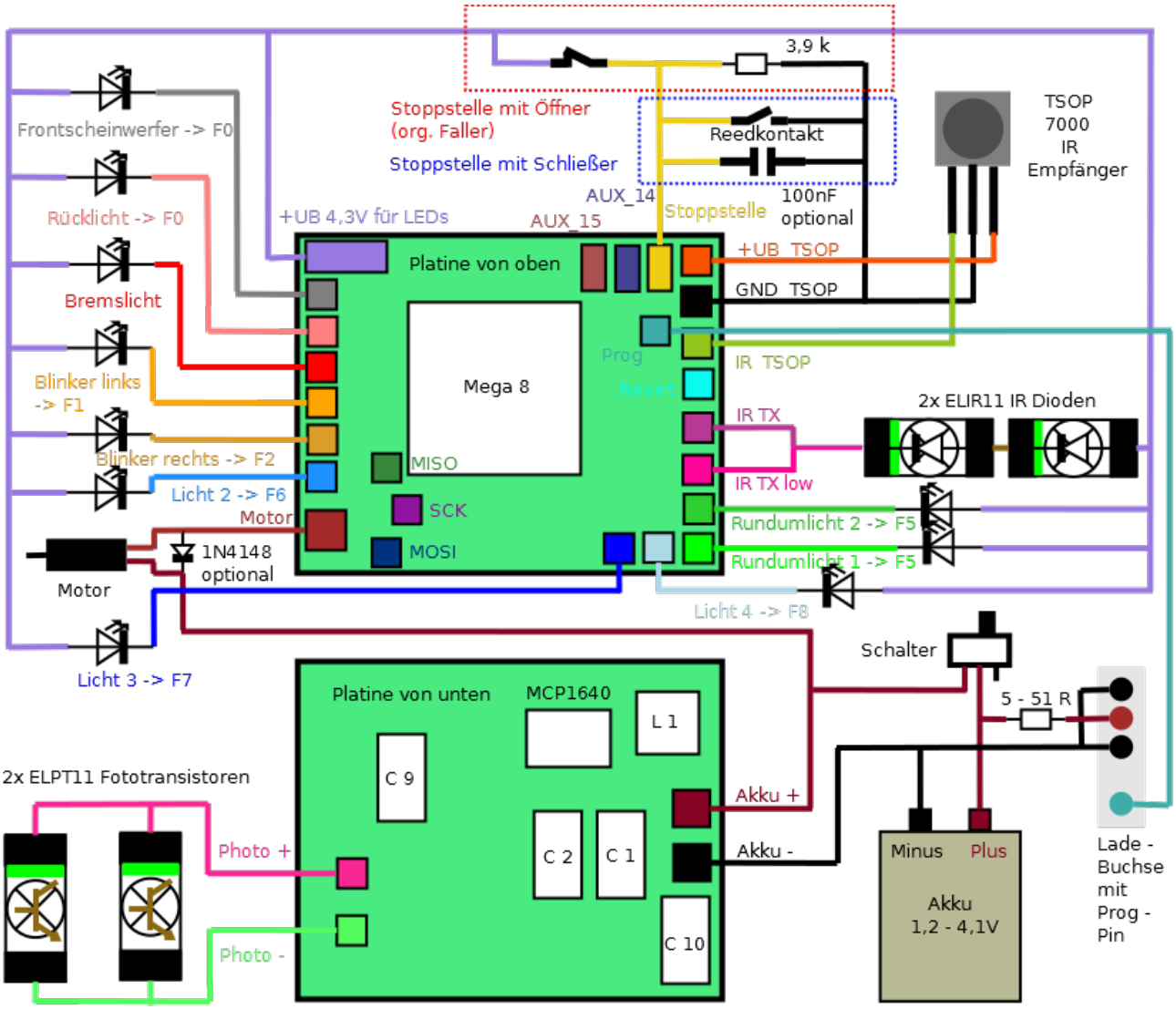
Hier ist eine kurze Videoanleitung zum Firmwareupdate zu finden:

Decoder 1 - wire Programmierung

Hier noch eine weitere zum Test der Decoderfunktionen:

Decoder Test

## Anschlussbelegung



**Konfigurationsvariablen**

| CV | Vorgabe | Beschreibung |
|---|---|---|
| 1 | 3 | kurze Decoderadresse |
| 2 | 20 | minimal PWM Impulsbreite (V min) |
| 3 | 10 | Anfahrverzögerung |
| 4 | 1 | Bremsverzögerung |
| 5 | 220 | maximal PWM Impulsbreite (V max.) |
| 9 | 5 | PWM Frequenz 1 - 7 -> 31kHz,4kHz,1kHz,500Hz,250Hz,125Hz,30Hz |
| 17 | 192 | Erweiterte Adresse Höherwertiges Byte der langen Adresse plus 192 |
| 18 | 3 | Erweiterte Adresse Niederwertiges Byte der langen Adresse |
| 23 | 2 | Faktor zur Geschwindigkeitsanpassung bei sinkender Akkuspannung<br>0(aus) – 3(max.) |
| 24 | 30 | minimale Akkuspannung (Akku leer Meldung kommt) in:<br>V * 10 (Akku Spannung ohne , ) |
| 25 | 42 | maximale Akkuspannung (voller Akku) in:<br>V * 10 (Akku Spannung ohne Komma bei R14/R15 33k/10k<br>das ist ca. auf 10% genau) |
| 26 | 0 | V_soll nach Decoderstart (0-255) |
| 27 | 0 | OSCCAL Korrekturwert für den AVR (wird automatisch ermittelt) |
| 29 | 0 | Konfiguration nach DCC-Norm<br>+32:Erweiterte (lange) Adresse (CV17 und CV18), sonst CV1 |
| 42 | 0 | AUX_7 Betriebsart<br>+0 : AUX_7 = aus -> hochohmig und AUX7 = ein -> GND<br>+16 : AUX_7 = aus -> +UB und AUX7 = ein -> GND<br>+32 : AUX7 = aus -> GND und AUX7 = ein -> +UB |
| 49 | 0 | Fahrzeugtyp 0-7 zur eigenen Verwendung für Ablaufsteuerungen |
| 55 | 0 | Softwareupdate: 1 = Updatemodus ein |
| 56 | 30 | Abstandsreglung FS vom Vordermann wird um CV56 Prozent verringert |
| 65 | 5 | Kickstartzeit *8ms |
| 66 | 80 | Kickstart Motorpulsbreite (PWM Wert) |
| 70-8 | | Effekte AUX_6 / F6 |
| 80-8 | | Effekte AUX_7 / F7 |
| 90-8 | | Effekte AUX_8 / F8 |
| 100-108 | | Effekte AUX_9 / F5 |
| 110-118 | | Effekte AUX_10 / F5 |

CVx0-7 sind je 8 Zeitwerte, die abwechselnd für die "ein" (CVx0/x2/x4/x6) und die "aus" (CVx1/x3/x4/x7) Zeiträume des AUX stehen (je * ca. 8ms), wobei ein Wert von 0 bedeutet, das dieser "Zeitraum" übersprungen wird. Werte zwischen 0 und 255 sind möglich. Beispiel:

CV70 = 10 -> 10* ca. 8ms = ca. 80ms ist der AUX an
CV71 = 20 -> danach ist der AUX ca. 160ms aus
CV72 = 10 -> danach ist der AUX wieder 80ms an
CV73 = 255 -> danach ist der AUX 2,04 Sek. aus
CV74 = 0 -> diese "ein" Zeit wird übersprungen
CV75 = 255 -> noch einmal 2,04 Sek. aus (insgesamt also über 4 Sek.)
CV76 = 100 -> jetzt wieder 800ms an
CV77 = 10 -> zum Schluss noch einmal 80ms aus danach beginnt es wieder von vorn

CVx8 kann Werte von 1-255 annehmen, mit folgender Bedeutung:

CVx8 = 1-253 es wird 1-253 mal die Schaltzeiten aus CVx0-CVx7 abgearbeitet
CVx8 = 254 es werden endlos die Schaltzeiten aus CVx0-CVx7 abgearbeitet

CVx8 = 255 alle Schaltzeiten werden ignoriert, der AUX ist immer an