”Noch eine DCC-Zentrale?
Gibt es doch wie Sand am Meer, kann
man doch kaufen oder irgend einen
anderen Bauvorschlag nehmen!”

OpenDCC

Version 2009

Wolfgang Kufer

www.opendcc.de

Rechtliche Hinweise:

Die Autoren dieser Anleitung sind in ihrer Eigenschaft als Autoren dieser Anleitung nicht verantwortlich für die Funktion oder Fehler der in dieser Anleitung beschriebenen Software. In keinem Fall haften die Autoren für entgangenen Umsatz oder Gewinn oder sonstige Vermögensschäden die bei der Verwendung oder durch die Verwendung dieser Programme oder Anleitungen entstehen können.
Bei der Erstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden.

Die Autoren können jedoch für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgend eine Haftung übernehmen.

In dieser Anleitung werden Warennamen ohne der Gewährleistung der freien Verwendbarkeit und meist ohne besondere Kennzeichnung benutzt. Es ist jedoch davon auszugehen, dass viele der Warennamen gleichzeitig eingetragene Warenzeichen oder als solche zu betrachten sind.

Alle verlinkten Webseiten in diesem Manual sind vor deren Aufnahme geprüft worden. Trotzdem weisen die Autoren darauf hin, dass auf die Inhalte und die zukünftige Gestaltung fremder Webseiten keinen Einfluß sowie Verantwortung besteht. Zum Zeitpunkt der Aufnahme konnten auf die verlinkten Seiten keine rechtsverletzenden Inhalte gefunden werden.
Sollten die Inhalte einer verlinkten Webseite gegen irgendwelche Rechtsnormen verstoßen, bitten die Autoren um einen Hinweis. Der betreffende Link wird dann umgehend entfernt.

Für Verbesserungsvorschläge und Hinweise auf Fehler sind die Autoren dankbar.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Beschreibung

OpenDCC[21] - Eine Zentrale für DCC

Abb. 1: Encoder (Quelle: archiv)

1.1 Überblick - OpenDCC

Ja, es gibt genug käufliche Zentralen, auch eine Reihe Selbstbauprojekte. Nur bei der Erprobung entdeckten wir das eine oder andere Defizit: mal ist die Zahl der zu fahrenden Loks (in weiser Selbstbeschränkung) zu stark beschränkt, mal gibt es sporadische Resets, mal ist die Priorisierung des DCC-Stacks ungeeignet für einen Anlagenbetrieb - wie kann man nur dem Lokpfiff eine höhere Priorität als dem Bremsbefehl geben?

Und manche Zentrale ist überhaupt nicht auf Datendurchsatz oder kurze Latenzzeiten ausgerichtet. Da wird nach einem Kommando erst mal CTS=off zurückgesendet, dann das empfangene Kommando bearbeitet und dann erst geht die Zentrale wieder online. Fällt nicht weiter auf, wenn nur 4 Loks fahren ...

Eine DCC-Message dauert etwa 6-12ms (je nach Adresse und Fahrstufen), daher ist eine intelligente Verwaltung und Priorisierung unabdingbar, sonst kommt es bei mehreren gleichzeitig beschleunigenden oder bremsenden Loks zu unschönem Fahrverhalten bzw. zum "Verbremsen".

Nachdem unsere bisherige Zentrale für den Fahrbetrieb nicht optimal geeignet war, enstand diese Selbstbauzentrale. **OpenDCC erreicht maximalen Durchsatz** am Gleis und auf der PC-Schnittstelle durch die intelligente Abarbeitung von Befehlen. Zudem ist es natürlich auch eine kompakte und preislich attraktive Lösung, welche speziell im Zusammenspiel mit dem PC andere Zentralen ersetzen kann.

Der maximale Durchsatz an DCC-Befehlen war oberstes Designziel bei der Entwicklung. Es ist ein intelligenter Refreshalgorithmus implementiert, der neue Fahrbefehle priorisiert und dann zyklisch wiederholt, wobei die Wiederholrate bei schon lange nicht mehr benutzten Loks zurückenommen wird. Bei der Priorisierung wird zusätzlich noch zwischen Funktionen, Beschleunigung und Abbremsen unterschieden: Bremsbefehle und damit die Fahrsicherheit haben höchste Priorität, selbst wenn viele Loks gleichzeitig gefahren werden.

Die Platine ist universell ausgelegt und kann mit passender Software (Kapitel 2.10 auf Seite 17) als Ersatz für gängige Zentralen (z.B. Lenz[1] oder Intellibox®[2]) bzw. Rückmeldebausteine (HSI88) verwendet werden. Bei den Bauteilen und beim Layout habe ich darauf geachtet, dass der Selbstbau leicht möglich ist, d.h. überwiegend "normale" Bauelemente und DIL-ICs. Es ist ein kleiner Booster integriert, um z.B. "schnell mal" fahren oder programmieren zu können. Für den Anlagenbetrieb insbesondere mit Rückmeldern sollte ein optogekoppelter Booster nachgeschaltet werden. Es ist bis auf Nothalt keine Tastatur vorgesehen, auch die Anzeige beschränkt sich auf ein paar Status-LEDs . Mit passender Software (z.B. P.F.u.Sch.) [6] kann aber ganz einfach mit einem PC oder Laptop gefahren werden.

[1] DigitalPlus ist ein eingetragenes Warenzeichen der Fa. Lenz Elektronik GmbH [15].
Seit Beginn der Entwicklung von Digital plus by Lenz wurde die Technik offengelegt. Diese Offenheit, Leistungsfähigkeit und vor allem die Ausbaufähigkeit haben dazu geführt, dass Digital plus von der NMRA als Basis der Norm für digitale Modellbahnsteuerungen verwendet wurde.

[2] Intellibox ist das eingetragene und geschützte Warenzeichen der Fa. Uhlenbrock [17].

Abb. 2: Ansicht eines fertigen Encoders (Kufer, archiv)

Eigenschaften:	
Digitalsystem:	DCC
Ausgänge:	Hauptgleis mit Minibooster(1.0A) Programmiergleis
Rückmeldung:	drei S88-Busse und zusätzlich echte Weichenrückmeldung
Bedienung am Gerät:	2 Taster (Stop-Go) 4 LEDs (CTRL, STOP, GO, RS232)
PC-Schnittstelle:	USB oder Seriell (RS232)
Bedienung am PC:	wahlweise Intellibox® oder Xpressnet™-Emulation
Handbedienung:	über Xpressnet™-Geräte, z.B. multiMaus® oder eigene Geräte ...
Nothalt:	Taster - Optokoppler Ein- und Ausgang
Lichtsteuerung:	Raumlichtsteuerung mit DMX **... wird in Zukunft nicht mehr unterstützt!**
Maße:	105 * 125 * 46mm

2 Eigenschaften

2.1 Spannungsverorgung:

Die Stromversorgung erfolgt mit einem externen Netzteil, welches geregelte Gleichspannung liefert. Das Netzteil sollte passend zur Spurweite gewählt werden. Für N empfiehlt sich 12-14V, für H0 16-18V.

2.2 PC-Anschluß:

Die Verbindung erfolgt entweder über USB (VID: 0x0403, PID: 0xBFD8, Product Description: "USB-IF OpenDCC V1.2") oder seriell (9-polige DSUB-Buchse RS232 an der Rückseite), es ist 19200 Baud, 8 Bit, keine Parity, 1 Stopbit einzustellen (bei Betriebsart Lenz) Die Auswahl der verwendeten Schnittstelle erfolgt durch Jumper auf der Platine. (Bestückungsalternative bzw. über Lötbrücken umschaltbar). Bei Verwendung der USB-Schnittstelle ist auf der PC-Seite ein VCP-Treiber zu installieren.
Das verwendete Protokoll des Interfaces hängt von der geladenen Software ab. Zur Zeit gibt es eine Universalsoftware, die per Jumper bzw. Compileoption auf Lenz V3.0 bzw. P50X (Intellibox) oder HSI88 eingestellt werden kann. Des weiteren ist bereits eine Portierung von miniDMX erfolgt.
Die Software ist über die normale Schnittstelle updatebar (Kapitel 3.10 auf Seite 109) , es ist hierfür kein Programmieradapter erforderlich.

Sicherheitshinweis:
Es ist (im Gegensatz zu manch käuflicher Lösung) eine vollwertige RS232 Schnittstelle mit korrekten Pegeln implementiert, diese ist galvanisch nicht vom PC getrennt. Dadurch ist fallweise die Schutztrennung, welche bei Spielzeug gefordert wird, aufgehoben. Zur Erreichung der Schutztrennung ist ein optogekoppelter Booster nachzuschalten. Siehe auch NEM 609. [30]

2.3 Schnittstellen zur Modellbahn:

- Es gibt zwei DCC-Ausgänge, Programmiergleis und Hauptgleis. Der Hauptgleisausgang ist mit einer intelligenten Datenverwaltung ausgerüstet, welche das Optimum aus dem DCC-Protokoll herausholt.

Das verwendete Stecksystem ist Wago Mini, Raster 3,5mm; (Der Einlötstecker hat die Art.Nr. 734-164, das Steckerteil die Nummer 734-104;) Alternativ sind auch handelsübliche Schraubklemmen RM5.04 möglich. Beide Ausgänge sind kurzschlußgeschützt (Programmiergleis 0,5A, Hauptgleis 1,5A), wobei der zu entnehmende Dauerstrom 1,0A nicht übersteigen soll. Dies kann zu einer Zwangsabschaltung wegen Übertemperatur führen.

- Optokopplereingang und Ausgang für Nothalt oder Weichenrückmeldung. Das Stecksystem ist wieder Wago wie oben.
- Es sind drei S88-Ports vorgesehen, diese dürfen in Summe mit 1,5A auf den 5V belastet werden, wenn der bestückte 5V-Regler das auch kann (siehe hierzu auch die Hinweise zur Schaltung). Die s88-Ports sind wahlweise als übliche Stiftleiste oder als RJ45-Buchse zu bestücken. Die Verkabelung mit den aus der Computertecknik bekannten Ethernetkabeln CAT-5 wird empfohlen, um die Störeinkopplungen auf den S88-Bus zu minimieren.

- **Hinweis zur Verdrahtung:**
  Der Gleisausgang wird bezogen auf die Netzteilmasse von OpenDCC[21] zwischen etwa 0,3V und 14,7V geschaltet (DCC wechselt ja Polarität) und darf daher nicht mehr mit Masse verbunden werden. Insbesondere Rückmelder können hier direkt oder über den Umweg RS232 und PC eine ungewollte Masseverbindung schaffen. In diesem Fall ist das Potential zwischen der Endstufe in OpenDCC[21] und dem Rückmelder durch einen optogekoppelten Booster zu trennen! Näheres hierzu auf der Seite Anlagenverdrahtung.

2.4 Bedienelemente und Anzeigen:

- LEDs:

		Anzeige:
LED1 orange PROG ●	**Normalbetrieb:**	
	aus:	alles okay
	blinkend:	zu viele Befehle, dem Host wird "not ready" gemeldet.
	Programmierbetrieb:	
	dauernd leuchtend:	Programmierung erfolgreich.
	flackernd:	ACK wird gerade gelesen
	blinkend:	Programmierfehler
LED2 rot STOP ●	dauernd leuchtend:	DCC wurde per Taster abgeschaltet.
	blinkend:	Nothalt über den externen Eingang ausgelöst.
	Rot und grün gemeinsam:	Fahrstufe 0
LED3 grün GO ●	dauernd leuchtend:	alles okay
	mit STOP im Wechsel blinkend:	Fehler in der Ausgangsstufe, entweder Kurzschluß oder Übertemperatur.
LED4 grün RS232 ●	**Diese LED zeigt den Zustand der Datenverbindung zu PC an:**	
	langsam blinkend:	keine Verbindung
	dauernd leuchtend:	Verbindung vorhanden
	leuchtend, kurze Pausen:	Datentransfer
	schnell blinkend:	Datentransfer blockiert (Fifo voll)

- Taster:

Taster	Funktion
STOP:	Erster Druck: Das System wird angehalten, die Spannung an Gleis bleibt bestehen, es wird jedoch nur Fahrstufe 0 ausgegeben. Zweiter Druck: Die Spannung am Gleis wird abgeschaltet (Nothalt)
GO:	Die Spannung am Gleis wird wieder eingeschaltet.

- Über eine kleine Zusatzplatine (siehe Kapitel 5) sind externe Handregler mit **Xpressnet™ - Anschluß** anschließbar. Das PC Interface und das Xpressnet™ - Interface laufen parallel und unabhängig voneinander.

2.5 Programmierschnittstelle:

- JTAG Port: Stecker X4, realisiert als übliche 10pol. Stiftleiste, RM 2.54
- Ponyprog[3] Port: für das bei www.lancos.com ladbare Programmiertool Ponyprog ist eine zweite RS232 auf der Platine vorgesehen. Hinweis: bei der Konfiguration von Ponyprog ist der RST-Ausgang invertiert zu setzen.
- 6-polige ISP-Schnittstelle (JP3): Hier kann für die Erstinbetriebnahme der übliche Programmieradapter eigesteckt werden.
- Ein Update der Software ist ohne Öffnen des Gerätes und ohne Programmieradapter entweder über RS232 oder USB möglich.

2.6 Sonstige Ports:

RS485-Port für direkte DMX-Ansteuerung oder Lichtsteuerung mit DCC. Das Raumlicht kann mit normalen Weichenschaltbefehlen von der Modellbahnsteuerung aus gedimmt werden. Es gibt preiswerte DMX-Dimmer, diese können direkt an OpenDCC angeschlossen werden. Der Anschluß auf der Platine erfolgt mit Schraubklemmen.

... wird in Zukunft nicht mehr unterstützt!

2.7 Technische Basis:

OpenDCC[21] ist auf einem Atmel AVR Prozessor programmiert. Dieser ist zusammen mit einem kleinen Booster (H-Brückenschaltung, basierend auf STM L6206) und kleineren Peripherieschaltungen auf auf einer Platine 120*100mm integriert. Das Layout ist zweiseitig mit breiten Leiterbahnen und Durchsteckbauelemente erstellt, so dass der Nachbau auch mit Hobbymitteln möglich sein sollte.

Die Software ist modular in C geschrieben (mit WinAVR[2]) und als Open Source verfügbar.

2.8 Bisherige Erprobung und bekannte Mängel:

folgende Tests wurden bisher durchgeführt:

- HSI88-Mode: Betrieb mit Traincontroller V 5.8[7] per RS232 und USB. Kontrolle der richtigen Bitpositionen, Kontrolle der Signale am S88-Bus
- HSI88-Mode: Betrieb mit Railware 5.0[27] und USB (Alex)
- Lenz-Mode: Baudratenumschaltung, Modeumschaltung
- Lenz-Mode: Betrieb mit Traincontroller und 28 Fahrstufen
- Lenz-Mode: Schaltbefehle
- Lenz-Mode: Betrieb mit P.F.u.Sch.[6] und 28 Fahrstufen, Lichtfunktion
- Lenz-Mode: Decoder auslesen im CV-Betrieb mit P.F.u.Sch.[6]
- Lenz-Mode: Test der S88 unter Railware[27] (Alex)
- Lenz-Mode: Programmieren der Opendecoder (ab V0.14)
- Intellibox®-Mode: Betrieb mit TC (mit RS232 und USB inkl. BABI) an einer mittleren Anlage (16 Booster, 320 Meldeabschnitte)
- Intellibox-Mode: Test der S88 unter Railware[27] (Alex)
- Intellibox-Mode: Betrieb mit Railware[27] und 28 Fahrstufen, Fahren, Schalten, Funktionen
- Intellibox-Mode: Schaltbefehle und Zusammenspiel mit OpenDecoder

- Intellibox-Mode: Programmieren mit TrainProgrammer[7] von www.freiwald.com sowie mit railware[27] (Alex)
- Intellibox-Mode: Betrieb mit srcpd[26] und spdrs60[18] - Schalten, Fahren, Rückmeldung mit S88.
- Intellibox-Mode: Betrieb mit Rocrail[28] (Open Source)
- DMX-Mode: Lichtansteuerung für Raumlicht

2.9 Einschränkungen und Mängel:

Der Zustand von Weichen wird in OpenDCC[21] nicht permanent gespeichert, bei der Initialisierung von PC-Programmen wird (z.Z.) immer "Stellung grün (=0)" zurückgeliefert. Während des Betriebes wird die Weichenposition aber gespeichert. Wenn sich ein PC Programm daran stören sollte, gibt es folgenden Workaround: Vor dem Steuern der Anlage mit dem PC einmal ALLE Weichen durchschalten.

2.10 Softwaregrundlagen

OpenDCC[21] kann mit unterschiedlicher Software geladen werden. Diese nachfolgend beschriebene Software kann per Jumper auf folgende Betriebsarten programmiert werden:

Jumper / Betriebsarten		
JP2	JP1	Mode
offen	offen	Betrieb als Zentrale, Befehlssatz für eine Emulation der Intellibox®von Uhlenbrock. Der Befehlssatz kann auch auf Lenz-Emulation umgestellt werden (per #define in config.h, danach neu übersetzen).
offen	zu	Betrieb als DMX Steuerzentrale, Bedienung durch Tasten (Remote Control noch nicht enabled) **... wird in Zukunft nicht mehr unterstützt!**
zu zu	offen zu	Betrieb als HSI88 Interface, Befehlssatz gemäß Littfinski[22] Testbetrieb: Die LEDs blinken, alle Daten auf der Schnittstelle werden mit 19200 Baud einfach wieder zurückgesendet.

Wenn kein Jumper gesteckt ist, dann wird mit dieser Software aus OpenDCC eine Zentrale. Es werden Befehle vom PC nach DCC umgesetzt, wobei OpenDCC wahlweise eine Lenz-Zentrale oder eine Intellibox (default) emuliert. verhält. Es sind jedoch nicht alle Befehle nach Lenz V3.0 oder IB implementiert. Es fehlt

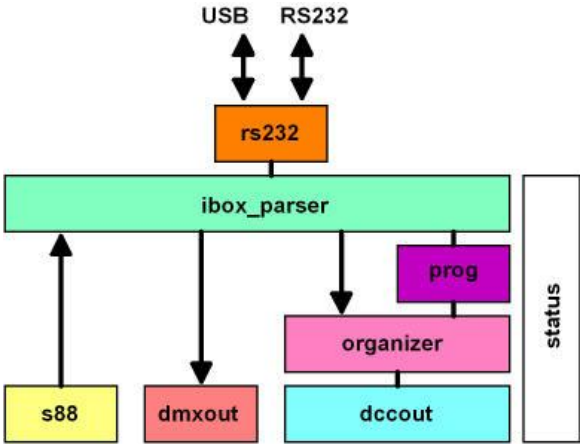


Abb. 3: Module der Software (Kufer, archiv)

z.B. der Support für 27 Fahrstufen und die Unterstützung für Mehrfachtraktionen.

Die Software wurde in C mit WinAVR[2] (basiert auf dem avrgcc[1]) geschrieben und mit AVR-Studio[5] (gibts auf www.atmel.com) simuliert.
Das Ganze ist als Open Source unter der GNU License freigegeben.
Bei der Entwicklung wurde besonders auf maximalen Durchsatz an DCC-Befehlen geachtet; hierzu ist auch ein intelligenter Refreshalgorithmus implementiert, der neue Fahrbefehle priorisiert und dann zyklisch wiederholt, wobei die Wiederholrate bei schon lange nicht mehr benutzten Loks zurückgenommen wird. Bei der Priorisierung wird zusätzlich noch zwischen Funktionen, Beschleunigung und Abbremsen unterschieden: Bremsbefehle haben höchste Priorität˙

Die Software besteht aus aus mehreren Modulen:

- HARDWARE.H: Einstellen des Prozessors und Portdefinitionen
- CONFIG.H: Einstellen der Betriebsarten, Speichergrößen, usw.
- DCCOUT.C: Erzeugung des DCC-Protokolls (DCC Encoder)
- DMXOUT.C: Erzeugung des DMX-Protokolls (DMX Encoder)
- ORGANIZER.C: DCC-Kommandoverwaltung (Repeat), Refresh)
- LENZ_PARSER.C, IBOX_PARSER.C: Übersetzen der PC-Befehle in die entsprechenden internen Aufrufe
- LENZ_PROGRAMMER.C, IBOX_PROGRAMMER.C: Abarbeiten von Programmierbefehlen

- RS232.C: Schnittstelle zum PC (mit Interrupt + Fifos)
- STATUS.C: Zustandsverwaltung, Timertick, Keys und LEDs
- S88.C: Auswerten der Rückmelder
- MAIN.C: Init, Loop

2.10.1 DCC-Erzeugung

Das Modul DCCOUT kümmert sich um die Erzeugung des Gleissignals.

- **Wie sieht DCC überhaupt aus?**
  Bei DCC wird das Gleissignal immer mit konstanter Spannung, aber wechselnder Polarität übertragen. Durch die Zeitpunkte der Polaritätswechsel wird die Information kodiert, welche an die Lok/Weiche übertragen wird.

  Die Codierungsregel ist relativ einfach: zwei Polaritätswechsel nach je 58 µs bedeuten **1**, zwei Wechsel nach je 116 µs bedeuten **0**. Da es immer 2 Polaritätswechsel sind, ist das Ausgangssignal (normalerweise) gleichspannungsfrei und die Polarität am Empfänger ist egal. Eine **1** dauert also zweimal 58 µs = 116 µs, eine **0** ist doppelt so lang.

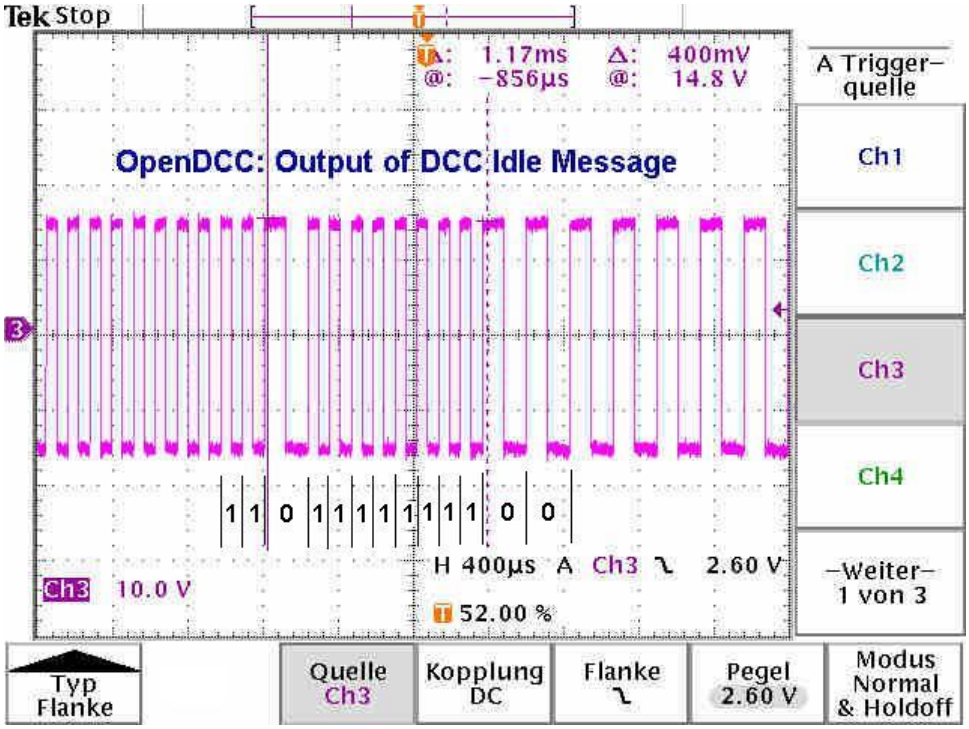


Abb. 4: DCC Signal (Quelle: archiv)

Auf diese Art und Weise wird ein serieller Bitstrom an die Lok gesendet. Um nun dem Empfänger eine Wertung dieser Bits zu ermöglichen, muß der Datenstrom gegliedert werden. Bei DCC geschieht das durch Aufteilung in Preamble, Nutzbytes und Prüfsumme. Die Preamble kennzeichnet jeweils den Beginn einer neuen Nachricht. Preamble, Nutzbytes und Prüfsumme (je 8 Bits) sind jeweils durch eine **0** voneinander getrennt.

Damit die Preamble leicht vom Empfänger leicht erkannt werden kann, muß sie nach einer Regel aufgebaut sein, die im normalen Datenstrom nicht vorkommt. Bei DCC ist die Preamble eine Folge von mind. zwölf **1** Bits. Da ja die Nutzbytes nach 8 Bits durch eine **0** getrennt sind, kommt diese lange **1**-Sequenz im normalen Datenstrom nicht vor.

Nach der Preamble kommen zwei bis fünf Nutzbytes (jeweils durch **0** getrennt), welche i.d.R. die Adresse und die Fahrstufe enthalten. Das ursprüngliche DCC-Format sah nur 2 Nutzbytes vor, diese wurde später bis zu 5 Nutzbytes erweitert. An diese Nutzbytes wird die 8-Bit XOR-Summe dieser Nutzbytes als Prüfsumme (checksum) angehängt. Wenn der Empfänger nun die XOR-Summe über die alle empfangenen Bytes bildet, dann muß sich 00000000 ergeben, nur dann ist die Nachricht gültig.

- **Welche Bedeutung haben die einzelnen Bits**?
  Mit dem ersten Byte wird eine generelle Aufteilung in Lokbefehle, Zubehördecoder (Accessory Decoder) und Programmierbefehle (Service Mode) vorgenommen. Alle folgenden Bytes sind dann je nach Befehlsgruppe unterschiedlich codiert.

		Erstes Byte
0	00000000	Die Adresse 0 ist als Broadcast für alle Lokomotiven reserviert.
1-127	0AAAAAAA	Beginnt das erste Byte mit einer 0, so handelt es sich um einen Lokfahrbefehl mit kurzer Adresse.
128-191	10AAAAAA	Der Bereich (128-191)(einschließlich) ist für Zubehördecoder.
192-231	11000000-11100111	Es handelt sich um einen Lokbefehl mit langer Adresse, die weiteren Adressbits folgen dann im nächsten Byte.
232-254	11101000-11111110	Dieser Bereich ist reserviert für Erweiterungen.
255	11111111	Damit wird keiner adressiert (IDLE).

  Im Prinzip beginnt eine DCC-Message immer mit einem (oder zwei Adressbytes), gefolgt von einem (oder mehreren) Befehlsbytes, in denen die jeweilige Aktion bzw. Lokgeschwindigkeit kodiert ist.
  Das DCC-Protokoll ist im Laufe der Jahre immer wieder erweitert worden; Dabei haben sich leider verschiedene Darstellungen für die Geschwindigkeit (14,27, 28 und 126 Fahrstufen) etabliert. Die Bedeutung der weiteren Bytes kann man bei der NMRA nachlesen.

- **Beispiel 1: Fahrbefehl an Lok 5, Fahrstufe 4 (von 14 Stufen), vorwärts**
  Es wird ein Befehl mit 3 Byte geschickt:

```
Preamble Startbit  Adressbyte (=0AAAAAAA)
         Startbit  Datenbyte  (=01DCSSSS)
         Startbit  Prüfsumme
         Stopbit
```

  Dabei bedeutet im Adressbyte AAAAAAA [A6 .. A0] die Lokadresse und und im Datenbyte D [Direction=Richtung], C [Lichtfunktion], SSSS [Speed]. Das DCC-Telegramm wird wie folgt zusammengebaut:

```
Adressbyte = (Lokadresse & 0x7F);
Datenbyte  = 0b01000000 | (speed & 0x0F) |
             (Richtungsbit << 5);
Prüfsumme  = Adressbyte ^ Datenbyte;
```

Das Richtungsbit ist 1 für vorwärts, Null für rückwärts.

Preamble	0	Adressbyte	0	Datenbyte	0	Prüfsumme
11111111111111	0	00000101	0	01100100	0	01100001

- **Beispiel 2: Weichenansteuerung per DCC**
  Es wird ein Befehl mit 3 Byte geschickt:

```
Preamble Startbit  Adressbyte 10AAAAAA
         Startbit  Datenbyte  1AAA1BBR
         Startbit  Prüfsumme
         Stopbit
```

Dabei bedeutet im Adressbyte AAAAAA [A5 .. A0] und im Datenbyte AAA [A8 .. A6], wobei die Adressen A8 bis A6 invertiert übertragen werden. BB ist die lokale Adresse am Decoder (0,1,2,3), R ist das Outputbit, d.h. welche Spule aktiviert werden soll. In der Zentrale wird nun z.B. aus der von PC übermittelten Adressen wie folgt das DCC-Telegramm zusammengebaut:

```
Adressbyte = 0x80 + (adresse & 0x3F);
Datenbyte  = 0x80 + (adresse / 0x40) ^ 0x07) * 0x10;
Prüfsumme  = Adressbyte ^ Datenbyte;
```

Die Adresse 0 ist als Broadcast reserviert, so dass die erste Weichengruppe bei Adresse 1 liegt.
Leider gibt es bei den PC-Protokollen der einzelnen Zentralen kleine, aber störende Unterschiede im Adressbereich und in der Interpretation des Outputbits sowie des Activatebits (im obigen Beispiel die 1).

Uhlenbrock (Intellibox®) beschreibt wie folgt (und überträgt auch so):

'r' (red = thrown) and ('r' may also be spec'd as '0')
'g' (green = closed) and ('r' may also be spec'd as '1')

Der Adressbereich in der PC Schnittstelle beginnt dagegen entgegen der Dokumentation bei 1. Das heißt, der erste Decoder liegt auf Adressen 1

... 4; Die Intellibox sendet auch nie den Abschaltbefehl auf das Gleis, auch wenn dieser vom PC aus angefordert wird.

Lenz beschreibt das Richtungsbit als D2 mit folgendem Text:
D2:
D2 = 0 bedeutet Ausgang 1 der Weiche gewählt.
D2 = 1 bedeutet Ausgang 2 der Weiche gewählt.

Allerdings läßt es sich nirgends rausfinden, was Ausgang 1 oder 2 bedeutet. An Hand eines Logfiles scheint rot und grün vertauscht zu sein. OpenDCC bildet diese Invertierung (abschaltbar) durch eine CV nach.
Dafür beginnen die Weichenadressen bei Lenz bei der Adresse 0 und den Abschaltbefehl gibt es.

- **Allgemeine Regeln für die Pakete**

  - Präambel: Ein Empfänger muß eine Präambel aus mind. 12 Einsern erkennen, eine Preamble mit weniger als 10 Einsern muß er verwerfen; eine Zentrale muß mind. 14 Einser senden. Das Stopbit darf gleichzeitig zur Präambel des nächsten Befehls gehören. Wenn die Zentrale im Programmiermode betrieben wird, dann muß die Präambel 20 Bits lang sein.
  - Länge: Solche Packete können von 3 bis 6 Bytes lang sein, jedoch ist bei allen Packeten das letzte Byte eine XOR-Prüfsumme über die vorherigen Bytes.
  - Zeitlicher Abstand: Pakete an den *gleichen* Decoder müssen 5ms Abstand haben. Nach spätestens 30ms muß wieder irgendein DCC Packet gesendet werden, damit die Decoder nicht die Lust verlieren und auf analog umschalten.

- **Wie werden nun diese Bits in OpenDCC[21] erzeugt?**
  Es wird der Timer 1 benutzt, dieser wird im CTC-Mode (Clear Timer on Compare-Match) betrieben, d.h. nach Erreichen des Compare-Wertes beginnt der Timer wieder bei Null. Zugleich wird beim Erreichen des Comparevalues das DCC-Bit per Hardware umgeschaltet. Die Ausgänge der Komparatoren sind mit dem Leistungstreiber verbunden und erzeugen das Gleissignal.

  Gleichzeitig mit dem Neustart des Timers wird ein Interrupt ausgelöst, die Interruptroutine trägt die Compare-Werte für das nächste Bit ein, je nach dem ob 0 oder 1 ausgegeben werden soll. Jeder Polaritätswechsel des DCC-Signals erzeugt also einen Interrupt, jeder zweite Interrupt schaltet

um ein Bit innerhalb der DCC-Message weiter.

Beginnend mit der Preamble hangelt sich die Interruptroutine nun Bit um Bit durch die DCC-Message, wobei sie auch noch gleich den Overhead des DCC Protokolls (sozusagen Layer 1) mit erledigt, d.h. es wird die Preamble und die Prüfsumme (XOR-Byte) in der ISR errechnet und mit ausgegeben.

Das "nächsthöhere" Programm muß also nur die Nutzbytes zusammenstellen und an DCCOUT übergeben (im Datenarray "**next_message**"). Die Übergabe ist durch ein Semaphor ("**next_message_count**") verriegelt - DCCOUT übernimmt nur, wenn dieses Semaphor größer Null ist und decrementiert bei jeder Übernahme dieses Semaphor um 1.

2.10.2 S88.C, Einlesen der Rückmelder

In S88.C werden die Rückmelder eingelesen und auf Änderungen überwacht. Es gibt drei S88-Stränge, diese werden parallel eingelesen. Wenn ein Strang kürzer ist (d.h. weniger Bits hat), dann wird dieser Strang zwar mitgetaktet, die Bits werden jedoch nicht mehr ausgewertet.

Es werden 1024 Rückmelder unterstützt, diese sind allerdings nicht in allen Interface-Formaten verfügbar. Wenn der HSI88-Mode aktiviert ist, dann können nur 31*16 = 496 Rückmeldekontakte abgefragt werden. Mehr ist in der HSI88 Beschreibung nicht vorgesehen und auch PC-Steuerprogramme limitieren bei Wahl eines HSI88 auf 496 Rückmelder.

Es gibt zwei Datenarrays, **s88_data** und **s88_change**. Wenn beim Einlesen des S88-Bus nach S88-Data ein Wechsel zum bisherigen Zustand festgestellt wird, dann wird an der entsprechenden Stelle in **s88_change** das Bit gesetzt; Wenn dieser Wechsel an den PC gemeldet worden ist, dann wird das Change-Flag wieder gelöscht;
Die Meldung an den PC erfolgt je nach Protokoll entweder je Nibble (Lenz) oder als 16-Bit Wert (Littfinski[22] HSI88).

2.10.3 STATUS.C, Timertick und Zustandsverwaltung

In Status.c werden die LEDs angesteuert, Tasten und externe Eingänge abgefragt und der Zustand von OpenDCC verwaltet.

Die LED Ansteuerung erfolgt timergesteuert mit Hilfe einer Kontrollstruktur **led_pwm**. Diese enthält für jede LED die Restleuchtdauer in aktuellen Zustand, die Einschaltzeit und die Ausschaltzeit. Wenn die Restleuchtdauer zu Null gesetzt wird, bleibt der Zustand für immer erhalten. Damit lassen sich ganz einfach unterschiedlich Blinkmodi generieren:

```
LED_RS232_OFF;
   led_pwm.rs232_rest = 20000L / TICK_PERIOD;
   led_pwm.rs232_ontime = 50000L / TICK_PERIOD;
   led_pwm.rs232_offtime = 150000L / TICK_PERIOD;
   break;
```

Beispielsweise wird mit diesen Einstellungen die RS232-LED für 20ms ausgeschaltet, fortan blinkt sie mit einem Puls-Pausenverhältnis von 1:3 und 5Hz. **TICK_PERIOD** ist das Interval der Timerinterrupts und wird in config.h auf 5ms gesetzt. Alle Zeitangaben erfolgen in µs, d.h. 50000L bedeutet 50ms; Zu beachten ist, dass die Zeiten als unsigned char gespeichert werden, d.h. die Zeiten müssen kleiner 1,25s sein.

Weitere Verwendung findet der Timer als Entprellmaschine der Tasten und als Timeoutzähler.

2.10.4 ORGANIZER.C, Lokverwaltung und Befehlsverwaltung

Diesem Modul obliegt es, sowohl für geringe Latenzzeit und hohen Datendurchsatz bei den Fahrbefehlen zu sorgen, als auch den Refresh bereits fahrender Loks zu organisieren. Hier liegt einer der Hauptvorteile von OpenDCC gegenüber käuflichen Zentralen!

Die eigentlich limitierende Grenze ist der Durchsatz am Gleis - DCC hat im Schnitt etwa 4000 - 5000 Bit/s bzw. 100 Nachrichten / s. Deshalb ist in OpenDCC ein Regelwerk definiert, das dafür sorgt, dass neue Befehle schnell aufs Gleis kommen und dann mit sinkender Priorität in den allgemeinen Refresh einsortiert werden.

Bremsbefehle (d.h. Befehlen, bei denen die Geschwindigkeit gegenüber der aktuellen Geschwindigkeit verringert ist) werden mit höchster Priorität aufs Gleis gelegt, parallel anliegende Beschleunigungsbefehle an eine andere Lok müssen ein klein bischen warten.

Das Regelwerk des Organizers

```
für Lokfahrbefehle:
Lokgeschwindigkeit in den Lokspeicher eintragen.

Alle Queues und Buffer nach dieser Lok durchsuchen, ob ev.
da noch ein älteres Kommando an diese Lok wartet - wenn ja,
dieses ersetzen.

Geschwindigkeit größer oder kleiner als die aktuell gefahrene
Geschwindigkeit?
   - falls kleiner: -> Befehl in die High Priority Queue.
   - falls größer (oder Funktion bzw. Accessory):
                    -> Befehl in die Low Priority Queue.

Wenn ein Befehl aus einer Queue verarbeitet worden ist, dann
kommt er anschließend in den Repeat Buffer:
   Speedbefehle mit 3 Repeats, Accessory-Befehle mit 2 Repeats.

Bei der Gleisausgabe gilt dann folgende Reihenfolge:
   1. High Priority Queue.   wenn diese leer ist:
   2. Low Priority Queue.    wenn diese leer ist:
   3. Repeatbuffer.          wenn dieser auch leer ist:
   4. Lokspeicher refreshen. (jeweils in der Sequenz:
              Speed, Funkt1, Speed, Funkt2, Speed, Funkt3, Speed)
```

Darüber hinaus beachtet der ORGANIZER noch ein paar Zusatzregeln wie z.B. kein Befehl an die gleiche Loks in Folge, Vorrang für Broadcast-Befehle.

2.10.5 Anzahl von Lokomotiven und Lokformat

OpenDCC kann 9999 (Lenz-Parser) bzw. 16383 (Intellibox-Parser) verschiedene Loks adressieren, diese können aber nicht gleichzeitig fahren. Die Zahl der gleichzeitig zu fahrenden Loks ist (bei der aktuellen SW-Version) auf 64 begrenzt (Speicherplatz im Lokspeicher), mehr Loks kann das DCC-Protokoll eigentlich nicht sinnvoll ansprechen. Eine DCC-Message dauert etwa 8-12ms, je nach Adresse und Fahrstufen, d.h. bei 10 oder mehr gleichzeitig bremsenden Loks beginnt es interessant zu werden. Es können aber durchaus mehr Loks vorhanden sein, der Lokspeicher und Repeatbuffer werden bei Platzproblemen durch Verdrängen der "ältesten" Lok angepasst. Diese "älteste Lok" wird dann nicht mehr refreshed.

OpenDCC kann die Loks mit DCC14, DCC28 bzw. DCC128 ansprechen. Das wird für den Fall des Lenz-Protokolls einfach durch den Fahrbefehl festgelegt. Für den Fall des IB-Protokoll gibt es keinen Befehl, welcher das Format definiert. Es wurden daher zwei zusätzliche Befehle definiert, welche die Übergabe eines

Lokformats vom PC erlauben:

- **P50X-ASCII**

```
LS {Lok#, [Format], [Steps]}
als Format ist nur "2" (=DCC) erlaubt.
```

- **P50X-BINÄR**

```
0x86: XLokCfgSet - length 1+3;
byte1+byte2 = addr, byte3 = format
```

Manche PC Steuerungsprogramme speichern das Lokformat nicht selbst, sondern fragen bei der Modellbahnzentrale nach, welches Format die einzelne Lok hat. OpenDCC unterstützt das dadurch, dass ein einmal gesendetes Lokformat für eine bestimmte Lok im EEPROM der Zentrale gespeichert wird und bei der der nächsten Abfrage des PC-Programms wird dann dieses gespeicherte Format zurückgemeldet.

Wenn eine Lok noch nie gefahren wurde, so wird das DEFAULT Format zurückgemeldet. OpenDCC arbeitet mit DCC28 als default-Einstellung (Dies ist durch eine CV änderbar). Um nun eine Lok auf ein anderes Format umzustellen, genügt es, einmal diese Lok in dem gewünschten Format anzusprechen. Es können maximal 64 Loks gespeichert werden, die ein von der default-Einstellung abweichendes Format haben.

Intern werden alle Geschwindigkeiten im DCC128-Format abgespeichert. Innerhalb des Parsers wird fallweise vom reduzierten Lenz-Format auf DCC128 umgesetzt. (Routinen: **convert_speed_to_rail()** und **convert_speed_from_rail()**). Diese Umsetzung erfolgt auch bei der Gleisausgabe je nach aktuell für diese Lok eingestelltem Format. Wenn als Parser der Intellibox®-Parser verwendet wird, so ist keine Umsetzung auf der Parserseite erforderlich, P50X liefert immer 0..127 als Geschwindigkeit. Im MSB der Geschwindigkeit ist die Fahrrichtung kodiert.

Zusammenfassung der Funktionsaufrufe des ORGANIZER

```
init_organizer(void)
organizer_ready(void)
do_loco_speed_f(addr, speed, format)   Geschwindigkeit und Format
                                       einer Lok einstellen
do_loco_speed(addr, speed)             Geschwindigkeit einer Lok
                                       einstellen
do_loco_func_grp0(addr, funct)         Lichtfunktion
do_loco_func_grp1(addr, funct)         Funktionen f1-f4
do_loco_func_grp2(addr, funct)         Funktionen f5-f8
do_loco_func_grp3(addr, funct)         Funktionen f9-f12
do_accessory(addr, output, activate)   Weichenschaltbefehl
do_all_stop(void)                      Bremsbefehl
```

2.10.6 RS232.C, Kontrolle der seriellen Schnittstelle

OpenDCC läuft im Falle LENZ mit 19200 Baud (das ist auch default bei Lenz V3.0). Diese Rate kann mit dem Befehl BAUD (0xF2) auch auf 38400, 57600 und 115200 Baud umgestellt werden bzw. schon beim Übersetzen auf einen anderen Wert gestellt werden. (**init_rs232**-Aufruf in main.c)
Das Datenformat ist immer 8 Datenbits, kein Parity, 1 Stopbit (8n1).

Die gesamte Kommunikation erfolgt interruptgesteuert und wird mit Fifos (Tiefe je 64 Bytes) gepuffert. 5 Bytes bevor der Eingangsspeicher voll läuft, wird per Hardware-Handshake der PC angehalten.

Wenn OpenDCC ein BREAK empfängt (d.h. Rx liegt sehr lange auf +12V, bzw. es werden lauter 0 empfangen), dann wird wieder automatisch 19200 Baud eingestellt. Siehe hierzu auch die Kommentare in ISR(**SIG_UART_RECV**).

2.10.7 MAIN.C, Hauptprogramm

Es gibt 4 Interruptroutinen:

- DCC Counter: Dieser Interrupt veranlaßt das Neuprogrammieren des Counters 1. Damit werden die Sequenzen für DCC0 und DCC1 auf den Vergleichsausgängen dieses Zählers erzeugt.
- Timer Tick: damit werden Tastaturabfragen und Timeouts gesteuert.
- UART Rx: Empfangenes Byte wird in Fifo abgelegt.
- UART Tx: Ein Byte im Sendefifo wird abgeschickt.

Diese 4 Interrupts übernehmen somit alle zeitkritischen Funktionen. Die Abarbeitung der Befehle und die Kontrolle wird vom Hauptprogramm gemacht, das zyklisch die Module aufruft. Die Routinen in den Modulen müssen zurückgeben, sofern sie nichts zu tun haben.

Das Hauptprogramm gibt diese Struktur wieder:

```
int main(void)
  {

    init_main();                    // all io's
    init_dccout();                  // timing engine for dcc
```

```
    init_organizer();                 // engine for command repetition,
                                      // memory of loco speeds and types
    init_rs232();                     // isrs+fifos from/to pc
    init_interrupt();
    init_tick();                      // 5ms timer tick
    init_programmer();                // dcc programmer
    init_parser();                    // command parser

    set_opendcc_state(RUN_OFF);       // abgeschaltet

    while(1)
     {
       run_state();                   // check short and keys
       run_organizer();               // let run command organizer
       run_parser();                  // check commands from pc
     }
  }
```

2.11 Rückmeldungen:

2.11.1 Einleitung

Hier ist eine Zusammenfassung der verschiedenen Aspekte (Adressbereiche, Updateraten usw.) bei der Rückmeldung in OpenDCC.

2.11.2 Physikalische Implementierung, Timing

Die Kontrolle der S88-Leitungen erfolgt per 'Bit-Bang' durch Software. Die Rückmelder werden innerhalb des Task-Schemas von OpenDCC mittels zylischer Abfrage eingelesen, wobei die eingestellten Mindestzeiten durch eine Timerüberwachung garantiert werden. Die Maximalzeiten können je nach aktuelle Prozessorlast etwas streuen, so dass die Signale CLK, LOAD, RESET etwas Jitter enthalten. Dieser ist aber ohne Belang.
Die Zeiten sind in Schritten von 10µs einstellbar, in der Defaulteinstellung werden je 20µs für High und Lowzeit des Clocks verwendet. OpenDCC ist damit vermutlich eine der schnellsten s88-Implementierung.
Es werden immer alle 3 Stränge simultan eingelesen, wobei der längste Strang die Anzahl der Taktpulse bestimmt.
Beispiel: 288 angeschlossene Rückmelder, welche auf 3 Stränge aufgeteilt sind werden ca. alle 5ms komplett eingelesen zu werden.

2.11.3 Hostprotokoll

Zum Absenden der Rückmeldungen sind 3 verschiedene Protokolle implementiert, diese haben folgende Merkmale:

Protokoll:	Intellibox (p50xb)	Lenz (xpressnet)	Littfinski HSI88
max. Anzahl:	1024	1024	496
unterteilt in:	je 8	je 4	je 16
Bitorder:	invers	normal	normal
Technik:	Polling	Pushing	Pushing

Eines der Protokolle kann per Compileoption als Hostprotokoll ausgewählt werden, wobei beim HSI88 natürlich kein Zentralenbetrieb möglich ist. Getestet wird bei mir i.d.R. mit p50xb.

2.11.4 Adresszuordnung

Es gibt innerhalb der Zentrale vier Adressbereiche für Rückmelder. Diese Bereiche folgen aufeinander, jeder Bereich darf auch 0 Byte groß sein:

Bereich	1	2	3	4
Zuordnung	S88-1	S88-2	S88-2	Weichenrückmeldung
festgelegt durch:	CV9	CV10	CV11	CV16

Die Hardwarebereiche S88-1, -2 und -3 werden jeweils durch ihre Größe definiert (Anzahl der Rückmeldebits in Einheiten von je 8 Bit) und schließen sich aneinander an. Es ist darauf zu achten, dass die jeweiligen Größen richtig eingestellt sind, da sonst die Rückmeldung nicht funktionieren kann. Der Beginn des Bereiches für die Weichenrückmeldung wird mit CV16 definiert, es kann/darf also durchaus eine Lücke in den Rückmeldebits vorhanden sein. Vorteil: Beim Hinzufügen weiterer S88-Module verschieben sich die Weichenrückmeldungen nicht.

Beim Protokoll p50xb werden immer alle vier Bereiche über den normalen Rückmeldeweg zurückgemeldet, die Bits für die Weichenrückmeldung sind hier einfach in die normalen Rückmeldebits einsortiert. Die Auswertung ob eine Weiche umgelaufen ist, muß dann am PC erfolgen.

2.11.5 Xpressnet und Schaltinformationen

Beim Xpressnet ergeben sich Einschränkungen, weil die Rückmelder und Weichenbefehle sich den Adressbereich teilen. Laut Lenz-Doku gibt es 1024 Rückmelder, dabei wird das Byte 1 der Anfrage von 0..127 kodiert und es werden je 8 Rückmelder damit erfaßt. Des weiteren sind 1024 Weichen vorgesehen, hier deckt das Byte 1 einen Bereich von 0..255 ab, es werden je 4 Weichen adressiert. Die Adressbereiche liegen übereinander, es werden also max. 2048 Bits addressiert.

0——— Feedback ———1024
0————————— Weichen —————————2048

Anhand der Graphik wird auch klar, warum das Xpressnet nur auf dem ersten 512 Weichen Schaltdekoder mit Rückmeldung kann.

Wenn bei Weichen eine Quittungnachricht gesendet wird (so wie Xpressnet das als Broadcast vorsieht), so überlagert diese Quittung zwangsläufig einen S88-Melder, sofern die Weichenadresse ⟨512 ist. Die Folge: bei s88-Betrieb kann man nur Weichenadressen ⟩= 512 vergeben, was unschön ist.

Wenn man hingegen versucht, die Rollen zu tauschen (also Weichen von 0 an adressieren und Feedback mit einem Offset meldet), so muß man das anhand von Byte 1 unterscheiden. Dadurch würden sich folgende Einschränkungen ergeben:

Byte 1:	Bedeutung
0..63	Dieser Bereich wird DCC-Accessory zugeordnet und mit einer Schaltdecoder-Rückmeldung aus dem Bereich 4 (Turnout-Buffer) geladen. Das bedeutet 256 mögliche Weichen
64 .. 127	Dieser Bereich wird als Feedback-Bereich interpretiert (d.h. die Melder beginnen bei 513 bzw. Baustein 65-1). Das bedeutet 512 mögliche Melder

Deshalb werden beim Xpressnet-Protokoll die Bereiche 1, 2 und 3 als Feedback-Broadcasts (TT = 10) beginnend bei 513 gemeldet. Weichen werden mit Broadcast einer Schaltinformation gemeldet, allerdings nur bis Weiche 256. Diese Meldung erfolgt beim Abschalten des Antriebes einer Weiche und Eintreffen der entsprechenden Rückmeldung, wobei die gemeldete (Schalt-)Adresse innerhalb des Bereiches 4 neu beginnend gezählt wird.

2.11.6 Multi-Protokollbetrieb

OpenDCC (in der Version OpenDCC_XP) kann p50xb und Xpressnet parallel bedienen. Hierzu sind die entsprechenden Change-Flags für die beiden Protokolle doppelt ausgeführt, um trotz eines per Push-Dienst am Xpressnet abgesandten Ereignisses dieses auch noch für die event-Abfrage am p50xb vorzuhalten. Die Rückmeldungen werden also verlustfrei parallel in beiden Protokollbereichen verteilt.

3 Softwaredetails

3.1 FAQ und Versionsübersicht

3.1.1 FAQ

Es tauchen immer wieder gleiche Fragen zu OpenDCC[21] auf, diese versuche ich hier zu sammeln und zu beantworten.

- **Was kostet OpenDCC?**
  Mut und Nerven ;-)
  Im Ernst: die Platine kostet 25,90 EUR die Bauteile etwa 30-40 Euro.
  Ich kann leider keine fertigen Zentralen bzw. Bausätze anbieten.
  Norbert Martsch hat sich bereit erklärt, Stückliste und ev. Bausätze zusammen zu stellen. Bei ihm gibt es eine ladbare Bestellliste für reichelt.de.
  Platinen gibt es entweder bei mir oder bei www.railroadcars.de.

- **Wie viele Loks kann ich mit OpenDCC fahren?**
  Der integrierte Booster ist nur für 2-3 Loks ausgelegt, die Steuerlogik für max. 64 gleichzeitig fahrende Loks. Mehr ist für DCC wirklich nicht sinnvoll. Es können aber beliebig viele Loks an der Zentrale aufgerufen werden (das wird dynamisch verwaltet) - fahren tun die letzten 64. *Natürlich ist diese Grenze im Sourcecode änderbar, wer will, kann auch 200 Loks fahren.*

- **OpenDCC wäre ja ganz schön, aber ich wünsche mir einen Handregler!**
  Es gibt eine Erweiterung für Xpressnet-Handregler (z.B. Roco®Multimaus®).
  Dafür wird auch eine bessere CPU benötigt: Statt Atmega32 wird Atmega644P verwendet, wichtig ist der Suffix P, nur diese CPU hat die zweite serielle Schnittstelle. Atmega644-20PU paßt nicht, es muß Atmega644P-20PU sein.

- **Unterstützt OpenDCC Märklin®**[3]
  Nein. Braucht der 'Marktführer' Unterstützung? *(Im Module dccout.c ist allerdings eine bisher nicht getestete Implementierung des MM1 und MM2 Protokolls enthalten, diese wird aber nicht vom ORGANIZER angesteuert.)*

[3] Märklin®ist das eingetragene und geschützte Warenzeichen der Firma Gebr. Märklin & Cie. GmbH, Göppingen , Deutschland.

- **I habe meine s88 Rückmelder angeschlossen, aber die Anzeige flackert. Was ist falsch?**
  Vermutlich ist in der Zentrale nicht die richtige Anzahl an Module je Strang eingestellt. Eine andere Ursache kann ein zu schnelles Lesetiming von OpenDCC sein - auch das ist per CV einstellbar. Alle Anbieter von S88-N Modulen sind verpflichtet, das Timing zu veröffentlichen.

- **Wie kann ich die Zahl der S88-Rückmeldemodule je Strang einstellen?**
  Entweder beim Übersetzen, oder durch Einstellen von Spezial-Optionen (Kapitel 3.3 auf Seite 80). Und es gibt auch spezielle IB-Befehle (Kapitel 3.2 auf Seite 45).

- **Wie kann man Loks programmieren?**
  OpenDCC unterstützt normales Programmieren und Programmieren auf dem Hauptgleis. Zur leichteren Bedienung kann eines der folgenden Programme verwendet werden:

  – P.F.u.Sch.[6] Shareware
  – rocrail[28] Open Source
  – Trainprogrammer[7]
  – Railware[27]

  Ungeeignet ist 'DecoderProgrammer' von Henning Voosen, dieses Programm steuert DCC Loks mit Märklinbefehlen :-(und da ist halt nach 14 Fahrstufen Schluß. (eingeschränkte Funktionalität im IB Mode seitens Decoder-Programmer)
  Auch die Unterstützung für lange Adresse und weitere CVs ist recht rudimentär.

- **Wie kann ich zwischen Programmiergleis und Hauptgleis umschalten?**
  Man kann beide Ausgänge parallel schalten (auf Polarität achten - gleiche Polung verbinden) - danach kann man auf einem Gleis sowohl programmieren als auch probefahren. Hierzu bitte M- und P- verbinden, zwischen M+ und P+ (Pins 2 und 4 von X5) einen Widerstand von 10 Ohm zwischenschalten. Das Gleis wird dann an M+ und M- angeschlossen. (Der Widerstand ist nur nötig, damit die Kurzschlußerkennung nicht verfrüht anspricht). **Achtung:** *nicht mit angeschlossener Anlage machen, das gibt*

dann ziemlich viele gleich programmierte Lokomotiven :-)

- **Warum ist OpenDCC beim Auslesen von CVs so viel schneller als meine bisherige Zentrale?**
  OpenDCC hat einen recht intelligenten Algorithmus zum CV lesen und fährt auch die Programmierbefehle mit dem Minimaltiming der NMRA. Fallweise muß das verlängert werden, s.u.

- **Ich kann die Lok zwar programmieren aber nicht auslesen, was ist falsch?**
  Wenn beim Lesen Timeout kommt, dann erkennt OpenDCC den ACK-Puls von der Lok nicht. Bitte folgendes kontrollieren:

  - Ist die Lok am Programmierausgang angeschlossen (Ausgang P)?
  - Ist die Stromzuführung zur Lok absolut einwandfrei?
    (Grund: Der Quittungspuls für das Lesen wird durch eine Stromerhöhung von mind. 60mA gemacht - die Decoder erzeugen diese Stromerhöhung durch 6ms langes Einschalten des Motors. Dadurch bewegt sich die Lok auch ein wenig und kann fallweise Kontaktprobleme haben.)
  - Ist die Lok mit einem Faulhabermotor ausgerüstet?
    Diese brauchen sehr wenig Strom, so dass die notwendige Stromerhöhung nicht zustande kommt. Hier kann eine Verkleinerung des Widerstandes R22 von 13k auf 12k die Empfindlichkeit erhöhen. Ansonsten muß der Lok ein Lastwiderstand (während des Programmierens) eingebaut werden.
  - Kann der Decoder die gewählte Programmierart?
    Hier kann es speziell bei älteren Decodern Probleme geben, diese können manchmal noch nicht 'DCC byteweise' oder Bitbefehle.
  - Hilft fallweise ein langsameres Lesetiming?
    Mittels zweier Sonderoptionen (18 und 19) (Kapitel 3.3 auf Seite 80) kann die Zugriffsgeschwindigkeit von OpenDCC auf CV's reduziert werden.
  - Ist die u.g. Hardwareänderung durchgeführt?

- **Brauche ich die Optokoppler?**
  Hängt von der Applikation ab: OK1 ist ein allgemeiner Ausgang, wird z.Z. nicht bedient. OK2 ist ein allgemeiner Eingang, wird für die echte Rückmeldung der Weichenlage verwendet. Wenn man das nicht will, kann

man die OC problemlos weglassen.

- **Mit welchen Programmen funktioniert OpenDCC?**
  OpenDCC unterstützt Lenz und Intellibox-Emulation. Dies ist weitgehend kompatibel, kann und will aber nicht alle Fälle abdecken. Z.B. sind Multitraktionsbefehle nicht vorhanden, da bisher kein PC-Progamm diese nutzt.

  Da Softwarehersteller sich am Markt orientieren (da spielen Selbstbausysteme keine Rolle), nehmen sie naturgemäß auch keine Rücksicht auf diese Systeme. Es kann daher sein, dass eine neue Version eines Steuerprogramms ev. neue Befehle verwendet, welche in OpenDCC (noch) nicht implementiert sind. Daher gelten die Tests logischerweise immer nur für den status quo der angesprochenen Software. Aus den erfolgten Tests ist auch keine Präferenz für eine bestimmte Steuersoftware abzuleiten.

  Die Abneigung gegen evtl. missliebige Zentralen geht sogar so weit, dass man sich explizit vorbehält, ohne nachvollziehbare Gründe nicht näher genannte Digitalsysteme zu blockieren (wer kann eigentlich als Käufer so eine AGB guten Gewissens unterschreiben?). Umgekehrt müßte ich mir eigentlich vorbehalten, die Steuerung durch bestimmte Softwaresysteme zu unterbinden.

  Bisher wurde OpenDCC mit folgenden Programmen erfolgreich getestet:

  1. srcpd[26] und spdrs60[18] (Open Source)
  2. TrackOne[13] (Open Source)
  3. Rocrail[28] (Open Source)
  4. Traincontroller[7] (Kommerziell)
  5. railware[27] (Kommerziell)

- **Ich möchte den Code oder Teile davon für ein eigenes Projekt verwenden, was muß ich beachten?**
  OpenDCC[21] ist keine Freeware, sondern unterliegt einer Lizenz (gnu). Diese erlaubt die private Nutzung, bei kommerzieller Nutzung bzw. Weiterentwicklung sind die Lizenzbedingungen zu beachten. Source Code ist entweder auf dieser Seite oder unter sourceforge.net verfügbar. Wenn ein Passwort erforderlich ist, bitte ich dieses unter Angabe des Namens und er Adresse per mail zu erfragen.

- **Ich möchte nachbauen, habe aber ein bestimmtes Bauelement nicht?**

  - Frage zu IC7: Vorgabe: 74AC244N. Diesen hat Reichelt nicht, ist auch 74HC244 möglich? An dieser Stelle geht im Prinzip jeder 244er; AC habe wegen der brachialen Treiberleistung genommen, der HC ist aber auch nicht schlecht; um die Pegel möglichst in der Mitte zu treffen, sollte es ein HC oder AC (kein ACT oder HCT) sein.
  - L6206N (Zweifacher H-Brücken-Treiber im PDIP24 Gehäuse) habe ich nicht. Der L6203 paßt nicht; Zur Not kann man den Booster mit L6203 extra aufbauen und nur die DCC-Leitung vom Prozessor adaptieren. Offenbar ist der L6206 nicht ganz leicht zu bekommen - fallweise per Mail bei mir melden.

- **Ich kann den Bootloader nicht in Ponyprog laden.**
  Zuerst den richtigen Prozessor einstellen, dann das File laden. Der Bootloader beginnt erst bei Hex 7800, es schaut daher im ersten Moment so aus, als wenn nichts passiert wäre - aber einfach mal auf 7800 runterscrollen, dann sieht man den Bootloader.
  Leider unterstützt ponyprog bis jetzt (Sept. 2008) den Atmega644P nicht. Es geht auch mit dem Ponyprog-Seriell Adapter und AVRDUDE (auch unter Linux):

  - Setzen der Fuses mit avrdude:
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U lfuse:w:0xCE:m
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U hfuse:w:0xDC:m
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U efuse:w:0xFD:m

  - Kontrollieren der Fuses mit avrdude:
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U lfuse:r:-:i
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U hfuse:r:-:i
    avrdude -c ponyser -p m644p -P /dev/ttyS0 -U efuse:r:-:i

  - Dann den Bootloader einspielen (nur flash):
    avrdude -c avr911 -p ponyser -P /dev/ttyS0 -U flash:w:bootloader.hex:i

– Optional: OpenDCC einspielen (flash und eeprom):
avrdude -c avr911 -p ponyser -P /dev/ttyS0 -U flash:w:OpenDCC_XP.hex:i
-U eeprom:w:OpenDCC_XP.eep:i

Für das Microsoft Betriebssystem muss man nur /dev/ttyS0 mit comX ersetzen. Alle späteren Updates sind dann über USB mit avrosp.exe möglich (bzw. mit der update_opendcc.bat).

- **Wie kann ich mich aktiv an dem Projekt beteiligen?**
Mithilfe beim Programmieren, Testen und Übersetzen ist gerne gesehen. Es gibt eine (anmeldepflichtige) Yahoo Group, wo man weiteres erfahren kann: http://groups.yahoo.com/group/opendcc/

- **Mir gefällt der S88-Bus nicht, ich hätte lieber RS-Bus oder Loconet.**
S88 ist leicht abspaltbar (die Adapterplatine weglassen und den #define Schalter S88_ENABLED ausschalten). Meine eigenen Rückmelder sind S88, daher habe ich kein anderes Protokoll implementiert - aber ein Entwurf für den RS-Bus geistert schon irgendwo bei mir rum. Zudem ist der S88 gar nicht so schecht, wie er oft gemacht wird: Mit OpenDCC ist er durchaus schnell und die Übertragungprobleme lassen sich lösen.

- **Die Kurzschlußüberwachung schaltet zu schnell ab, was kann ich tun?**
Die Verdrahtung und der Ausgangsfilter belastet die Ausgangsstufe mit einer Kapazität. Daher zieht die Kurzschlußüberwachung prinzipbedingt bei jedem Polaritätswechsel kurz an.
Deshalb sind Sind R19 und R20 auf 2k2 geändert worden und zusätzlich wird in der Software (status.c) ein Filter gerechnet. Dieses Filter könnte man eventuell weniger aggressiv machen:
statt main_short_mean += 4; z.B. main_short_mean += 2; verwenden. SHORT_TURNOFF_TIME ist die Zeitdauer, in der abgeschaltet wird; diese ist z.Z. auf 15ms eingestellt.

- **Ich will mit dem Lenz-Protokoll steuern, geht das?**
Die Entwicklung von OpenDCC startete mit dem Lenz Protokoll; Da aber dort u.a. die CV's und die maximal möglichen Adressen beschränkt waren,

habe ich dann auf IB-Protokoll umgestellt. Alle Tests der letzten Zeit wurden mit dem IB-Protokoll gefahren. Lenz ist nach wie vor vorhanden und kann per Compile-Switch wieder reaktiviert werden. Bitte hierzu per mail nachfragen (V0.14)

- **Die Dioden blinkern, was ist da kaputt?**
  Prog, Stop und Go gleichzeitig schnell blinkend bedeutet, dass die Software keine Daten im EEPROM (oder veraltete Daten) gefunden hat. Es wird nichts gestartet. Abhilfe: Abstecken, roten Knopf drücken, anstecken, Daten einspielen.

- **Wie kann ich ein Logging erzeugen?**
  Bei hartnäckigen Problemen brauche ich zur Analyse ein Logging der Schnittstelle, d.h. eine Mitschrift des Datenverkehrs vom PC zur Zentrale und zurück. Dies wird wie folgt erzeugt:

  - Das entsprechende Programm (portmon.exe) kann von der Microsoft - Webseite geladen werden.
  - Dieses muß man einfach *vor* der Modellbahnsoftware starten.
  - Im Programm muß man folgende Einstellungen vornehmen:
    Auswahl des Ports, welcher überwacht werden soll (kann auch USB sein) Filter setzten (das ist der Trichter) bei Exclude:
    IOCTL_SERIAL_GET_COMMSTATUS mit dazu nehmen (dann wird das das Log kleiner)
    bei Options: Show Time und Show Hex dazu.
  - Jetzt das Zielprogramm starten.

- **Was besser: OpenDCC oder DiCoStation[4]?**
  Das ist natürlich schwer objektiv zu beantworten - jeder Meister lobt seinen Kleister! Einfach zu vergleichen ist die Ausstattung: DiCo hat keinen Nothalttaster, keinen Booster, keinen Programmiergleisanschluß.
  Zur Leistung: ich halte meinen Algorithmus für den Lokstack von DCC für das Optimum an Performance, was aus DCC herauszuholen ist, die Latenzzeiten sind geringer als bei SX! Und diese Performance bleibt auch bei vielen Loks und 128 Fahrstufen erhalten! Beim Protokoll P50x verursacht die die DiCo erhebliche CPU-Last auf dem Host, OpenDCC nicht.

[4] DiCoStation ist das eingetragene und geschützte Warenzeichen von Littfinski Daten Technik[22]

Die S88-Implementierung von OpenDCC ist deutlich flotter und störsicherer unterwegs. Die DiCo kann Märklin Motorola, was ich nicht einbauen wollte.

3.1.2 Change Log bzw. Release Notes

Hier findet sich die Änderungsgeschichte. OpenDCC ist ein nicht kommerzielles Projekt (ich mache das in meiner Freizeit) - daher bitte ich um Nachsicht, wenn noch nicht alle Facetten getestet sind und vielleicht der eine oder andere Bug enthalten ist. Aber ich hoffe auf Unterstützung beim Test und auf möglichst konstruktive Kritik. Danke all jenen, die bisher zum Projekt beigetragen haben.

- Change Log bzw. Release Notes (Xpressnetversion, Atmega644P)

  - OpenDCC_XP_V0.20beta
    - 06.09.2008: Version 0.20.5 (release candidate) Neu dazu: Programmierbefehle, railcom
    - 28.08.2008: Zweite beta: neu dazu LOCO DATA BASE, bisher getestet: Einträge hinzufügen, Ausgeben, Übergabe an multiMaus.
    - 23.08.2008 Erste Release, bisher getestet: Fahren, Schalten, Mehrfachhandregler, PC-Übergabe (beide Richtungen).
      offen: Programmierinterface

- Change Log bzw. Release Notes (Normalversion, Atmega32)

  - V0.16beta, 20.08.2008
    - Bugfix: Der Meldewert beim IB-Parser für Zubehörbefehle zeigt fäschlicherweise 'Queue fast voll' an.
    - Bugfix: Default Baudrate konnte nicht umgestellt werden.

  - V0.15beta , 12.08.2008

    Bitte beim Update auf V0.15beta sowohl Flash und EEPROM neu einspielen.
    - Die jeweiligen Wiederholungen für Geschwindigkeit, Function und PoM sind per CV einstellbar. Die default-Werte sind allerdings sinnvoll und erprobt, da braucht man nichts verstellen.

  - Die defaultmäßig verwendete Fahrstufenzahl (28) ist per CV veränderbar.
  - Externer Nothalteingang (Konfigurierbar per CV36).
  - Die Abschaltzeit bei Kurzschluß jetzt einfach per CV änderbar. (Voreinstellung: 15ms, 40ms für das Programmiergleis)
  - Es ist eine CV als Seriennummer beschreibbar - damit lassen sich verschiedene Boxen leichter indentifizieren.
  - Das Timing am S88 ist per CV programmierbar. Zusätzlich wird bei aktivierten Programmiergleis das s88-Lesen abgeschaltet, um keine Unterbrechungen im DCC Signal zu riskieren. Des weiteren habe ich die S88-Task auf Bitebene runtergebrochen - somit kann auch bei langer Einstellung der S88-CV kein Timingproblem auftreten.
  - Es gibt eine neue Configdatei für TC.
  - Bugfix: Bei heftigem Datenverkehr und vielen Lokkommandos konnte es passieren, dass ein Richtungswechsel nicht rechtzeitig die bisherigen Richtungsinformation ersetzt hat. Die entsprechende Lok fährt dadurch in falscher Richtung los. Jetzt wird ein Richtungswechsel mit hoher Priorität behandelt und kann nicht mehr von normalen Geschwindigkeitskommandos überholt werden.
  - Bugfix: Adressierungsfehler bei Rückmeldebits und aktivierter Weichenlagemeldung (nur bei entsprechend gesetztem compileswitch). Zusätzlich gibt zwei weitere CVs, mit denen man die Anzahl der rückgemeldeten Weichen einstellen kann.
  - 17.04.08: Bugfix: Baudrate 57600 und 115200 sind falsch initialisiert. das ist behoben, aber noch nicht im download.
  - 18.07.08: Bugfix: HEX Eingabe im p50xa-Befehlen hatte Fehler.
  - 10.08.08: In Vorbereitung der Xpressnetschnittstelle wurde Optimierungen und Codeänderungen am Organizer vorgenommen. Die S88 Task ist auf Timer umgestellt.
  - PT-Event wird jetzt im Falle eines Abbruchs vom Host erzeugt.

– Neue Platinenversion V1.4, lieferbar ab Jan 2008
  Bei der Nachbesellung von Platinen habe ich die Gelegenheit benutzt, einige kleinere Änderungen einfließen zu lassen. Änderungen von V1.3 auf V1.4:

* Die RJ45 Anschlüsse haben jetzt eine Pinbelegung gemäß dem neuen Standard S88-N (siehe Kapitel 7.2 auf Seite 166).
* Dieser Standard unterstützt auch die Übertragung von RAIL-DATA - dafür wird in der V1.4 das DCC Ausgangssignal mittels eines Single Gate Buffers (74AHC1G125) zu den S88 Leitungen übertragen. Für den normalen S88-Betrieb braucht man das nicht und kann es einfach weglassen.
* Um zukünftig auch mal CVs über diesen Weg lesen zu können, kann die Reset-Leitung tristate geschaltet und zurückgelesen werden.
* Das Umschalten zwischen USB und RS232 erfolgt jetzt über einfache Jumper (JP5 und JP6) auf Bergstiften.
* SJ4 ist jetzt auf der Bauteilseite
* Viele Texte und Markierungen im Kupfer hinzugefügt, um den Nachbau zu erleichtern.

– V0.14 , 20.10.2007
Bitte beim Update auf V0.14 sowohl Flash und EEPROM neu einspielen (auch für bisherige 0.14beta-Inhaber).

* neu: echte Weichenrückmeldung (Kapitel 3.5 auf Seite 93)
* Programmierbefehle funktionieren unter Lenz-Emulation - danke an Rainer für die Logs!
* Erweiterung um P50Xa-Programmierbefehle und Stop und Go (0x60 bzw. 0x61) der 6050 Syntax. Diese Befehle werden u.a. auch von WinDigipet gebraucht.
* Nach dem Programmierende wird wieder der Zustand vor dem Programmieren einschaltet - war vorher 'running', ist es danach auch wieder.
* Der default-Startmode im Intellibox-Mode ist jetzt P50/P50X-Mixed (war vorher P50X-only), damit funktioniert auch Rocrail[28]. Bei TC hat es keine Auswirkung.
* Bessere Aufteilung des Sourcecode in Module, um zukünftige andere Hardwareplattformen besser integrieren zu können.
* Beim Befehl XZzA1 wurde kein String zurückgegeben, das ist jetzt korrigiert.

* Beim Programmieren wird das Einlesen der ACK-Leitung mit einer Filterfunktion bewertet, damit können 'Problemloks' besser behandelt werden. Während OpenDCC auf den ACK wartet, wird die gelbe LED (PROG) kurz ausgeschaltet - damit sieht man die Leseaktivität recht gut.

* Der Programmieralgorithmus kann im Timing per Konfiguration langsamer gemacht werden:
  - Die Zahl der Resetpakete vor einem Lesebefehl kann erhöht werden.
  - Die Zahl der Lesekommandos kann erhöht werden, damit hat der Decoder mehr Zeit für die Antwort. (siehe Sonderoptionen (Kapitel 3.3 auf Seite 80) 18 und 19)

* Einfachere Konfiguration durch Programmierbefehle (Kapitel 3.4 auf Seite 90): Wenn beim Einschalten die GO-Taste gedrückt ist, dann werden die Programmierbefehle auf die internen Konfigurationsvariablen umgeleitet. Die Parameter der Zentrale sind dann über einen Programmer (z.B. Trainprogrammer[7]) les- und schreibbar. Natürlich kann man dann keine normalen Loks mehr programmieren. Für Trainprogrammer gibt es auch ein yrc-file: OpenDCC.yrc

– V0.13 , 15.03.2007
  Alle Optimierungen der V0.13 gelten nur für den IB-Mode!

* Bugfix im Programmer bei Intellibox-Emulation.
  Hintergrund: der Programmer war in den älteren Versionen 'busy waiting' implementiert und wurde wegen des IB-Parsers in den normalen Queuemanager einordnet ('multi tasking'). Leider hat der Queuemanager den bereits vorgegebenen Repeatcount der Programmierbefehle mit 1 überschrieben, so dass nicht mehr die normgerechte Anzahl an page-presets o.ä. rausgegangen ist.

* Der ACK-Puls bei Programmieren wird zusätzlich mit einer Software-Entprellschleife von 500 µs gewertet. Bitte auch die Hardwareänderungen (Kapitel 4.1 auf Seite 113) bachten: C44 - 10nF, R19 - 2k2, R20 - 2k2

* Neu dazu: Programmieren auf dem Hauptgleis - POM *Hinweis: das geht logischerweise nur write-only.*

* Falls OpenDCC im Programmiermodus steht, werden auch die Lok-Kommandos, Accessory und POM auf das Programmiergleis

ausgegeben, aber nur, wenn nicht gerade eine Programmieraktion läuft. Diese Kommandos werden nicht zyklisch wiederholt oder im Lokspeicher abgelegt.

* Beschleunigung des DCC Direktmodes durch vorhergehenden Test auf Bit-Fähigkeit. Diese Decoderfähigkeit merkt sich OpenDCC für eine kurze Zeit (#define TIME_REMEMBER_BIT_OP). Damit dauert dann das Auslesen einer kompletten Geschwindigkeitskennlinie 20s, obwohl die Vorgaben der NMRA[24] im Gegensatz zu einer käuflichen Zentrale eingehalten werden.
* Zurückgemeldete Fehlercodes beim Programmieren sind jetzt konform zur IB.

– V0.12, 05.02.2007

* EEPROM-Routinen und mehrere Sonderoptionen der IB implementiert.
Hintergrund: Railware[27] fragt eine Menge von IB-SO's ab - diese werden bestmöglichst beantwortet.
* Compiler-Umstellung auf gcc[1] 4.1.1 und AVR-Studio 4.13[5] (wichtig).

– V0.11, 28.01.2007

* Bugfix im Zusammenspiel Intellibox®und TC.
Hintergrund: TC initialisiert mit 2 Stopbits. Wenn OpenDCC mit einem Stopbit sendet, kann es bei USB-to-Serial Konvertern Probleme geben
* DCC Accessory repeat count ist im EEPROM einstellbar.
* Bei der Lenz-Emulation werden die Accessory Commands invertiert. Hintergrund: hier unterscheiden sich IB und Lenz in Ihrer Interpretation von DCC (!)
* Bugfix im Zusammenspiel mit DMX
* **Hardwareänderung:** um allzu hektisches Abschalten der Ausgangsstufe bei kapazitiven Lasten zu vermeiden, muß die Erholzeit der Ausgangsstufe verkleinert werden: R19 und R20 ändert sich von 10k auf 2k2.

– V0.10, 22.01.2007

* OpenDCC hat heute mit der Intellibox-Emulation Verbindung zum Traincontroller aufgenommen. Lok ist gefahren, Funktionen wurden übermittelt und Weichen konnte ich schalten!
Stop-Go wird in beide Richtungen übermittelt. Das etwas verzwickte

Initialisierungsschema "BABI" inkl. der Abfrage vom Sonderoptionen (wie z.B. Baudrate) funktioniert auch.

3.1.3 geplante Erweiterungen

Das sind nur Idee, ob ich es mache, steht in den Sternen!

- Konfiguration von OpenDCC über einen virtuellen Lokdecoder - ab V0.14 enthalten
- Rückmeldung der Weichenlage mittels rückmeldefähiger Decoder und virtueller Melder - ab V0.14 enthalten.
- POM - Programming on the main auch im ASCII Mode
- Handregler (anstelle DMX)

3.2 Befehle im Intellibox-Mode, P50X

3.2.1 Überblick

Beim Projekt OpenDCC kann das zum Host hin emulierte Schittstellenprotokoll verschiedene Zentralen emulieren. Nachfolgend ist die Kommandoübersicht für den **Intellibox-Mode®** aufgelistet. Entdeckte Fehler oder Unklarheiten bitte ich per mail mitzuteilen. Verwendete allgemein übliche Bezeichungen für andere Zentralen sind fallweise registrierte Marken der jeweiligen Eigner.

3.2.2 Allgemeine Eigenschaften

- Schnittstelle:
  OpenDCC arbeitet mit einer Default Baudrate von 19200. Einstellbar sind jedoch auch 2400, 4800, 9600, 19200 und 38400 Baud. Das Datenformat ist immer 8 Datenbits, kein Parity, 2 Stopbit (8N2), wobei es beim Empfang egal ist, ob 1 oder 2 Stopbits empfangen werden. Beim Senden werden immer 2 Stopbits gesendet, damit unter Traincontroller auch USB-to-Serial Konverter funktionieren.
  Bei der Kommunikation über USB wird eine serielle Schnittstelle emuliert, hier liegt die empfohlene Einstellung bei 38400 Baud. OpenDCC unterstützt die automatische Baudratenerkennung BABI wie die Intellibox. Die Implementation ist etwas anders, funktionierte jedoch bei einem Test mit Traincontroller.

HINWEIS: Nach Neustart von OpenDCC (z.B. durch Aus- und Einschalten) oder abziehen des USB-Kabels ist die benutzte serielle Schnittstelle über USB nicht mehr gültig, da sich OpenDCC über USB nach dem Einschalten neu am System anmeldet. In diesem Fall ist das Filehandle (des Steuerprogramms) für die geöffnete Schnittstelle unbrauchbar (es können keine Zeichen mehr versandt werden und es wird natürlich nichts mehr empfangen). Das kann je nach Qualität der Hostsoftware zum "Hängenbleiben" des Steuerprogramms führen.

Es wird voller Duplexbetrieb mit Hardwarehandshake unterstützt, die Transmit- und Receive-Fifos sind 64 Bytes tief, 10 Byte vor dem Vollwerden wird CTS = High zurückgemeldet.

- Protokoll:
  Bei der Intellibox gibt es drei Protokolle:

  - Märklin 6050 binäre Interfacebefehle
    (bei der Intellibox [IB] P50-Kommandos genannt).
  - ASCII Kommando-Erweiterung
    (bei der IB P50Xa-Kommandos genannt).
  - Binär Kommando-Erweiterung
    (bei der IB P50Xb-Kommandos genannt).

  Wenn im P50-Mode das erste Zeichen ein X ist, dann wird auf die erweiterte Kommandoebene umgeschaltet; Ist das zweite Zeichen ein ASCII Zeichen (¡0x80), so wird die ASCII Erweiterung angesprochen, ansonsten die Binärerweiterung.

  OpenDCC unterstützt den P50-Mode gar nicht (Ausnahme: 0xc4 für Autobaudumschaltung), den ASCII-Mode und den Binärmode weitgehend. Im ASCII Mode wird jede Antwort mit der Zeichenfolge + ']' abgeschlossen.

- Lokprotokoll, Lokstack:
  OpenDCC arbeitet nicht mit einen Lokstack oder Slots, welche zyklisch arbgearbeitet wird, sondern mit dynamisch priorisierten Listen, welche von einer Organisationseinheit gefüllt werden. Dadurch wird eine extrem schnelle Reaktion auf Fahrbefehle vom PC erreicht. Diese Organisationseinheit kann 64 gleichzeitig *fahrende* Loks verwalten. Loks, welche längere Zeit nicht mehr bewegt wurden, werden beim Aufrufen weiterer Loks automatisch von der Organisationseinheit nicht mehr betrachtet, es ist kein An- oder Abmelden erforderlich. Ein weiterer Vorteil dieser dynamischen Listen ist

gleichbleibende Performance auch bei längerem Fahrbetrieb.
Diese Grenze von 64 fahrenden Loks kann durch eine entsprechende #define-Einstellungen beim Übersetzen auch anders liegen. Allerdings macht es kaum Sinn, mit DCC mehr als 40 Lok gleichzeitig fahren zu lassen. Da stößt das Timing am Gleis an die Grenzen.

OpenDCC erlaubt Lok-Adressen von 1 .. 10239. (gesamter DCC Adressraum). Der Bereich von 10239 bis 16384 bleibt reserviert. Die Adresse Null ist grundsätzlich ungültig und wird nicht in einen entsprechenden Broadcast umgesetzt.

Das Lokformat von OpenDCC wird (per Compileeinstellung) auf eine Voreinstellung festgelegt. Alle Loks werden in diesem Format (z.B. DCC28) angesprochen. Da das orginale Intellibox-Protokoll keinen Befehl zum Festlegen des Formats kennt, wurde ein neues Kommando Lok_Cfg_Set (LS) eingeführt, das es erlaubt, Lokomotiven mit einem anderen Format anzusprechen.
Per Lok_Cfg_Set definierte Ausnahmen vom Defaultprotokoll werden permanent im EEPROM gespeichert; es gibt 64 Speicherplätze für Ausnahmen. Mit Hilfe des Programms OpenDCC_configtool lassen sich diese Listen leicht verwalten.

Geschwindigkeitseinstellung:
Die Geschwindigkeit am Interface ist unabhängig vom Format der Lok 0 bis 127 und wird intern auf die jeweils verfügbaren Geschwindigkeitsstufen umgerechnet. Die Geschwindigkeitsstufe 1 bedeutet dabei immer Not-Halt (Emergency-Stop).

Unterstützte Funktionen:
OpenDCC kann die Funktionen F1 bis F12, sowie F0 (bzw. FL / Licht).

- Rückmelden (S88):
  OpenDCC macht grundsätzlich s88-Autoreset, d.h. das Auslesen eines s88-Moduls liefert den aufsummierten Zustand (Oder-Verknüpfung) seit dem letzten Reset. Diese Bits werden nach dem Auslesen wieder auf Null gesetzt.

  Über das S88 Interface können auch die Weichenrückmeldungen (Kapitel 3.5 auf Seite 93) eingelesen werden, dies muß entsprechend mit Spezial-Optionen (Kapitel 3.3 auf Seite 80) eingestellt werden.

3.2.3 P50-Kommandos (Märklin 6050)

OpenDCC unterstützt dieses Format nicht - nur die beiden Kommandos 'x' (oder 'X') und 0xC4 (liefert als Antwort immer 0x00 bzw. 0x00 0x00, je nach Mode) werden unterstützt, damit die Baudratenerkennung funktioniert und auf einen der P50X Befehlsätze umgeschaltet werden kann.
Auch der Märklin MM1 und MM2-Mode wird bis auf weiteres nicht freigegeben.

3.2.4 P50Xa-Kommandos (ASCII-Commands):

Allgemeine Regeln:
P50Xa Befehle beginnen immer mit einem X (es sei denn, man ist bereits permanent im P50X-Modus).
Ein Befehl darf bis zu 63 Zeichen lang sein und wird mit (=0x0D) abgeschlossen.
Die Antwort wird immer mit "]" abgeschlossen.
Im Gegensatz zur IB kann nicht mit "" ein Zeichen gelöscht werden, welches bereits an OpenDCC abgesendet wurde.

Im folgenden sind optionale Parameter mit [..] eingeklammert. Wird ein Befehl ohne Parameter übergeben, so wird dieser Befehl als Abfragebefehl ausgeführt.

- **?** oder **H**: Hilfe
  Aktion: keine
  Antwort: "Help cmds: HL, HF, HT"

- **HF**: Hilfe für Funktionskommandos
  Aktion: keine
  Antwort: "F Lok#, [F1], [F2], [F3], [F4], [F5], [F6], [F7], [F8]"

- **HL**: Hilfe für Lok-Befehle
  Aktion: keine
  Antwort: "L Lok#, [Speed], [FL], [Dir], [F1], [F2], [F3], [F4]"

- **HT**: Hilfe für Turnout-Commands (Weichen)
  Aktion: keine
  Antwort: "T Trnt#, [Color], [Status]"

- **.** oder **STOP**: Stop
  Aktion: Führt einen STOP aus (selbe Aktion wie zweimaliges Drücken der roten STOP-Taste). Die Gleissignale werden abgeschaltet.
  Antwort: "Pwr off"

- **!** oder **GO**: Go
  Aktion: Führt einen GO aus (selbe Aktion wie Drücken der grünen GO-Taste). Die Gleissignale werden eingeschaltet.
  Antwort: "Pwr on"

- **HALT**: Halt
  Aktion: Die Funktion hält alle Loks an (per Emergency-Stop), schaltet die Gleis- signale jedoch nicht ab. Weichen können nach wie vor geschaltet werden.
  Antwort: "Halted!"

- **L**: Lok Befehl
  Syntax: L Lok#, [Speed], [FL], [Dir], [F1], [F2], [F3], [F4]
  Parameter:

Lok#:	Lokadresse (1 .. 10239)
Speed:	Geschwindigkeit (0 .. 127: 0 = Stop, 1 = Em.Stop)
FL:	Front-Light (0 = aus, 1 = ein)
Dir:	Direction (0 oder r = rückwärts, 1 oder f = vorwärts)
Fn:	Funktionsausgang n (0 = aus, 1 = ein)

  Unabhängig vom Lokformat wird die Geschwindigkeit immer im Bereich von 0 bis 127 angegeben. OpenDCC rechnet diese Geschwindigkeit intern auf die jeweiligen Fahrstufen der Lok um. Es gilt folgender Algorithmus (dieser ist indentisch zu Tams EasyControl und Intellibox; es wird jeweils als integer gerechnet):

  1. Geschwindigkeit 0 bleibt 0.
  2. DCC, 14 Fahrstufen: V = (Speed - 2) / 9 + 1;
  3. DCC, 28 Fahrstufen: V = (Speed - 2) * 2 / 9 + 1;
  4. DCC, 128 Fahrstufen: V = (Speed - 1);

  27 Fahrstufen werden nicht unterstützt. Die höchste Fahrstufe am Gleis liegt damit je nach Protokoll bei folgender Eingabe an:

Lokformat	Max bei
DCC14	119
DCC28	124
DCC128	127

- **LC**: Lok Protokoll Konfiguration ausgeben
  Syntax: LC Lok#
  Parameter:

Lok#:	Lokadresse (1 .. 10239)
Antwort:	"DCC" und Zahl der Fahrstufen

  War die Lok bisher noch nie benutzt und ist sie auch nicht als Ausnahme von der Voreinstellung in OpenDCC hinterlegt, so wird das voreingestellte Lokformat zurückgemeldet.

- **LS**: Lok Protokoll einstellen (neu in OpenDCC!)
  Syntax: LS Lok#, [Format], [Steps]
  Parameter:

Lok#:	Lokadresse (1 .. 10239)
Format:	MM1, MM2 or DCC, codiert als 0,1,2; bei OpenDCC ist nur DCC erlaubt.
Steps:	14, 28, 126

  Aktion: Falls Format von der Voreinstellung abweicht, wird dies Lok als Ausnahme gespeichert.

- **F**: Function
  Syntax: F Lok#, [F1], [F2], [F3], [F4], [F5], [F6], [F7], [F8]
  Parameter:

Lok#:	Lokadresse (1 .. 10239)
Fn:	Funktionsausgang n (0 = aus, 1 = ein)

  Aktion: Wenn mindestens ein Parameter vorhanden ist, so werden die Funktionsbits entsprechend gesetzt.
  Antwort: Ausgabe der Funktionsbits.

- **FX**: Function (eXtended)
  Syntax: FX Lok#, [F9], [F10], [F11], [F12]
  Parameter:

Lok#:	Lokadresse (1 .. 10239)
Fn:	Funktionsausgang n (0 = aus, 1 = ein)

Aktion: Wenn mindestens ein Parameter vorhanden ist, so werden die Funktionsbits entsprechend gesetzt.
Antwort: Ausgabe der Funktionsbits.

- **LOCADD**: Lok zu Datenbank hinzufügen
  Syntax: LOCADD Lok#, SPEEDSTEPS, FORMAT, NAME
  Parameter:

  Lok#: Lokadresse (1 .. 10239)
  SPEEDSTEPS: Anzahl der Fahrstufen, mögliche Auswahl: 14, 28, 126
  FORMAT: Lokformat, mögliche Auswahl: hier nur DCC
  NAME: max. 6 Zeichen (ASCII)

  Aktion: Die Lok wird in die interne Datenbank aufgenommen und permanent gespeichert. Diese Datenbank ist ***nicht*** der Lokstack - d.h. es können auch Loks gefahren werden, die nicht in der Datenbank enthalten sind. Umgekehrt sind Loks nicht in der Gleisausgabe enthalten, wenn Sie zwar in der Datenbank enthalten sind, aber noch nicht aufgerufen wurden. Die Namen werden in Großbuchstaben umgewandelt.
  Wenn eine Lok mit gleicher Adresse in der Datenbank bereits vorhanden ist, dann wird diese ersetzt.
  Nur verfügbar bei OpenDCC_XP

- **LOCDUMP**: Lok-Datenbank ausgeben
  Syntax: LOCDUMP

  Aktion: Es wird eine Liste mit einem Datensatz pro Zeile ausgegeben. Der Aufbau jeder Zeile ist wie folgt:

  Lok#, SPEEDSTEPS, FORMAT, NAME

  SPEEDSTEPS, FORMAT, NAME sind wie bei LOCADD formatiert. Die Liste wird mit einer Zeile "*END*" abgeschlossen.
  Nur verfügbar bei OpenDCC_XP

- **LOCXMT**: Lok-Datenbank an eine angeschlossene multiMaus®ausgeben
  Syntax: LOCXMT

  Aktion: Es wird die Lokdatenbank auf dem Xpressnet ausgegeben. Die Ausgabe erfolgt jeweils zweimal je Lok in Abständen von je 100ms. Je nach Umfang der Lokdatenbank kann dies also etwas dauern.
  Nur verfügbar bei OpenDCC_XP

- **LOCCLEAR**: Lok-Datenbank komplett löschen
  Syntax: LOCCLEAR

Aktion: Antwort: 'ok'; Die Datenbank wird gelöscht - und weg ist sie! (keine Nachfrage) Ein PC-Programm (Configtool) kann mit LOCDUMP alle Loks einlesen (oder bei Null beginnen) und eine interne Liste von Loks aufbauen (z.B. durch den Anwender editieren lassen). Anschließend löscht es die Lok-DB mit LOCCLEAR und sendet eine komplett neue Liste mit LOCADD an OpenDCC. Nach dem Übertragen der List kann diese dann mit LOCXMT an die Handregler verteilt werden.
Nur verfügbar bei OpenDCC_XP

- **T**: Turnout (Weiche oder Signalschaltbefehl)
  Syntax: T {Trnt#, [Color], [Status]}
  Parameter:
  Trnt#: Weichen-Nummer (1 .. 2040)
  Color: 0 oder 'r' = Rot / Abzweig,
  1 oder 'g' = Grün / gerade aus
  Status: 1 = ein,
  0 oder nicht angegeben = aus

  Aktion: OpenDCC sendet z.Z. sowohl den Status on als auch den Status off Befehl auf den Ausgang.

  Ich will das in Zukunft im Zuge der Weichenrückmeldung aber wie folgt abändern: Bei Status on wird der Befehl durchgereicht; bei Status Off wird auch der Befehl durchgereicht, allerdings wird danach der Acknowledge-Eingang abgefragt. Ein rückmeldefähiger Weichendecoder quittiert damit den vorangegangen Schaltbefehl und zeigt durch den ACK-Puls an, dass die Weiche in korrekter Lage liegt. Fehlt dieser Rückmeldepuls, so wird entweder ein Turnout-Event oder eine S88-Event generiert - genaueres ist noch mit Herrn Freiwald zu klären.
  Aktion: Schaltet die Weiche entsprechend oder gibt den Status der Weiche aus.
  Beispiel: "T 5 g 0"

- **RC**: Railcom
  Syntax: RC {[Mode]} Parameter:

  ohne Parameter: der aktuelle Zustand wird ausgegeben.
  0: Die Erzeugung von cutouts ist abgeschaltet.
  1: Im DCC Strom wird nach jedem Befehl für die Länge von 434us ein Cutout eingefügt.

  Antwort: "BiDi on" oder "BiDi off"

- **Y**: Status
  Aktion: Gibt den momentanen Systemstatus aus.
  Antwort: "Pwr off", "Halted!", "Pwr on", "DCC program" oder "RESET"

- **V**: Version
  Syntax: V
  Gibt die Versionsnummer von OpenDCC aus: "OpenDCC V0.12"

- **MT**: Magnetartikel Timer
  In OpenDCC nicht implementiert

- **SR**: s88-Autoreset
  Syntax: SR [Flag]
  Flag: 0 (aus) oder 1 (ein)
  In OpenDCC nicht implementiert, es wird immer die automatische Einstellung verwendet.

- **SS**: s88-Modul auslesen Syntax: SS Modulnummer
  Modulnummer: 1..32
  In OpenDCC nicht implementiert, alle Rückmeldungen laufen über XEVSens.

- **SE**: Anzahl s88-Module definieren
  Syntax: SE [AnzModul]
  AnzModul: Anzahl der s88-Halbmodule (0..64)
  In OpenDCC nicht implementiert, da drei Strings vorhanden sind. Diese werden akteulle über EEPROM Konstanten eingestellt.
  Die Anzahl der automatisch einzulesenden Module ist die angegebene Anzahl dividiert durch 2 (die IB rechnet hier in 'Halbmodulen', entsprechend Bytes). Ungerade Anzahlen werden nach oben gerundet. Eine Angabe von 0 als Anzahl zu lesender Halbmodule führt zum Abschalten des s88-Bus (es werden keine Daten eingelesen). Antwort: tbd.

- **ZZA**: Umschalten zwischen P50X-Only und P50/P50X-Mixed Modus
  Syntax: ZZA [Wert]
  Wert: 0 (mixed mode) oder 1 (P50X-only mode)

  Schaltet OpenDCC in den P50X-only mode (nur P50X Kommandos erlaubt) oder wieder zurück in den mixed mode (P50 und P50X Kommandos erlaubt).
  Antwortstrings:
  ZZA0: 'Mixed P50/P50X mode'
  ZZA1: 'P50X only mode (P50 is disabled)'

  Hinweis 1: Dieser Befehl ist im Gegensatz zur IB nicht case-sensitiv.
  Hinweis 2: Der Startmode kann mit einer Compileoption festgelegt werden. (default ab 0.14: Mixed Mode)

- **@@**: Kaltstart bzw. **@**: Warmstart
  Syntax: @@
  Sowas brauchen doch nur reset- und absturzanfällige Kisten - darum gibt es dieses Kommando in OpenDCC nicht. :-)

- **B**: Baudrate
  Syntax: B [Baud]
  Parameter: Baud: Baudrate (2400, 4800, 9600, 19200, 38400, 57600)
  Die Baudrate wird auf den angegebenen Wert gesetzt, wobei die Antwort dieses Befehls noch vollständig mit der alten Baudrate gesendet wird, sofort danach wird umgeschaltet. Sollte ein falscher Wert für Baud angegeben werden, so wird DEFAULT_BAUD (normalerweise 19200) eingestellt. Fehlt der Parameter, so wird die aktuelle Baudrate zurückgemeldet.

- **SO**: EEPROM-Variable auslesen oder schreiben
  Syntax: SO Adresse [Wert]
  Adresse: Die Adresse einer EEPROM-Variablen. Die IB ermöglicht Adressen von 0..999. OpenDCC hat eine andere Belegung der Adressen (siehe Sourcecode). Ein Ausnahme gilt für SO 1: hier wird bei einer Abfrage der interne ENUM für die BAUDRATE (welcher normalerweise der Lenz-Zählweise entspricht) auf die Intellibox Syntax umgesetzt. Damit funktionieren dann auch das pseudointelligente Verfahren von TC, bei bereits bestehender Verbindung nochmal anhand von SO 1 die Baudrate umzustellen.
  Zur weitestgehenden Kompatibilität mit railware sind die von Railware nachgefragten Sonderoptionen (Kapitel 3.3 auf Seite 80) mit möglichst sinnvollen Werten vorbelegt.

- **PT**: Einschalten oder Ausschalten Programmier-Modus
  Syntax: PT [0|1]
  0 schaltet Programmiermode aus, 1 schaltet ein. kein Parameter: aktueller Zustand wird angezeigt.

- **PTRD**: Lesen DCC-CV per Bytebefehle
  Syntax: PTRD [CV]
  Parameter: CV: 1..1024
  Antwort: Okay, gefolgt vom Inhalt der CV oder failed

- **PTWD**: Schreiben DCC-CV per Bytebefehle
  Syntax: PTRD [CV, Byte]
  Parameter: CV: 1..1024, Byte: das wird geschrieben
  Antwort: Okay oder failed

- **PA**: Schreiben DCC-CV per Programming on the main, für Accessory Decoder
  Syntax: PA [CV, Byte]

Parameter: CV: 1..1024, Byte: das wird geschrieben
Antwort: Okay oder failed

- **PAR**: Lesen DCC-CV per Programming on the main und BiDi, für Accessory Decoder
  Syntax: PAR [CV]
  Parameter: CV: 1..1024
  Antwort: gelesenes Byte (0xAB) oder failed
  Das ist eine geplante Erweiterung, diese ist noch nicht implementiert!

- **PD**: Schreiben DCC-CV per Programming on the main, für Lok Decoder
  Syntax: PD [Addr, CV, Byte]
  Parameter: Addr 1..10239 Lokadresse

  CV: 1..1024
  Byte: Dateninhalt Antwort: Okay oder failed

- **PDR**: Lesen DCC-CV per Programming on the main und BiDi, für Lokomotiv Decoder
  Syntax: PDR [CV]
  Parameter: CV: 1..1024
  Antwort: gelesenes Byte (0xAB) oder failed
  Das ist eine geplante Erweiterung, diese ist noch nicht implementiert!

3.2.5 P50Xb-Kommandos (Binary-Commands):

Wenn nach dem einleitenden 'X' ein Zeichen mit einer Binärcodierung >0x80 kommt, so wird das Binärprotokoll angesprochen. Wenn OpenDCC auf *nur* P50Xb eingestellt ist, so ist das führende 'X' wegzulassen.
Das erste Zeichen (Byte) ist das Kommandobyte und legt fest, wieviele Parameter-Bytes folgen. Diese Kommando wird in jedem Fall sofort quittiert (Returncode). Wenn mit dem Kommando eine längere Aktion verbunden war (z.B. ein Programmierbefehl) so ist das Ergebnis dieses Befehls mit einem Abfragebefehl zu holen.

Die jeweilige Antwort kann auch eine längere Liste (fallweise verkettet) von einzelnen Bytes sein. Hierbei verwendet das IB-Protokoll unterschiedliche Techniken:

1. Mit einem MSB (Erweiterungsbit) wird eine weiteres Antwort-Byte angekündigt.
2. Die erste Antwort enhält eine Kodierung, ob weitere Antworten folgen. Diese Antworten können auch jeweils Pakete von Bytes sein.

3. Die Antwort beginnt mit der Zahl, wie viele Antwortpakete übermittelt werden.

Offenbar ist das IB-Protokoll ziemlich historisch gewachsen und von verschiedenen Entwicklern im Freestyle entworfen worden :-(

Kommandoliste:

- **XLok (0x80)** - Länge = 1+4 bytes

```
Befehlsbytes:
        0: 0x80 XLok
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im Bereich
           1 .. 10239)
        3: Speed (0 .. 127). (unabhängig von Lokformat)
        4: Funktion- und Statusbits:
   bit#   7     6     5     4     3     2     1     0
       +-----+-----+-----+-----+-----+-----+-----+-----+
       |Chg-F| Forc| Dir | F0  | F4  | F3  | F2  | F1  |
       +-----+-----+-----+-----+-----+-----+-----+-----+
           Chg-F:     wenn 1: die Bits F4 .. F1 als
                      Funktionsbits übernehmen, bei
                      0 ignorieren.
           Force:     wird von OpenDCC ignoriert, ein
                      Lokcommand gilt immer.
           Dir:       Fahrtrichtung:
                      1 = vorwärts
                      0 = rückwärts
           F0:        Frontlicht (FL):
                      1 = an
                      0 = aus
           F4..F1:    Status der Funktionen F1 .. F4
                      (nur wenn Bit F-Valid gesetzt)

Antwort:   1 Byte:  0 oder Error-Code

        Mögliche Error-Codes:
        OK        - OK, Befehl ausgeführt
        XBADPRM   - Lokadresse außerhalb des Bereichs
                    (1 .. 10239)
        XNOSLOT   - Z.Zt. kann die Lok nicht in die
                    Queues aufgenommen werden
```

```
                (zu viele Befehle) Befehl später
                wiederholen.
        XLKHALT   - OpenDCC ist im HALT-Modus. Der
                Befehl wurde akzeptiert (bezüglich
                der Funktionen und der
                Fahrtrichtung) aber mit V=0 in die
                Buffer aufgenommen.
        XLKPOFF   - OpenDCC ist im STOP-Modus. Der Be-
                fehl wurde akzeptiert (bezüglich
                der Funktionen und der Fahrtrichtung)
                aber mit V=0 in die Buffer aufge-
                nommen und nicht ans Gleis gesendet.
```

- **XLokX (0x81)**

```
Dies ist eine Erweiterung von TAMS.
   Wie XLok, jedoch mit echter Decoderspeed;
   wird von OpenDCC nicht unterstützt.
```

- **XLkDisp (0x83)**

```
Dispatchstatus; wird von OpenDCC nicht unterstützt.
```

- **XLokSts (0x84)** - Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0x84 XLokSts
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse
           im Bereich 1 .. 10239)

Antwort: 1 / 4 Bytes
        0: Error-Code. Wenn OK (0x00), dann folgend die
           weiteren Antwort-Bytes
        1: Speed (0 .. 127). 1 bedeutet Emergency-Stop.
        2: Funktionen und Richtung (wie bei XLok):

    bit#   7     6     5     4     3     2     1     0
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |  0  |  0  | Dir | F0  | F4  | F3  | F2  | F1  |
        +-----+-----+-----+-----+-----+-----+-----+-----+

        Dir     Fahrtrichtung wie beim XLOK-Command:
```

```
                1 = vorwärts, 0 = rückwärts
        F0      Frontlicht (FL): 1 = an, 0 = aus
        F4..F1  Status der Funktionen F1 .. F4

        3: Echte Lok-Geschwindigkeit wie sie am Gleis
           ausgegben wird (je nach Lokformat)
           0 = Stop, 1 .. 15/29/127

Mögliche Error-Codes:
        OK       - OK, Befehl ausgeführt
        XBADPRM  - Lokadresse außerhalb des Bereichs
                (1 .. 10239)
        XNODATA  - Es liegen keine Lokdaten vor (Lok
                   nicht in Refresh-Queue)
```

- **XLokCfg (0x85)** - Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0x85 XLokCfg
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)

Antwort: 1 / 5 Bytes
        0: Error-Code. Wenn OK (0x00), dann folgen
           vier weitere Antwort-Bytes
        1: Lok-Format: bei OpenDCC immer 2, bedeutet DCC
        2: Anzahl Geschwindigkeitsstufen
           (ohne Stop): 14, 28 oder 126
        3: 0xFF (bei IB zur Kennzeichnung virtueller
           Loks -> gibt es hier nicht)
        4: 0xFF (bei IB zur Kennzeichnung virtueller
           Loks -> gibt es hier nicht)

Mögliche Error-Codes:
        OK      - OK, Befehl ausgeführt
        XBADPRM - Lokadresse außerhalb des
         Bereichs (1 .. 10239)
        XNODATA - Es liegen keine Lokdaten vor
                  (Lok nicht in Refresh-Queue)
```

- **XLokCfgSet (0x86)** - Länge = 1+4 bytes

```
Diese Kommando gibt es bei der IBox nicht, Tams hat es
dankenswerterweise kompatibel übernommen.
Befehlsbytes:
        0: 0x86 XLokCfg
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)
        3: Lok-Format: wird von OpenDCC ignoriert
           (wir sprechen nur DCC ;-)
        4: Anzahl Geschwindigkeitsstufen
           (ohne Stop): 14, 28 oder 126

Antwort: 1 Byte
        0: Error-Code

Mögliche Error-Codes:
        OK        - OK, Befehl ausgeführt
        XBADPRM   - Lokadresse außerhalb des
                    Bereichs (1 .. 10239)
        XNOSLOT   - Zu viele Loks mit extra Format
                    (vielleicht sollte ein solcher Anwender
                    das default-Format umstellen).
```

Erläuterung: Die Lok wird durch dieses Kommando *nicht* aufgerufen, sondern es wird nur das Format festgelegt. Wenn das Format dem voreingestelltem Format entspricht, so passiert nichts weiter. Wenn es nicht der Voreinstellung entspricht, so wird diese Lokadresse als Ausnahme permanent im EEPROM gespeichert; es gibt 64 Speicherplätze für solche Ausnahmen.

- **XFunc (0x88)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0x88 XFunc
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)
        3: Status: F1 (bit #0) .. F8 (bit #7)

Antwort: 1 Byte 0 oder Error-Code

Mögliche Error-Codes:
```

```
        OK        - 0x00: OK, Befehl ausgeführt
        XBADPRM   - 0x02: Lokadresse außerhalb des
                    Bereichs (1 .. 10239)
        XNOSLOT   - 0x0B: Queues sind voll, Kommando
                    wurde verworfen.
```

- **XFuncX (0x89)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0x89 XFuncX
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)
        3: Status: F9 (bit #0) .. F16 (bit #7)

Antwort: 1 Byte 0 oder Error-Code

Mögliche Error-Codes:
        OK        - 0x00: OK, Befehl ausgeführt
        XBADPRM   - 0x02: Lokadresse außerhalb des
                    Bereichs (1 .. 10239)
        XNOSLOT   - 0x0B: Queues sind voll, Kommando
                    wurde verworfen.
```

Hinweis: OpenDCC unterstützt nur F1 bis F12, F13 bis F16 wird nicht ausgeführt.

- **XFuncSts (0x8C)**- Länge = 1+2 bytes

```
 Befehlsbytes:
        0: 0x8C XFuncSts
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)

Antwort: 0 (gefolgt von einem Byte) oder Error-Code
        Status: F1 (bit #0) .. F8 (bit #7)

Mögliche Error-Codes:
        OK        - 0x00: OK, Befehl ausgeführt
        XBADPRM   - 0x02: Lokadresse außerhalb des
                    Bereichs (1 .. 10239)
        XNODATA   - 0x0A: Die Lok ist nicht im
```

```
                    Refreshbuffer
```

- **XFuncXSts (0x8D)**- Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0x8C XFuncSts
        1: LSB der Lokadresse
        2: MSB der Lokadresse (Adresse im
           Bereich 1 .. 10239)

Antwort: 0 (gefolgt von einem Byte) oder Error-Code
        Status: F9 (bit #0) .. F16 (bit #7)

Mögliche Error-Codes:
        OK        - 0x00: OK, Befehl ausgeführt
        XBADPRM   - 0x02: Lokadresse außerhalb des
                    Bereichs (1 .. 10239)
        XNODATA   - 0x0A: Die Lok ist nicht im
                    Refreshbuffer
```

Hinweis: OpenDCC unterstützt nur F1 bis F12, für F13 bis F16 wird 0 zurückgeliefert.

- **XTrnt (0x90)**- Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0x90 XTrnt (= turnout bzw. Schaltbefehl)
        1: LSB der Weichenadresse (A7 ... A0)
        2: MSB der Weichenadresse und Statusbits
          (Farbe und Spuleaktivierung)
          (Adresse im Bereich 1 .. 4095)
   bit#   7     6     5     4     3     2     1     0
       +-----+-----+-----+-----+-----+-----+-----+-----+
       |Color| Sts | Res |NoCmd| A11 | A10 |  A9 |  A8 |
       +-----+-----+-----+-----+-----+-----+-----+-----+
Bedeutung:
         A11..A0: Addr of Turnout -> max. 4095
                  (die IB erlaubt hier weniger)
                  Hinweis: Es können zwar alle Adressen
                  angesteuert werden, jedoch nur die Lage
                  der untersten 512 wird im OpenDCC
                  gespeichert. (siehe SIZE_TURNOUTBUFFER).
         Color:   1 = closed (grün), 0 = thrown (rot)
```

```
                (Hinweis: wird bei Lenz anders herum
                interpretiert)
        Sts:    Spulenzustand: (1 = ein, 0 = aus)
        Res:    Hier könnte man eine Weiche verriegeln
                - macht OpenDCC nicht
                (und offenbar auch die Intellibox nicht)
        NoCmd:  wird ingoriert.

Antwort: 0  oder Error-Code

Mögliche Error-Codes:
        OK      - 0x00: OK, Befehl ausgeführt
        XPWOFF  - 0x06: abgeschaltet!
        XBADPRM - 0x02: Weichenadresse außerhalb des
                  Bereichs (1 .. 4095)
        XLOWTSP - 0x40: Die Queue ist fast voll
```

Hinweis: fallweise ändert sich die Reaktion auf einen Befehl mit Spulenstatus = Aus wegen Abfrage einer Rück- meldung - s.o.

- **XTrntFree (0x93)**- Länge = 1 Byte

```
Nicht unterstützt: es gibt keine Weichenreservierung
in OpenDCC

Antwort: 0 = Ok, accepted
```

- **XTrntSts (0x94)**- Länge = 1+2 Bytes

```
Befehlsbytes:
        0: 0x94 XTrntSts (= turnout bzw. Schaltbefehl
           - Status Abfrage)
        1: LSB der Weichenadresse (A7 ... A0)
        2: MSB der Weichenadresse

Antwort: 0 = Ok, accepted (ein Byte folgt)
         oder Fehlercode
 2. Byte   7     6     5     4     3     2     1     0
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |n.u. |n.u. |n.u. |n.u. |Conf1|color| Res |Conf0|
        +-----+-----+-----+-----+-----+-----+-----+-----+
```

```
Conf0/1:  00            Motorola
          10            DCC
          01            SX
          11            FMZ

Hinweis: OpenDCC liefert immer DCC zurück;
         Nach Kaltstart: immer "rot",
         "nicht reserviert"
mögliche Fehlercode:

 XBADPRM (02h)   illegal parameter value
 XBADTNP (0Eh)   Error: illegal Turnout
                 address for this protocol
```

Hinweis: Wegen beschränkter RAM-Resourcen können nich alle Weichenlagen gespeichert werden. Zur Zeit ist der Speicher auf 512 Weichen beschränkt: (siehe config.h, SIZE_TURNOUTBUFFER).

- **XTrntGrp (0x95)**- Länge = 1+1 Bytes

```
Befehlsbytes:
        0: 0x95 XTrntGrp (= turnout bzw. Schaltbefehl
           - Status Abfrage, gruppenweise)
        1: Adresse der Gruppe:
           address = ((turnout address - 1) / 8) + 1;
        2: MSB der Weichenadresse

Antwort: 0 = Ok, accepted (zwei Bytes folgen)
         oder Fehlercode
 2. Byte   7     6     5     4     3     2     1     0
        +-----+-----+-----+-----+-----+-----+-----+-----+
        |  W8 |  W7 |  W6 |  W5 |  W4 |  W3 |  W2 |  W1 |
        +-----+-----+-----+-----+-----+-----+-----+-----+
    Bitfeld mit den Positionen der Weichen: jedes Bit
    steht für eine Weiche,
    1 = gerade/grün, 0 = abbiegen/rot;
 3. Byte
    Bitfeld mit den "Reservierungsflags" der Weichen; wird
    von OpenDCC nicht unterstützt, immer als 0 zurückge-
    meldet.
```

Hinweis: Nach Kaltstart: immer ”rot”, ”nicht reserviert” mögliche Fehlercode:

```
XBADPRM (02h)    illegal parameter value
```

Hinweis: Wegen beschränkter RAM-Resourcen können nich alle Weichenlagen gespeichert werden. Zur Zeit ist der Speicher auf 512 Weichen beschränkt: (siehe config.h, SIZE_TURNOUTBUFFER).

- **XSensor (0x98)**- Länge = 1+1 Bytes

```
Befehlsbytes:
        0: 0x98 XSensor (Abfrage S88 Rückmelder)
        1: s88 Modulnummer (1..128) (Modul = 16 Bits)

Antwort:
        1. Byte: 0 = Ok, accepted (zwei Bytes folgen)
                 oder Fehlercode
        2. Byte  Kontakte 1..8 dieses Moduls (Bits 7..0)
        3. Byte  Kontakte 9..16 dieses Moduls
```

Im Gegensatz zur IB liefert OpenDCC immer den akkumulierten Status der S88 Bits. Beim Lesen werden auch die Change-Flags zurückgenommen. (*@modeltreno: wer hat eigentlich diese Bitanordnung verbrochen?)*

- **XSensOff (0x99)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0x99 XSensOff (Löschen und Rücksetzen S88)

Antwort: 1. Byte: 0 = Ok, accepted
```

Löscht alle S88-Bits auf Null und löscht alle Change-Flags.

- **XP88Get (0x9C)**- Länge = 1+1 Bytes

```
Befehlsbytes:
        0: 0x9C XP88Get (Einlesen der S88 Konfiguration)
        1: Parameternummer:
           0: Zahl der automatisch gelesenen Bytes
           1: Zahl der Bytes auf Port 1
           2: Zahl der Bytes auf Port 2
           3: Zahl der Bytes auf Port 3
```

```
Antwort: 1. Byte: 0 = Ok, accepted oder Fehlercode
         2. Byte: Wert des abgefragten Parameters
```

Hinweis: die orginal IB liefert bei 1 und 2 die Quelle für Timer und Counter zurück; dies gibt es bei OpenDCC nicht und ist durch die Angabe der Länge der Teilstrings ersetzt worden. TC benutzt dieses Kommando mit Parameter 0.

- **XP88Set (0x9D)**- Länge = 1+2 Bytess

```
Befehlsbytes:
        0: 0x9D XP88Set (Einstellen der S88 Konfiguration)
        1: Parameternummer:
           0: Zahl der automatisch gelesenen Bytes
           1: Zahl der Bytes auf Port 1
           2: Zahl der Bytes auf Port 2
           3: Zahl der Bytes auf Port 3

Antwort: 1. Byte: 0 = Ok, accepted oder Fehlercode
```

Hinweis: die orginal IB stellt bei 1 und 2 die Quelle für Timer und Counter ein; dies gibt es bei OpenDCC nicht und ist durch die Angabe der Länge der Teilstrings ersetzt worden.
Die Längen der Teilstrings könne auch durch entsprechende Sonder - Optionen eingestellt werden.

- **Xs88Tim (0x9E)**- Länge = 1+1 Bytes
Hinweis: wird von OpenDCC nicht unterstützt. Wer braucht das?

- **Xs88Cnt (0x9F)**- Länge = 1+1 Bytes
Hinweis: wird von OpenDCC nicht unterstützt. Wer braucht das?

- **XVer (0xA0)**- Länge = 1 Bytes

```
Befehlsbytes:
        0: 0xA0 XVer (Versionsabfrage)

Antwort: Liste, jeder Eintrag ist wie folgt aufgebaut:
         1. Byte: Zahl der Bytes in diesem Listen-
                  element (0 bedeutet Liste zu Ende)
```

```
          2. Byte: Niederwertiges Byte der Version.
          3.-n. Byte: Höherwertigere Bytes, sofern vor-
                      handen.

Im Falle OpenDCC gibt es nur einen Listeneintrag
der Länge 1.

Beispiel:  0x01 = Länge 1
           0x0D = Version 0.13
           0x00 = Ende der Liste
```

Hinweis: die orginal IB liefert 6 Listenelement inkl. einer Seriennummer zurück.

- **XP50XC (0xA1)**- Länge = 1+1 Bytes
  Umstellen des "Escape-Zeichen" für das P50X Protokoll auf ein anderes Zeichen. Dies benutzt offenbar kein PC-Programm und wird auch von OpenDCC nicht unterstützt. War wohl von modeltreno eingebaut worden, um auf irgendwelche heimtückischen Protokolländerung von Tante M reagieren zu können.

- **XStatus (0xA2)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xA2 XStatus (Statusabfrage für Spannung,
           Zustand usw.)

Antwort: Bitfeld, folgende Zuordnung:
         Bit 7:  Erweiterungsbit, wenn 1, kommt noch
                 ein weiteres Byte als Antwort.
                 (momentan immer 0)
         Bit 6: VREG:  1: Spannungsregelung einge-
                          schaltet (ist bei OpenDCC
                          gesetzt)
                       0: keine Spannungsregelung
         Bit 5:  I2C   1: externes I2C Gerät vorhanden
                       0: nichts angeschlossen (OpenDCC)
         Bit 4:  HALT  1: Loks angehalten, aber Gleis-
                          spannung vorhanden
         Bit 3:  PWR:  1: Zustand "EIN",
                          grüne LED leuchtet.
                       0: Zustand "AUS",
```

```
                    rot leuchtet.
Bit 2:  HOT    1: zu heiss
               0: (OpenDCC)
Bit 1:  GO     1: falls gerade der grüne Taster
                  gedrückt wird.
                  (keine Entprellung)
Bit 0:  STOP   1: falls gerade der rote Taster
                  gedrückt wird.
                  (keine Entprellung)
```

Hinweis: Der HALT Zustand wird, da er bei der IB manuell nicht ausgelöst werden kann, von Steuerungsprogrammen einfach ignoriert.

- **XSOSet (0xA3)**- Länge = 1+3 Bytes

```
Befehlsbytes:
        0: 0xA3 XSOSet (Spezial-Option setzen)
        1: Adresse niederwertiger Teil.
        2: Adresse höherwertiger Teil. Gültiger
           Bereich 0...1023
        3: Inhalt

Antwort: entweder 0 (Kommando okay) oder Fehlercode
```

Hinweis: Dieses Kommando gibt es bei Orginal Intellibox nicht, es ist eine Erweiterung von OpenDCC.
Achtung:*Mit Bedacht verwenden, es können Variablen so verändert werden, dass OpenDCC nicht mehr funktioniert.*

- **XSOGet (0xA4)**- Länge = 1+2 Bytes

```
Befehlsbytes:
        0: 0xA4 XSOGet (Spezial-Option auslesen)
        1: Adresse niederwertiger Teil.
        2: Adresse höherwertiger Teil. Gültiger
           Bereich 0...1023

Antwort: entweder 0 (Kommando okay, ein Byte wird
         folgen) oder Fehlercode
         2. Byte: Inhalt der Spezual-Option
```

Hinweis: Die Spezial-Optionen sind in Ihrer Adresse so gewählt, dass bestmögliche Übereinstimmung mit der Intellibox erreicht wird. Der Grund war die Analyse diverser Einstellungen durch railware, durch die Belegung wird die *"Meckerquote"* von railware reduziert.

Hinweis: SO 1 wird beim Lesen umkodiert, jedoch nicht beim Schreiben.

- **XHalt (0xA5)**- Länge = 1 Bytes

```
Befehlsbytes:
         0: 0xA5 XHalt (Loks anhalten aber DCC
            bleibt aktiv)

Antwort: 0 (Kommando okay)
```

Hinweis: Offenbar benutzt kein Steuerprogramm diese Option, die schalten immer gleich ab.

- **XPwrOff (0xA6)**- Länge = 1 Bytes

```
Befehlsbytes:
        0: 0xA6 XPwrOff (DCC Hauptgleis aus-
           schalten)

Antwort: 0 (Kommando okay)
```

- **XPwrOn (0xA7)**- Länge = 1 Bytes

```
Befehlsbytes:
        0: 0xA7 XPwrOn (DCC Hauptgleis ein-
           schalten)

Antwort: 0 (Kommando okay)
```

- **XNop (0xC4)**- Länge = 1 Bytes

```
Befehlsbytes:
         0: 0xC4 XNop (nichts tun)

Antwort: 0 (Kommando okay)
```

Klar, "Nichts tun", das können wir, da sind wir stark! Nein, im Ernst: der Befehl dient für das komische BABI bei der Vielfalt der IB-Protokolle zum Rausfinden der Baudrate.

- **XP50Len1 (0xC6)**- Länge = 1+1 Bytes
  Dieser Befehl dient bei der orginal Intellibox zum Durchreichen des Befehl an das darunter liegende P50 Protokoll. Man war sich offenbar bei modeltreno nicht wirklich sicher und hat sich neben dem X-Char Befehl noch ein weiteres Hintertürchen aufgehalten.
  Wird von OpenDCC nicht unterstützt.

- **XP50Len2 (0xC7)**- Länge = 1+2 Bytes
  Noch so eine Geheimtür wie oben.

- **XEvent (0xC8)**- Länge = 1 Byte

```
Befehlsbytes:
      0: 0xC8 XEvent (Statusabfrage für Zustands-
      änderungen, wie z.B. S88, Programmierung usw.)

Antwort: Liste von Bitfeldern (8 Bit) , jeweils das
         MSB zeigt an, ob noch ein weiteres Bitfeld
         folgt:

 Byte 1:
    Bit 7:  Erweiterungsbit, wenn 1, kommt noch ein
            weiteres Byte als Antwort.
    Bit 6:  reserviert
    Bit 5:  TRNT  1: da war mindestens ein Weichen-
                     Event
    Bit 4:  Tres  1: da war mindestens ein Zugriff
                     auf eine reservierte Weiche
    Bit 3:  PWR   1: da war mind. ein
                     Power Off Event(*)
    Bit 2:  S88   1: da war mind. ein s88-Event (*)
    Bit 1:  GO    1: da war ein IR-Event
    Bit 0:  LOK   1: da war mind. ein Zugriff auf
                     eine Lok
 Byte 2:
    Bit 7:  Erweiterungsbit, wenn 1, kommt noch ein
            weiteres Byte als Antwort.
```

```
      Bit 6:  Sts:  1: Änderung im Status
      Bit 5:  Hot   1: Temperaturanzeige
      Bit 4:  PTsh  1: Schluß vom Programmiergleis zum
                       Hauptgleis
      Bit 3:  RSsh  1: Schluß auf dem Programmiergleis
                       oder auf dem Boosteranschluß
      Bit 2:  IntSh 1: Kurzschluß intern
      Bit 1:  LMSh  1: Kurzschluß auf den Lokmausanschluß
      Bit 0:  LOK   1: Kurzschlußmeldung eines externen
                       Boosters
   Byte 3:
      Bit 7:  Erweiterungsbit, wenn 1, kommt noch ein
              weiteres Byte als Antwort.
      Bit 6:  reserviert
      Bit 5:  reserviert
      Bit 4:  ExVlt 1: Fremdspannung vorhanden
      Bit 3:  TkRel 1: Übernahme einer Lok durch andere
                       Controller
      Bit 2:  Mem   1: Memory Event
      Bit 1:  RSOF  1: RS232 Overflow
      Bit 0:  LOK   1: Programming Track Event (*)
```

Hinweis: OpenDCC liefert nur die (*) gekennzeichneten Statusbits.

- **XEvtLok (0xC9)**- Länge = 1 Byte

  ```
  Antwort: 0x80
  ```

  Hinweis: wird von OpenDCC nicht unterstützt.

- **XXEvtTrnt (0xCA)**- Länge = 1 Byte

  ```
  Antwort: 0x00
  ```

  Hinweis: wird von OpenDCC (noch) nicht unterstützt. Dies kann fallweise zur Rückmeldung nicht schaltender Weichen verwendet werden.

- **XEvtSen (0xCB)**- Länge = 1 Byte

```
Befehlsbytes:
   0: 0xCB XEvtSen (Abfrage Sensor Events)

Antwort: Liste aus maximal 128 Einträgen, jeder
         Eintrag ist wie folgt aufgebaut:
   1. Byte: falls  0: dieses ist der letzte Listen-
                      eintrag.
            falls !0: s88 Modulnummer (1..128)
            (Modul = 16 Bits)
   2. Byte  Kontakte 1..8 dieses Moduls (Bits 7..0)
   3. Byte  Kontakte 9..16 dieses Moduls (Bits 15 ..8)
```

Das Vorliegen eines S88-Event wird mit XEvent gemeldet; Es kann sein, das ein Event gemeldet wurde, sich aber kein S88 geändert hat, weil es inzwischen schon wieder zurückgesetzt ist. Beim Lesen werden die Change-Flags zurückgenommen, es wird kein Modul zweimal gemeldet.
(@modeltreno: wer hat eigentlich diese Bitanordnung verbrochen?)

- **XEvtPT (0xCE)**- Länge = 1 Byte

```
Befehlsbytes:
   0: 0xCE XEvtPT (Abfrage Programmierergebnisse)

Antwort:
   1. Byte: falls 0xF5: noch nicht fertig, es gibt
                        nichts zu melden
            falls [1,2,3,4,6] Zahl der Folgebytes
   2. Byte  Status der Programming Task

         0x00   Command completed, no errors
         0xFF   Timeout
         0xFE   No acknowledge from decoder
                (but a write maybe was successful)
         0xFD   Short! (on the PT)
         0xFC   No decoder detected
         0xFB   Generic Error
         0xFA   Error during DCC direct bit mode
                operation
         0xF9   No acknowledge to paged operation
                (paged r/w not supported?)
         0xF8   Error during Selectrix read
         0xF7   XPT_DCCQD: Ok (direct bit read mode
```

```
              is (probably) supported)
        0xF6  XPT_DCCQD: Not Ok (direct bit read
              mode is (probably) not supported)
        0xF4  Task terminated (see XPT_Term cmd)
        0xF3  No task to terminate
              (see XPT_Term cmd)
        0xF2  Cannot terminate task
              (see XPT_Term cmd)

  3. Byte  (nur falls vom 1. Byte angekündigt)
                Erstes Ergebnis der Programmier-
                task, dies kann sein:
                  - der Inhalt eines Register oder
                    CV (DCC Read cmds);
                  - low byte einer langen Adresse
                    (DCC long addr read cmd);

  4. Byte  (nur falls vom 1. Byte angekündigt)
                Zweites Ergebnis der Programmier-
                task, dies kann sein:
                  - high byte einer langen Adresse
                    (DCC long addr read cmd);
```

- **XDCC_PDR (0xDA)**- Länge = 1+4 Bytes

```
Befehlsbytes:
        0: 0xDA XDCC_PDR (= Lok-Programmieren auf dem
           Hauptgleis = POM, CV-Lesen)
        1: LSB der Lokadresse
        2: MSB der Lokadresse (1-10239)
        3: Low Byte der CV-Adresse, welche zu lesen ist.
        4: High Byte der CV-Adresse, welche zu lesen ist.
           (1..1024)

Antwort: 0 = Ok, accepted.
         0x80 = busy, command ignored
```

Die Antwort wird über XEvtPT übergeben.
Mit POM können Programmierbefehle am Hauptgleis gesendet werden. Vorläufig ist noch kein Lesen per POM möglich - NMRA-BiDi hat OpenDCC (noch) nicht.

- **XDCC_PAR (0xDB)**- Länge = 1+4 Bytes

```
Befehlsbytes:
        0: 0xDB XDCC_PAR (= Accessory-Programmieren auf
           dem Hauptgleis = POM, , CV-Lesen)
        1: LSB der Zubehöradresse
        2: MSB der Zubehöradresse (1-510)
        3: Low Byte der CV-Adresse, welche zu schreiben
           ist.
        4: High Byte der CV-Adresse, welche zu schreiben
           ist. (1..1024)

Antwort: 0 = Ok, accepted
         0x80 = busy, command ignored
```

Die Antwort wird über XEvtPT übergeben.
Unter Adresse ist hier die Decoderadresse gemeint, nicht die Weichenadresse! Mit POM können Programmierbefehle am Hauptgleis gesendet werden. Vorläufig ist noch kein Lesen per POM möglich - NMRA-BiDi hat OpenDCC (noch) nicht.

- **XPT_DCCEWr (0xDC)**- Länge = 1+4 Bytes

```
Antwort: 0x00
```

Erzeugen und Schreiben einer Geschwindigkeitskennlinie.
Hinweis: wird von OpenDCC (noch) nicht unterstützt.

- **XDCC_PD (0xDE)**- Länge = 1+5 Bytes

```
Befehlsbytes:
        0: 0xDE XDCC_PD (= Lok-Programmieren auf dem
           Hauptgleis = POM)
        1: LSB der Lokadresse
        2: MSB der Lokadresse (1-10239)
        3: Low Byte der CV-Adresse, welche zu schreiben
           ist.
        4: High Byte der CV-Adresse, welche zu schreiben
           ist. (1..1024)
        5: Wert
```

```
Antwort: 0 = Ok, accepted
         0x80 = busy, command ignored
```

Mit POM können Programmierbefehle am Hauptgleis gesendet werden - diese werden nicht quittiert. Es ist kein Lesen per POM möglich - NMRA-BiDi hat OpenDCC (noch) nicht.

- **XDCC_PA (0xDF)**- Länge = 1+5 Bytes

```
Befehlsbytes:
        0: 0xDF XDCC_PA (= Accessory-Programmieren
           auf dem Hauptgleis = POM)
        1: LSB der Zubehöradresse
        2: MSB der Zubehöradresse (1-510)
        3: Low Byte der CV-Adresse, welche zu schreiben
           ist.
        4: High Byte der CV-Adresse, welche zu schreiben
           ist. (1..1024)
        5: Wert

Antwort: 0 = Ok, accepted
         0x80 = busy, command ignored
```

Unter Adresse ist hier die Decoderadresse gemeint, nicht die Weichenadresse! Mit POM können Programmierbefehle am Hauptgleis gesendet werden - diese werden nicht quittiert. Es ist kein Lesen per POM möglich - NMRA-BiDi hat OpenDCC (noch) nicht.

- **XPT_Sts (0xE0)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xE0 XPT_Sts (= Abfrage Status der
                         Programmierroutine)

Antwort: zwei Bytes, wie folgt kodiert;
  bit#  7     6     5     4     3     2     1     0
     +-----+-----+-----+-----+-----+-----+-----+-----+
     | Evt |  x  |  x  |  x  |  x  |  x  | Rel | Mode|
     +-----+-----+-----+-----+-----+-----+-----+-----+
               Evt:  0 = no PT event is pending
                     1 = a PT event is pending
               Rel:  0 = PT relay is off
                     1 = PT relay is on
```

```
                    (always in OpenDCC Mode)
                Mode: 0 = PT in 'PT only' mode
                    1 = PT in 'auto' mode

                2. Byte
                0 = es läuft keine Programmierroutine
                1 = es läuft gerade was
```

- **XPT_On (0xE1)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xE1 XPT_On (= Programmiermodus)

Antwort: 0 = Ok, accepted
```

- **XPT_Off (0xE2)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xE2 XPT_Off (= Verlassen Programmiermodus)

Antwort: 0 = Ok, accepted
```

OpenDCC schaltet nach diesem Kommando auf RUN_OFF d.h. Hauptgleis ist gewählt, aber noch abgeschaltet.

- **XPT_DCCSr (0xEA)**- Länge = 1 Byte
  Search for address of a DCC decoder (decoders which are not 'readable', e.g. Roco/Lenz digital crane) - not supported by OpenDCC

- **XPT_DCCQA (0xEB)**- Länge = 1 Byte
  Read address using obsolete "Address Query" packet (can only report 7-bit addresses in range 1..111) - not supported by OpenDCC

- **XPT_DCCRR (0xEC)**- Länge = 1+2 bytes

```
        0: 0xEC XPT_DCCRR (= Lesen mit Hilfe des
           Register-Mode Befehls)
        1: Register, das zu lesen ist (1..8).
        2: 0

Antwort: 0 = Ok, accepted
```

- **XPT_DCCWR (0xED)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0xED XPT_DCCWR (= Schreiben mit Hilfe
           des Register-Mode Befehls)
        1: Register, das zu schreiben ist (1..8).
        2: 0
        3: Der zu schreibende Inhalt (0..255)

Antwort: 0 = Ok, accepted
```

- **XPT_DCCRP (0xEE)**- Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0xEE XPT_DCCRP (= Lesen mit Hilfe des
           Register-Mode Befehls und Pages)
        1: Low Byte der CV-Adresse, welche zu
           lesen ist.
        2: High Byte der CV-Adresse, welche zu
           lesen ist. (1..1024)

Antwort: 0 = Ok, accepted
```

Beim Paged Mode wird zuerst das Register 6 mit einer Auswahladresse beschreiben. Diese Auswahladresse wählt eine "Speicherseite" (= page) im Decoder an. Diese wird dann anstelle der Register 1-4 eingeblendet. Dann wird auf diesem Registern die tatsächliche Lese bzw. Schreiboperation vorgenommen. Das ist nicht wirklich schnell, deshalb sollte besser der nachfolgende Direktbefehl verwendet werden.

- **XPT_DCCWR (0xEF)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0xEF XPT_DCCWP (= Schreiben mit Hilfe
           des Register-Mode Befehls und Pages)
        1: Low Byte der CV-Adresse, welche zu
           schreiben ist.
        2: High Byte der CV-Adresse, welche zu
           schreiben ist. (1..1024)
        3: Der zu schreibende Inhalt (0..255)

Antwort: 0 = Ok, accepted
```

- **XPT_DCCRD (0xF0)**- Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0xF0 XPT_DCCRD (= Lesen mit Hilfe des
           Direct-Modes)
        1: Low Byte der CV-Adresse, welche zu
           lesen ist.
        2: High Byte der CV-Adresse, welche zu
           lesen ist. (1..1024)

Antwort: 0 = Ok, accepted
```

Dieser Befehl liest mit Hilfe des DCC direkt-read Kommandos das CV aus der Lok aus. Mit diesem DCC-Befehl kann die Lok nur gefragt werden, ob das CV den angegebenen Inhalt hat. Also muß normalerweise die Zentrale der Reihe nach durchsuchen, bis ein Datum gefunden wird. Um das zu Beschleunigen, gibt es folgende Besonderheit beim direkten Lesen eines Bytes per DCC:
OpenDCC prüft vorher nach, ob der Decoder Bitoperationen beherrscht. Falls ja, wird das Byte mit den Bitbefehlen gelesen und dann noch gegengeprüft. Fall nein, wird konventionell mit einer Suchschleife über alle 256 möglichen Zustände gesucht. OpenDCC merkt sich für 500ms, ob der Decoder Bitoperationen kann. Wenn innerhalb dieser 500ms ein weiterer Lesebefehl ausgeführt wird, so entfällt automatisch die erneute Prüfung auf Bitfähigkeit.

- **XPT_DCCWD (0xF1)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0xF1 XPT_DCCWD (= Schreiben mit Hilfe
           des Direct-Modes)
        1: Low Byte der CV-Adresse, welche zu
           schreiben ist.
        2: High Byte der CV-Adresse, welche zu
           schreiben ist. (1..1024)
        3: Der zu schreibende Inhalt (0..255)

Antwort: 0 = Ok, accepted
```

- **XPT_DCCRB (0xF2)**- Länge = 1+2 bytes

```
Befehlsbytes:
        0: 0xF2 XPT_DCCRB (= Lesen mit Hilfe des
           Bit-Modes)
        1: Low Byte der CV-Adresse, welche zu
           lesen ist.
        2: High Byte der CV-Adresse, welche zu
           lesen ist. (1..1024)

Antwort: 0 = Ok, accepted
```

Dieser Befehl liest mit Hilfe des DCC bit-read Kommandos das CV aus der Lok aus. Obwohl es ein Bitbefehl ist, wird das ganze CV-Byte ausgelesen, die tatsächliche Bitmaske muß der Host machen. Vom den implementierten Lesemethoden ist diese die schnellste, da nur 8 Zugriffe zzgl. ein Kontrollzugriff erforderlich sind.

- **XPT_DCCWB (0xF3)**- Länge = 1+3 bytes

```
Befehlsbytes:
        0: 0xF1 XPT_DCCWD (= Einzelbit Schreiben
           mit Hilfe des Bit-Modes)
        1: Low Byte der CV-Adresse, welche zu
           schreiben ist.
        2: High Byte der CV-Adresse, welche zu
           schreiben ist. (1..1024)
        3: Bitposition (0..7)
        3: Der zu schreibende Inhalt (0|1)

Antwort: 0 = Ok, accepted
```

- **XPT_DCCQD (0xF4)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xF4 XPT_DCCQD (= Abfrage, ob der
                           Decoder Einzelbit kann)

Antwort: 0 = Ok, accepted
```

- **XPT_DCCRL (0xF5)**- Länge = 1 Byte

```
Befehlsbytes:
        0: 0xF5 XPT_DCCRL (= Abfrage lange Adresse)
```

```
Antwort: 0 = Ok, accepted
```

Die lange Adresse ist in CV17 und CV18 abgelegt und speziell codiert. In der CV-Übersicht ist ein Rechner dafür.

- **XPT_DCCWL (0xF6)**- Länge = 1+2 Bytes

```
Befehlsbytes:
      0: 0xF6 XPT_DCCWL (= Schreiben lange Adresse)
      1: Low Byte der langen Adresse.
      2: High Byte der langen Adresse.

Antwort: 0 = Ok, accepted
```

Es wird automatisch auch das Bit 5 in CV29 mitgeschrieben; dieses Bit schaltet die lange Adresse aktiv. Der Decoder muß Bit-Mode beherrschen.

- **XPT_DCCRA (0xF7)**- Länge = 1 Byte

```
Befehlsbytes:
         0: 0xF7 XPT_DCCRA (= Abfrage lange Adresse,
            Accessory Decoder)

Antwort: 0 = Ok, accepted
```

Die lange Adresse ist in CV513 und CV521 abgelegt und speziell codiert. In der CV-Übersicht ist ein Rechner dafür. Dieser Befehl ist noch nicht implementiert!!!

- **XPT_DCCWA (0xF8)**- Länge = 1+2 Bytes

```
Befehlsbytes:
        0: 0xF6 XPT_DCCWA (= Schreiben lange Adresse,
           Accessory Decoder)
        1: Low Byte der langen Adresse.
        2: High Byte der langen Adresse. (0..511)

Antwort: 0 = Ok, accepted
```

Dieser Befehl ist noch nicht implementiert!!!

- **XPT_Term (0xFE)**- Länge = 1 Byte

```
Befehlsbytes:
         0: 0xFE XPT_Term (= Beenden der aktuellen
            Programmieraufgabe (Abbruch))

Antwort: 0 = Ok, accepted and terminated
         0xF3 = no task is active
```

3.2.6 Konfiguration und Firmwareupdate:

- Konfiguration
  Es gibt eine Reihe von Parameter, die das Verhalten von OpenDCC verändern. Diese sind permanent im EEPROM gespeichert.

- Firmware-Update
  Einfach beim Start des Gerätes die Stoptaste gedrückt halten und dann mit dem entsprechenden Tool die Firmware (Kapitel 3.7 auf Seite 103) einspielen.

3.3 Übersicht Spezial-Optionen und Konfiguration

3.3.1 Konfigurieren per Optionseinstellungen

OpenDCC kann sowohl beim Übersetzen als auch zur Laufzeit flexibel auf die Bedürfnisse des jeweiligen Anwenders angepaßt werden.

- Die Anpassmöglichkeiten bei Übersetzen sind in der Regel in config.h zusammengefaßt. Hierunter fallen z.B. die Größen der internen Speicher und Kommandolisten sowie der generelle Support von bestimmten Funktionen und Protokollen wie z.B. Intellibox, Lenz, S88, DMX. Nähere Erläuterungen und Beispiele finden sich in config.h und config.c.
- Eine gewählte Konfiguration hat darüber hinaus noch weitere Anpassmöglichkeiten. Die sind durch interne, dauerhaft gespeicherte Variablen während des Betriebes einstellbar. Diese Variablen sind im EEPROM gespeichert. Generell gilt, dass OpenDCC nach einem Verändern dieser Variablen neu gestartet werden muß, damit die Änderungen wirksam werden.
  Diese EEPROM-Variablen können mit drei unterschiedlichen Techniken

beschrieben werden:

- Vorbelegung beim Übersetzen. Der Inhalt des EEPROMs wird im config.c festgelegt.
- Zugriff über den Befehl SO (P50X-ASCII-Mode) (siehe Kapitel 3.2.4 auf Seite 48) oder XSOset (0xa3) bzw. XSOget (0xa4) bei P50X-Binary-Mode(siehe Kapitel 3.2.5 auf Seite 55).
- Zugriff über den CV-Programmierbefehl einer virtuellen Lok (ab V0.14). Hierzu muß beim Starten von OpenDCC die GO-Taste gedrückt sein; anschließend werden die Programmierbefehl (DCC byteweise) auf die internen EEPROM Variablen umgelenkt. (Beispiel: Konfiguration mit Trainprogrammer (siehe Kapitel 3.4 auf Seite 90)

3.3.2 Liste der Einstelloptionen (SO)

Im folgenden sind die default-Einstellungen der V0.15 mit einem '*' gekennzeichnet. Diese Voreinstellung ist aber leicht änderbar und wird sich auch ändern, daher diese Einstellungen vor der Benutzung immer kontrollieren.

- **SO# 0**- Version Bytewert, ist hexadezimal zu interpretieren, z.B. 0x0E = Version 0.14

- **SO# 1**- Baudrate Diese SO enhält kodiert die Baudrate. Die gewählte Zuordnung entspricht dem Lenzinterface:
  0 = 9600 Baud
  1 = 19200 Baud (default)
  2 = 38400 Baud
  3 = 57600 Baud
  4 = 115200 Baud
  5 = 2400 Baud
  6 = 4800 Baud

  Bei Abfrage über den XSOget-Befehl (0xa4) wird allerdings im Intellibox-Parser umcodiert. Grund war das Reconfigurieren der Schnittstelle nach dem Auslesen dieser SO durch Traincontroller. Also mußte in diesem Spezialfall die Zuordnung der Baudrate gemäß der Vergabe der Intellibox erfolgen:
  0 = 2400 Baud
  1 = 4800 Baud
  2 = 9600 Baud

3 = 19200 Baud
4 = 38400 Baud
5 = 57600 Baud (vermutlich - ist noch ungetestet)
6 = 115200 Baud (vermutlich - ist noch ungetestet)

- **SO# 2**- OpenDCC Mode reserviert
- **SO# 3**- virtueller Decoder low
  Bildet gemeinsam mit SO 4 einen 16-Bit Wert
- **SO# 4**- virtueller Decoder high
  16-Bit-Wert: Zugriffe auf Schaltdecoder mit einer größeren Adresse als hier angegeben werden nicht als DCC-Befehl auf das Gleis gelegt, sondern intern zu einem Aufruf eines DMX-Makros umge formt. Das DMX-Makro ist somit als virtueller Decoder ansprechbar, siehe Raumlichtsteuerung. Mit der Einstellung 0xffff ist dieses Umformen komplett abgeschaltet.
  Voreinstellung: 0xffff
- **SO# 5**- reserviert
- **SO# 6**- CTS-Usage
  255, CTS is unused - diese SO ist für railware-Kompatibilität notwendig. Railware würde hier gerne 20 eingestellt sehen, aber bei OpenDCC will ich die CTS-Leitung nur für Kommunikation, nicht für Zustandsmeldungen (ist ja höchst unhygienisch ;-) verwenden.
- **SO# 7**- S88 Mode

  Bit 0:
  Zustand 0 = kein Einlesen der externen Rückmelder
  Zustand 1 = normaler S88-Betrieb mit Einlesen der externen Rückmelder (default*)

  Bit 1:
  Zustand 0 = keine Weichenrückmeldung (default*)
  Zustand 1 = Weichenrückmeldung eingeschaltet

  Die entsprechenden S88 Bits werden von der Weichenrückmeldung angesteuert. Sieh S0 17 für die Bedeutung des Rückmeldebits. Die S88-Bitnummer entspricht der Weichennummer zzgl. eines Offsets, welcher in SO16 eingestellt wird.
  Bei der Einstellung von S88 Mode muß beachtet werden, dass sich Weichenrückmeldung und normales Einlesen nicht in die Quere kommt: wenn

beides zugleich aktiv ist, dann muß der Offset in SO16 so groß gewählt werden, dass die Weichenrückmeldungen erst hinter den normalen S88 Modulen abgespeichert werden.

- **SO# 8** S88 Autoread
  Zahl der Bytes, die bei S88 automatisch gelesen werden. Diese Zahl muß gleich der Summe aus SO9, SO10, SO11 sein. Innerhalb OpenDCC wird diese Variable nicht weiter ausgewertet (OpenDCC macht immer autoread), aber railware will es wissen :-). Es können max. 128 Bytes eingestellt werden. (siehe auch S88_SIZE_MAX in config.h)

- **SO# 9** S88 Module 1
  Anzahl der Bytes auf dem S88-Strang 1. Ein übliches Modul hat 2 Byte. *Diese Variable kann auch mit X88PGet (0x9C-0x01) abgefragt und mit X88PSet (0x9D-0x01) gesetzt werden.* **default: 2**

- **SO# 10** S88 Module 2
  Anzahl der Bytes auf dem S88-Strang 2. Ein übliches Modul hat 2 Byte. *Diese Variable kann auch mit X88PGet (0x9C-0x02) abgefragt und mit X88PSet (0x9D-0x02) gesetzt werden.* **default: 2**

- **SO# 11** S88 Module 3
  Anzahl der Bytes auf dem S88-Strang 3. Ein übliches Modul hat 2 Byte. *Diese Variable kann auch mit X88PGet (0x9C-0x03) abgefragt und mit X88PSet (0x9D-0x03) gesetzt werden.* **default: 2**

- **SO# 12** Invertiere Weichenbefehl
  Gilt nur bei Lenzprotokoll!
  Falls = 1(*): Die Ziellage der Weiche wird beim Lenz-Protokoll invertiert. *Hintergrund: Intellibox und Lenz unterscheiden sich hier in der Interpretation des DCC-Standard (!). Mit dieser Option wird diese Diskrepanz auf der Ebene der Zentrale behoben und wird nicht in das Steuerprogramm ausgelagert.*

- **SO# 13** Weichenwiederholung
  Ein Weichenbefehl wird bei der Gleisausgabe sooft wiederholt.
  Voreinstellung: 2

- **SO# 14** Weichenschaltzeit
  Nach dieser Zeit (in Einheiten von 50ms) werden die Weichenspulen wieder abgeschaltet. OpenDCC schaltet die Weichenspulen nie selbst ab (das soll der Decoder machen). Ein ev. Abschaltbefehl vom Host wird durchgereicht und ev. zur Lagerückmeldung der Weiche benutzt. Diese SO wird aber von railware abgefragt, deshalb gibt es sie.
  Voreinstellung: 100

- **SO# 15** Einschaltzustand
  Hier wird die Protokollart bei IB-Emulation festgelegt:
  ***0: Normalmode (P50/P50X mixed)**
  1: P50X-Only Mode

- **SO# 16** Offset für Weichenrückmeldung
  Bei eingeschalteter Weichenrückmeldung werden jeweils s88-Ereignisse generiert, wenn die Weiche nicht geschaltet hat. Diese werden unter Berücksichtigung dieses Offsets in den S88 Puffer eingetragen. Der Offset ist in Einheiten von Bytes angegeben.
  Beispiel: Weiche 10 schalte nicht, der Offset sei auf 6 eingestellt.
  →es wird das Rückmeldebit 58 aktiviert (58 = 10 + 6*8). Es ist zu beachten, dass der S88-Puffer nur 1024 Bits groß ist, d.h. es können nicht alle möglichen Weichenadressen zurückgemeldet werden.
  Voreinstellung: 0

- **SO# 17** Weichenrückmeldung Mode
  Mit dieser SO wird festgelegt, wie die Weichenrückmeldung gemeldet wird:

  1. SO17 = 0
     Das jeweilige S88 Bit geht auf 1, wenn die Bestätigung der Weiche für den letzten Schaltvorgang vorliegt (positive Rückmeldung). Es ist ein Bit je Weiche vorhanden, d.h. es wird der jeweils letzte Schaltvorgang bestätigt.

  2. SO17 = 1
     Das jeweilige S88 Bit geht auf 1, wenn der Schaltvorgang nicht bestätigt wurde. (Fehlerbit).

  3. **SO17 = 2 (default*)**
     Das jeweilige S88 Bit zeigt die Lage der Weiche an:
     1: Weiche steht auf Abzweig (rot)
     0: Weiche steht auf Gerade (grün)

  4. SO17 = 3
     Es werden zwei Bits je Weiche belegt, diese zeigen je für sich das Anlegen der Weichenzunge auf der jeweiligen Seite an:

Bits	Bedeutung
00:	Weiche läuft gerade um
01:	Weiche steht auf Abzweig (rot)
10:	Weiche steht auf Gerade (grün)
11:	Ungültiger Zustand, Rückmeldesystem defekt
	Achtung: dieser Optionwert ist (noch) nicht implementiert.

Zur Zeit nur SO17 = 0,1 oder 2 erlaubt.

- **SO# 18** Extended Resets
  Mit dieser SO wird festgelegt, ob und um wieviel die Zahl der Reset-Pakete vor einem Programmierbefehl erhöht wird:

  0: Es werden die in der NMRA-Norm vorgebene Paketanzahl gesendet.
  1-10: Es werden um die angegebene Zahl mehr Reset-Packets gesendet.
  Voreinstellung: 3

  Hintergrund: Manche Decoder (z.B. Lokpilot V.2) brauchen nach dem Lesen einer CV eine gewisse Zeit, bis sie Lesebefehle auf die nächste CV annehmen. Deswegen wird beim blockweisen Lesen von CV's der erste Befehl vom 2. Byte nicht korrekt beantwortet. Dies führt je nach erwartetem Bitmuster zu einem falsch gelesenem CV, welches dann am Schluß bei der Kontrollüberprüfung durchfällt.

- **SO# 19** Extended Program Commands
  Mit dieser SO wird festgelegt, ob und um wieviel die Zahl der Programmierbefehle erhöht wird:

  0: Es werden die in der NMRA-Norm vorgebene Paketanzahl gesendet.
  1-10: Es werden um die angegebene Zahl mehr Programmierpakete gesendet.
  Voreinstellung: 3

  Hintergrund: Manche Decoder (z.B. ältere Lenz/Roco) lassen sich mit der Programmierquittung etwas Zeit. Deswegen kann bei NMRA-konformen Timing vorkommen, das OpenDCC nicht lang genug auf die Quittung wartet und deswegen ein CV nicht korrekt liest. (siehe hierzu auch die FAQ von Uhlenbrock)

  SO18 und S019 können bei Verwendung NMRA-konformer Decoder auf 0 gestellt werden; die Verlängerung der Timings schadet aber nicht (Das CV-Programmieren dauert halt unwesentlich länger).

- **SO# 20** PoM (Programmieren auf dem Hauptgleis) Wiederholung
  Mit dieser SO wird festgelegt, wie oft ein PoM-Befehl wiederholt wird. Voreinstellung 3. (lt. NMRA mindestens zweimal)

- **SO# 21** Geschwindigkeitsbefehle Wiederholung
  Mit dieser SO wird festgelegt, wie oft ein Speedbefehl wiederholt wird. Voreinstellung 3.
  Dies sind die direkten Wiederholungen, welche möglichst bald erfolgen. Unabhängig davon erfolgt eine zyklische Auffrischung aller bisher benutzten Loks, sofern keine neuen aktuellen Befehle vorliegen.
  Befinden sich zu viele Loks in der Liste der direkten Wiederholungen, so wird diejenige mit dem kleinsten Restwiederholungszähler verdrängt. Dadurch passt sich OpenDCC auch bei großer Verkehrslast dynamisch an.

- **SO# 22** Funktionsbefehl Wiederholung
  Mit dieser SO wird festgelegt, wie oft ein Funktions-Befehl wiederholt wird. Voreinstellung 0.
  Dies sind die direkten Wiederholungen, welche möglichst bald erfolgen. Unabhängig davon erfolgt eine zyklische Auffrischung aller bisher benutzten Loks, sofern keine neuen aktuellen Befehle vorliegen.

- **SO# 24** Voreinstellung DCC Format
  Mit dieser SO wird festgelegt, in welchem Format die Loks angesprochen werden.
  - 0: DCC mit 14 Fahrstufen
  - 1: reserviert (DCC mit 27 Fahrstufen, nicht implementiert)
  - 2: DCC mit 28 Fahrstufen (Voreinstellung *)
  - 3: DCC mit 128 Fahrstufen (bzw. 126, um genau zu sein)

  Normalerweise fragen PC-Programme bei der Zentrale nach, in welchem Format die jeweilige Lok angesprochen werden soll. OpenDCC meldet alle Loks standardmäßig mit dem hier angegebenen Format zurück. Soll eine Lok anders angesprochen werden, so muß für diese Lok eine Ausnahmeregel definiert werden. Dies geschieht entweder über XLokCfgSet oder über CV40 (und fortfolgende).
  Empfohlen ist die Einstellung 28 Fahrstufen - 128 Fahrstufen verursachen weniger Performance auf dem Gleis und werden oft auch nicht vom PC-Programm ausgenutzt. Z.B. benutzt TC[7] offenbar nur 50 Fahrstufen.

- **SO# 25** Voreinstellung BiDi (railcom)
  Mit dieser SO wird festgelegt, ob OpenDCC die Austastlücke (cutout) für NMRA-BiDi (RailCom™) erzeugt.
  - 0: Kein Cutout (Voreinstellung *)
  - 1: Mit Cutout

  Die Austastlücke wird 30µs nach dem letzten Bit der vorausgehenden Nachricht erzeugt und ist 434µs lang. Während der Austastlücke sind die beiden DCC-Ausgänge kurzgeschlossen. Nach der Austastlücke folgende 13 Präambelbits. Wenn keine Austastlücke erzeugt wird, so werden 14 Präambelbits ausgegeben.

- **SO# 29** Xpressnet Feedback Mapping
  Mit dieser SO wird festgelegt, wie Weichen und Rückmelder am Xpressnet behandelt werden. Wegen der Einschränkungen des Protokolls ist hier eine Unterscheidung nötig.
  - 0: Schaltdekoder (Weichen) mit einer Adresse ¡ 256 werden komplett mit Abfragebefehl und Broadcastmeldung am Xpressnet behandelt. Die S88-Rückmelder werden beginnend mit Adresse 65 (Bitadresse 513) als Broadcast gemeldet.
  - 1: reserviert (z.Z. nicht implementiert): Alle Adressen sind Rückmeldern zugeordnet.
  - 2: reserviert (z.Z. nicht implementiert): Alle Adressen sind Schaltdekodern zugeordnet.

- **SO# 30** Timing S88
  Mit dieser SO wird festgelegt, wie lang die Taktpulse am S88-Bus sein sollen. Die Einheit ist 10µs; Beim Lesen am S88-Bus wird jeweils ein Puls erzeugt, dessen High und Lowzeit hier vorgegeben wird.
  Voreinstellung 2, also 20µs High und 20µs Low.
  Hintergrund: manche Module sind mittels eines Prozessors implementiert und sind je nach Implementierung nicht in der Lage ein schnelles Auslesen zu unterstützen. Z.B. hat sich 2 als zu schnell für die Module von railwaylauf.de erwiesen.
  Diese Zahl sollte nicht zu groß gewählt werden, da die Zeiten mittels busywaiting implementiert sind und bei Werten über 10 der Durchsatz von OpenDCC absinken kann.

- **SO# 31** Anzahl Weichenrückmelder im S88
  Anzahl der Bytes auf dem (virtuellen) S88-Strang der Weichenrückmelder. Je 8 Weichenrückmelder benötigen 1 Byte, d.h. die höchste vorkommende (Weichenadresse / 8) ist hier einzutragen. Es ist zu beachten, dass der S88-Puffer nur 1024 Bits groß ist, d.h. es können nicht alle möglichen Weichenadressen zurückgemeldet werden.
  Diese Variable kann auch mit X88PGet (0x9C-0x04) abgefragt und mit X88PSet (0x9D-0x04) gesetzt werden.
  (Voreinstellung 0

  Allgemeine Erläuterungen zu den S88-Melderbit:
  Es gibt insgesamt 4 Quellen für Melderbits: 3 echte Hardwareränge und die Weichenrückmeldung. Die drei Hardwareränge werden nahtlos hintereinander in die Melder eingefügt, die Weichenrückmeldung mit dem in CV 16 eingestelltem Offset. Die Summe dieser Zahlen wird bei der Abfrage mittels X88PGet (0x9C-0x00) zurückgeliefert. Sollte jedoch zuvor mit X88PSet (0x9D-0x00) bereits eine größere Zahl an Melder definiert worden sein, so wird die größere Zahl zurückgeliefert.

- **SO# 33** I2C vorhanden
  Ein angeschlossenes I2C-Device (wie z.B. Märklin Keyboard) kennt OpenDCC nicht, daher wird diese Anfrage von railware mit 0 beantwortet.
- **SO# 34** Abschaltzeit bei Kurzschluß Hauptgleis
  Hier wird die Zeit eingestellt, die OpenDCC nach Erkennen eines Kurzschlusses bis zum Abschalten verwendet.
  Voreinstellung 3
  Einheit: 5ms; Die Voreinstellung von 3 bedeutet also Abschalten nach 15ms.

- **SO# 35** Abschaltzeit bei Kurzschluß Programmiergleis
  Hier wird die Zeit eingestellt, die OpenDCC nach Erkennen eines Kurzschlusses bis zum Abschalten verwendet.
  Voreinstellung 8
  Einheit: 5ms; Die Voreinstellung von 8 bedeutet also Abschalten nach 40ms.

- **SO# 36** Externe Abschaltung erlaubt
  Hier kann eingestellt werden, ob OpenDCC über den externen Eingang (Ctrl In) abgeschaltet wird.
  - 0: kein Abschalten über diesen Eingang (Voreinstellung *)

  - 1: bei Anliegen einer Spannung von mind. 5V wird der Gleisausgang abgeschaltet - nur bei regulärem Betrieb, nicht bei Programmiermodus. Die Zentrale bleibt nach dem Abschalten in diesem Zustand und wird entweder über Taster oder PC wieder freigegeben. Zugleich wird bei abgeschalteter Zentrale der Ctrl-Out aktiviert - damit kann eine Abschaltmeldung als Open Collector Verknüpfung verdrahtet werden.

Hinweis: Das kann naturgemäß nicht zusammen mit der Weichenrückmeldung oder DMX funktionieren, da die Ein-und Ausgänge nur einmal verwendet werden können.

- **SO# 39** Seriennummer
  Hier kann eine Seriennummer hinterlegt werden, damit lassen sich verschiedene Boxen an einem PC unterscheiden. Sonst hat diese Angabe keine Bedeutung.
  (Voreinstellung 1)

- **SO# 40** Lok mit abweichendem Format (low)
  ... bildet gemeinsam mit SO41 einen 16 Bit Wert

- **SO# 41** Lok mit abweichendem Format (high)
  ... bildet gemeinsam mit SO40 einen 16 Bit Wert, dieser wird wie folgt interpretiert:

Bits	Bedeutung
0-13	Lokadresse der Lok, die ein anderes Format verwendet
14,15	00 = DCC14
	01 = DCC27 (not impl.)
	10 = DCC28
	11 = DCC128

Leichter und empfohlen ist allerdings der Zugriff über XLokCfgSet (0x86) (siehe Kapitel 3.2.5 auf Seite 55), da erledigt OpenDCC auch die Speicherverwaltung dieses Ausnahmenspeichers.

- **SO# 42** Lok mit abweichendem Format (low)
  ... bildet gemeinsam mit SO43 einen 16 Bit Wert

- **SO# 43** Lok mit abweichendem Format (high)
  ... bildet gemeinsam mit SO42 einen 16 Bit Wert

- **usw.** 64 SO-Paare

3.3.3 wichtige Compile-Optionen

- #define PARSER
  INTELLIBOX oder LENZ: definiert das Protokoll zum Host, Intellibox[19] ist default.
- #define SHORT_TURNOFF_TIME
  15ms, Zeit, bis die Kurzschlußabschaltung anspricht. Diese wirkt zusätzlich zur Strombegrenzung.
- #define DMX_ENABLED
  0: kein DMX (default)
  1: inkl. Code für DMX
- #define STORE_TURNOUT_POSITIONS
  0: Weichenpositionen werden nicht gespeichert.
  1: Weichenpositionen werden gespeichert (default).
- #define TURNOUT_FEEDBACK_ENABLED
  0: normaler S88-Betrieb (default)
  1: Zusätzlicher Code, um statt externer S88 Ereignisse Weichenlagen zurück zu melden (max. 1024 und sofern aktiviert)
- #define TURNOUT_FEEDBACK_ACTIVATED
  0: Die Weichenrückmeldung ist nicht aktiviert.
  1: Die Weichenrückmeldung ist aktiviert.
- #define DCC_DEFAULT_FORMAT
  DCC14
  DCC28(default)
  DCC128
  Wenn eine neue Lok aufgerufen wird, dann wird sie in diesem Format angesprochen.
- #define XPRESSNET_ENABLED
  0: ohne Xpressnet Code
  1: mit Xpressnet, inklusive Handling beider Interfaces (PC und Xpressnet) parallel.

3.4 Konfiguration mit Trainprogrammer

3.4.1 Überblick - Programmierbefehle

Die Zentrale OpenDCC bietet die Möglichkeit, eine Reihe von Parametern mittels EEPROM Variablen zu verändern. Dies kann entweder über die zugehörigen IB-

Befehle erfolgen oder (ab V0.14) über 'umgelenkte' Programmierbefehle. Nachfolgend ist die Konfiguration mittels TrainProgrammer™[5] beschrieben.

Bei diesem Programm können eigene Decoder-Beschreibungen importiert werden - diese Beschreibung wird dann als Menu zur einfachen Konfiguration des Dekoders dargestellt. Der große Vorteil liegt darin, dass man nicht mehr einzelne CV-Adressen raussuchen muß. Auch Variablen, die über mehrere CVs hinweg gehen, lassen sich mit nur einer Eingabe definieren.

Ich habe für die Zentrale OpenDCC eine Datei für TrainProgrammer[8] erstellt. (Download)

3.4.2 Aktivierung des Konfigurationsmodus

Drückt man während des Einschaltens von OpenDCC die 'GO'-Taste (grün), so werden in Folge alle DCC Befehle für byteweises Schreiben und Lesen auf den internen Konfigurationspeicher umgelenkt. Generell gilt, dass OpenDCC nach einem Verändern dieser Variablen neu gestartet werden muß, damit die Änderungen wirksam werden.

[5] TrainProgrammer ist ein Warenzeichen der Firma: Freiwald Software, Kreuzberg 16 B, 85658 Egmating von Jürgen Freiwald - http://www.freiwald.com - 21.12.2007

3.4.3 Menus in TrainProgrammer

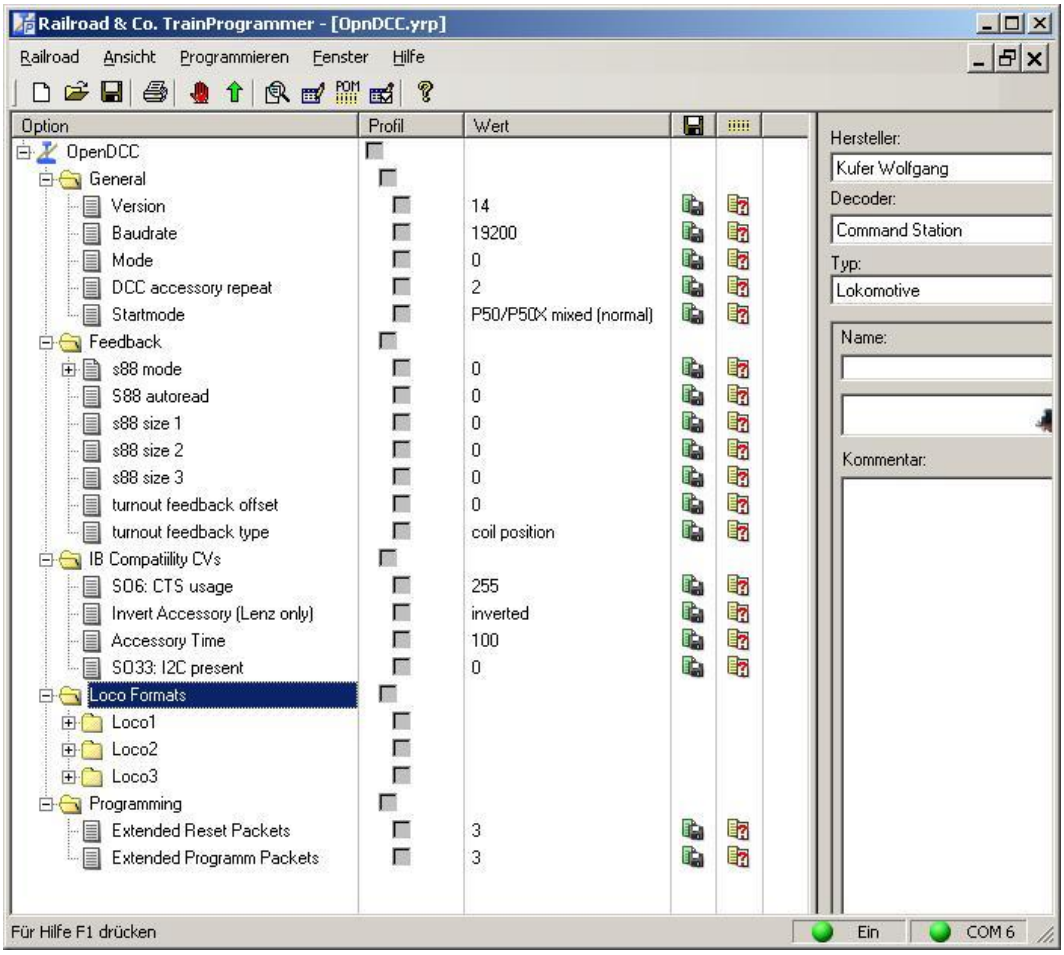


Abb. 5: TrainProgrammereinstellungen (Quelle: archiv - Screenshot der Software Trainprogrammer (Fa. Freiwald))

Die verschiedene Bereich der Zentrale werden in verschiedenen Ordnern gegliedert:

- **General**
  Hier finden sich Version, Baudrate, Startmode usw.
- **Feedback**
  Dieser Ordner enthält alle Konfigurationseinstellungen für die Rückmeldung, entwder per S88 und/oder als Weichenrückmeldung (siehe Kapitel 3.5 auf Seite 93)
- **IB compatibility CVs**
  Das sind CVs, die von diversen PC-Programmen abgefragt werden. Hier

sind Werte hinterlegt die das Verhalten von OpenDCC bestmöglich beschreiben, damit PC-Programme von den korrekt Annahmen ausgehen. Insbesondere railware[27] frägt ziemlich viele Sonderoptionen ab.

- **Programming**
  Hier sind Sonderoption für das Programmieren. (erweiterte Timingeinstellungen).
- **Loco Formats**
  Hier sind von der default-Einstellung abweichende Lokformate gespeichert; *Leichter und empfohlen ist allerdings der Zugriff über XLokCfgSet (0x86) (siehe Kapitel 3.2.5 auf Seite 55), da erledigt OpenDCC auch die Speicherverwaltung dieses Ausnahmenspeichers für Lokformate.*

3.5 Weichenrückmeldung

3.5.1 Einleitung

Beim Schalten von Weichen ist immer die Gefahr gegeben, dass eine Weiche nicht schaltet. Die Folgen sind in einem automatisierten Betrieb fallweise heftig, da speziell Digitalsteuerungen normalerweise keine Möglichkeit kennen, einen Irrläufer abzubremsen - die Lok bekommt unabhängig von ihrem Ort die Fahrstufe, auch wenn sie z.B. wegen nicht schaltender Weiche auf dem Gegengleis unterwegs ist.
Zuverlässige Weichenantriebe sind daher (neben einer funktionierenden Belegtmeldung) das A und O beim Anlagenbetrieb. Nun, Vertrauen ist gut, Kontrolle ist besser - daher haben wir im Verein nach entsprechenden Erfahrungen beschlossen, die tatsächliche Lage der Weiche zu überwachen.

Erste Voraussetzung ist die Erfassung der Lage der Weichen - hier gibt es verschiedene Möglichkeiten (siehe www.opendcc.de - opendecoder) .

Weiters muß ein Rückkanal vom Weichendecoder in die Zentrale geschaffen werden - dies kann entweder über ein bidirektionales Bussystem (Sx-Bus, Loconet, MR-Bus, CAN-Bus) oder über die geplante Erweiterung von DCC (BiDi) geschehen. Auch eine Einspeisung in ein vorhandenes Rückmeldesystem (z.B. S88) wäre denkbar. Sieht man sich jedoch die notwendige Informationsmenge in Rückkanal an (max. 20 Baud, "ja, ich habe geschaltet"), so müßte auch eine einfache Ringleitung reichen, die im Multiplexbetrieb den Decodern zugeteilt wird.

Nachdem mir die Kosten und der Implementierungsaufwand für einen Bus zu hoch waren, entschied ich mich für die Ringleitung.

Bei dieser Ringleitung gibt es zwei verschiedene Implementierungstechniken:

1. klassischer OpenCollector-Ring, d.h. ein zentraler Pullup und die Decoder legen in dem ihnen zugeteilten Zeitschlitz den Ring auf Null (oder eben nicht).
   Nachteil: man braucht diese Leitung
   Vorteil: kein Einfluß durch Booster o.ä.
2. Rückwirkung auf der schon vorhandenen DCC-Leitung - z.B. durch Lasterhöhung oder Rückspeisung.
   Nachteil: diese Rückwirkung muß auch über Booster hinweggeführt werden, kann durch Verbraucher mit schwankender Stromaufnahme gestört werden.
   Vorteil: gleiche Leitung

Hier wird der Sicherheit halber die separate Leitung gewählt.

3.5.2 Wirkungskette

Wie funktioniert nun die Sicherung gegen Falschfahrten? Hierzu ist folgender Ablauf notwendig:

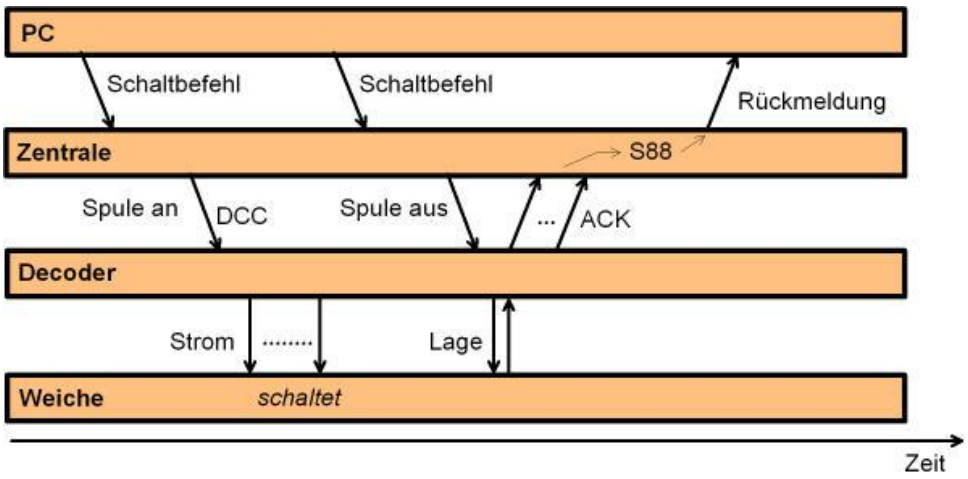


Abb. 6: Feedback-Ablauf (Quelle: archiv)

1. Die PC-Software stellt einen Fahrweg, dabei sei eine Weiche beteiligt. Sicherheitshalber werden die Ausfahrten aus den Gleisen, welche auf die Weiche zulaufen, gesperrt.
2. Die PC-Software sendet den Weichenschaltbefehl an die Zentrale.

3. Die Zentrale stellt das Rückmeldebit dieser Weiche auf **falsch** und sendet den Weichenschaltbefehl (Spule ein) an den Decoder, dieser wird vom Decoder erkannt und ausgeführt.

4. Nach der eingestellten Schaltzeit sendet die Zentrale den Abschaltbefehl für die Spule.

5. Der Decoder schaltet die Spule aus und prüft die Lage der Weiche. Der Decoder ist so programmiert, dass er einen kurzen Quittungspuls (=ACK) zurückliefert, wenn die Weiche jetzt in der richtigen Lage liegt.
(Die Lageprüfung durch den Decoder kann entweder ein Schalter an der Stellschwelle oder die Auswertung der integrierten Endabschaltekontakte sein)

6. Die Zentrale kontrolliert während des Aussendens der nächsten Präambel, ob der Quittungspuls für den letzten Befehl vorliegt. Davon abhängig wird das Rückmeldebit (S88) dieser Weiche wieder auf **richtig** gesetzt.

7. Die PC-Software kann nun alle Rückmeldebits der betroffenen Weichen auswerten und entsprechend reagieren, z.B. die Ausfahrt aus zulaufenden Gleisen auf die fragliche Weiche wieder freigeben.

3.5.3 Externe Verdrahtung

Die Verdrahtung ist recht einfach, zusätzlich zur normalen Verbindung per DCC werden alle Rückmeldeausgänge der OpenDecoder zusammengeschaltet und mit dem CTRL-Eingang von OpenDCC verbunden. Derjenige Decoder, welcher die Quittung sendet, zieht mit seinem Ausgang die Ringleitung auf GND, dadurch fließt Strom und der Optokoppler im CTRL-Eingang von OpenDCC schaltet durch. Sinnvollerweise soll diese Rückmeldeleitung als verdrillte Leitung ausgeführt werden.

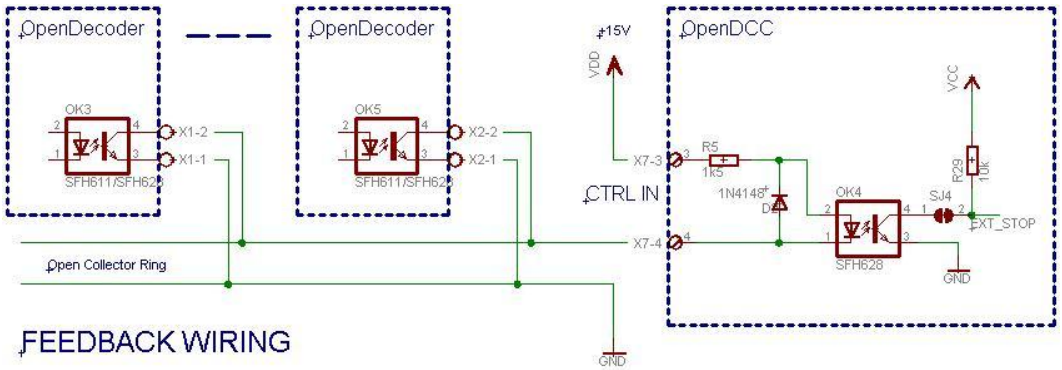


Abb. 7: Feedback-Verschaltung (Quelle: archiv)

(Hinweis: auf der Platine von OpenDCC muß die Lötbrücke SJ4 auf der Platinenunterseite geschlossen sein, damit die Rückmeldung auch am Prozessor ankommt.)

3.5.4 Software-Integration in die vorhandenen Module

Die Aufgaben der DCC-Zentrale in obiger Wirkungskette werden wie folgt auf die Module abgebildet:

Beim Empfang eines Weichenbefehles wird bereits im Parser das Rücksetzen des Meldebits veranlaßt. Der Befehl durchläuft dann wie üblich die weiteren Schichten bis zur Gleisausgabe. Es gibt zwei mögliche Positionen für dieses Rückmeldebit: im S88-Modul (als Ersatz für eine Gleisbesetzmeldung) und/oder im Speicher der Weichenpositionen.

Handelt es sich beim Weichenbefehl um einen Abschaltbefehl, wird auf der Ebene organizer dem Befehl zusätzlich die Eigenschaft is_feedback angeheftet; der Schaltbefehl wird aber ansonsten ganz normal in den Queues verarbeitet.

Auf der Ebene dccout wird bei Befehlen mit der Eigenschaft is_feedback die Adresse der Weiche ausgekoppelt und am Ende der nächsten Präambel wird die Rückmeldeleitung abgefragt. Dieses Feedbackereignis wird als Nachricht an die Module s88 und organizer gemeldet.

Beim nächsten Scan nimmt das S88-Modul das gemeldete Ereignis in die Rückmeldebits auf, wo es dann vom Steuerprogramm im PC erkannt und ausgewertet werden kann. Die Kodierung des Rückmeldebits wird in S017 festgelegt.
Um die Zuordnung zwischen Weichennummer und Rückkanal einfach zu gestalten, sollten exakt die gleichen Nummern für die Weiche sowie für das zugehörige S88-Bit verwendet werden. Deshalb muß bei aktivierter Weichenrückmeldung der normale S88 Betrieb unterbleiben, da sich sonst die Meldungen überlagern würden. Siehe hierzu die Spezial-Optionen 7 und 16. (Bei gleichzeitigem Betrieb von Weichenrückmeldung und S88 muß der S88-Block immer bei 0 beginnen, der Weichenrückmeldungsblock hat dann einen entsprechenden Offset. (SO16))

3.5.5 Auswertung in Traincontroller

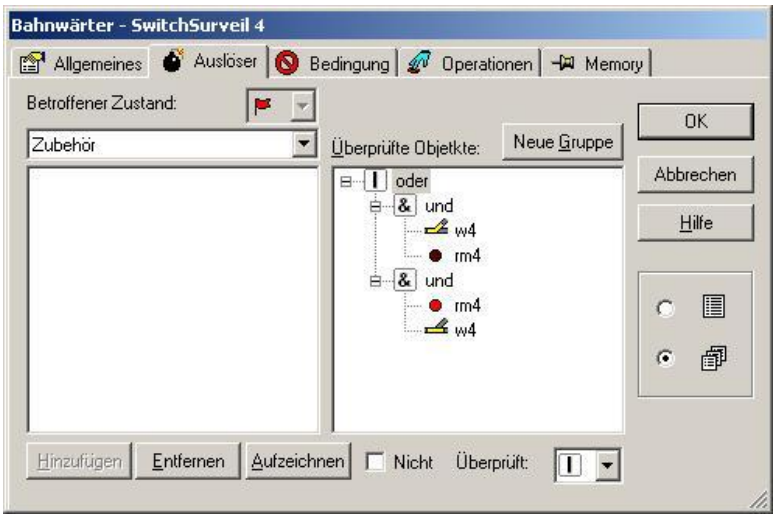


Abb. 8: Auswertung in Traincontroller (Quelle: archiv - Screenshot der Software TrainProgrammer der Fa. Freiwald)

Diese Bild zeigt die Auswertung in Traincontroller[8]: es wird für jede Weiche ein Bahnwärter erzeugt, bei dessen Auslöser jeweils die Stellung der Weiche (hier w4) und die Rückmeldung (hier rm4) eingetragen wird und zwar für beide Richtungen der Weiche. Der Bahnwärter wird aktiv, wenn Schaltstellung und Rückmeldung der Weiche nicht übereinstimmen.

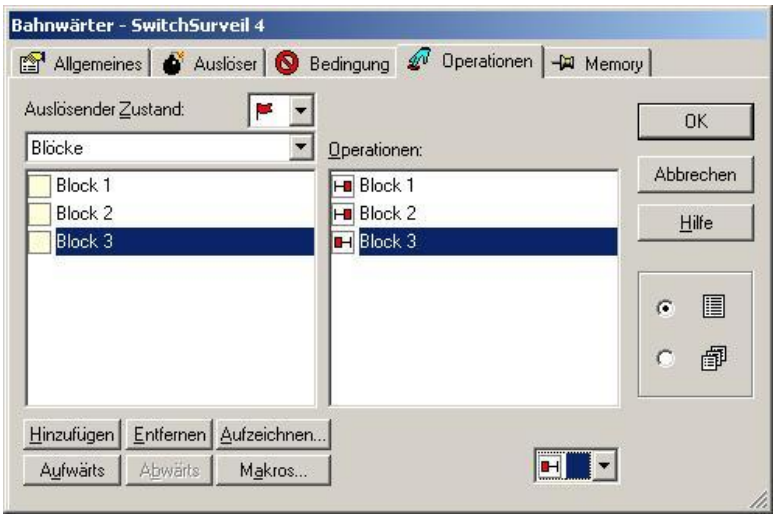


Abb. 9: Aktionen dieses Melders (Quelle: archiv - Screenshot der Software Train-Programmer der Fa. Freiwald)

Bei den Aktionen dieses Melders werden Ausfahrtssperren der benachbarten Blöcke eingetragen, und zwar in der Richtung, die auf die Weiche zuläuft.

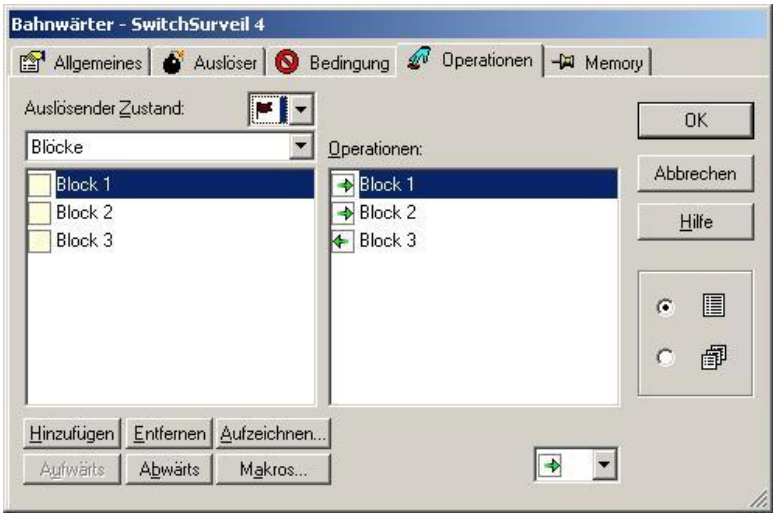


Abb. 10: Sperren (Quelle: archiv - Screenshot der Software TrainProgrammer der Fa. Freiwald)

Diese Sperren müssen bei deaktiviertem Bahnwärter wieder aufgehoben werden. Bei mehreren Weichen in einer Weichenstraße muß ein Sammelbahnwärter gebaut werden, welcher die Überwacher der Weichen verodert und dann das komplette Umfeld der Weichenstraße sperrt.

3.6 DCC Konfigurationsvariablen

3.6.1 Was ist eine CV?

Ein Lokdecoder oder ein Zubehördecoder (Accessory-Decoder) kann im seinem Reaktionsverhalten auf DCC-Befehle mit Konfiguationsvariablen (=CV) eingestellt werden. Darunter fallen neben der Lokadresse und Zahl der Fahrstufen auch die Einstellungen für Beschleuigung, Bremsen, Lichtansteuerung usw. Insgesamt sind 1024 verschiedene CV's vorgesehen, ein Großteil davon ist unbenutzt bzw. herstellerspezifisch belegt.

3.6.2 Wie werden diese CV-Variablen geschrieben?

Es gibt verschiedene Techniken, diese CV's zu programmieren. Diese sind in RP.9.2.3 der NMRA[24] Standards definiert.

- **Adress-Only**
  Hier wird einfach die Adresse umgestellt - d.h. CV1 neu programmiert. Das ist ein historischer Befehl.
- **Register-Programmierung**
  Diese Art der Programmierung erlaubt den Zugriff auf die 8 wichtigsten CVs:

Register	Code	CV, Bedeutung
1	000	CV1: Adresse
2	001	CV2: Anfahrspannung
3	010	CV3: Beschleunigung
4	011	CV4: Bremsverzögerung
5	100	CV29: Konfiguration
6	101	Page: reserviert
7	110	CV7: Version
8	111	CV8: Herstellerkennung

- **Paged**
  Die seitenweise (=paged) Programmierung ist ein Erweiterung der Registerprogrammierung, mit der kompletter Zugriff auf alle CV's möglich ist. Eine Seite ist als ein Paket von vier aufeinanderfolgenden CV's definiert, d.h. CV1, CV2, CV3 und CV4 bilden die Seite 0, CV5 bis CV8 die Seite 1, usw. Bei der Programmierung wird in das Register 6 zuvor die Adresse der Zielseite eingeschrieben. Der Decoder blendet daraufhin diese Seite anstatt der Register 1-4 ein.

- **Direkt Mode**
  Hierbei wird direkt die Adresse der zu schreibenden oder lesenden CV im DCC-Befehl angegeben. Es gibt sowohl Direkt Mode Befehle für den Bytezugriff als für den bitweisen Zugriff.

- **POM** (=Programming on the main, Programmieren auf dem Hauptgleis)
  Die zu schreibende oder lesende CV wird zusammen mit der Lokadresse auf dem normalen Gleis ausgegeben (nicht auf dem Programmiergleis). Der Decoder erkennt die Programmierung an, wenn er zweimal direkt hintereinander den gleichen POM-Befehl empfängt. Normalerweise kann der Empfang

nicht bestätigt werden, die Auswirkung der Programmierung sollte sofort kontrolliert werden.

3.6.3 CV-Variablen für Lokomotiven

Die hier aufgelistetem CV's sind einheitlich in RP.9.2.2 der NMRA[24] festgelegt. In der nachfolgenden Tabelle sind Bits entsprechend ihres mathematischen Gewichts benannt - also Bit 0 ist das LSB, Bit 7 ist das MSB. Manche Dekoderdokumentationen bezeichnen die Bits von 1 bis 8.

Eine übersichtliche Tabelle ist derzeit nur auf der Internetseite vorhanden.

3.6.4 CV-Variablen für Zubehör Decoder

Einfache Zubehördekoder werden i.d.R. mit der Tastermethode programmiert, d.h. Taster drücken, die erste nachfolgend empfangene Adresse ist die einzustellende Dekoderadresse.

Mit mehr Funktionen ausgestattete und damit komplexere Dekoder (wie z.B. OpenDecoder2) benutzen CVs zum Einstellen der Adresse und der Eigenschaften. Auch hier sind bereits eine Folge von Adressen durch die NMRA[24] vordefiniert, weitere (eigentlich *ziemlich* viele) sind reserviert.
Alle CVs für Zubehördekoder beginnen bei CV513 und reichen bis 1024. Der Sinn dieser Festlegung erschließt sich mir nicht ganz - zudem gibt es Zentralen, die nicht in der Lage sind diesen Adressbereich anzusprechen. Dafür sieht die Normung aber vor, dass ein Dekoder fallweise die Adressen der CV um 512 nach unten verlegen darf - also aus CV513 wird CV1, aus CV514 wird CV2 usw. Damit lassen sich dann die Dekoder auch mit kleinem Adressbereich ansprechen. Nachfolgend ist diese zweite Adresse in Klammern angegeben.

Eine übersichtliche Tabelle ist derzeit nur auf der Internetseite vorhanden.

Wie hängen Decoder-Addressing und Output-Addressing zusammen? Hier hilft folgende Graphik zur Veranschaulichung:

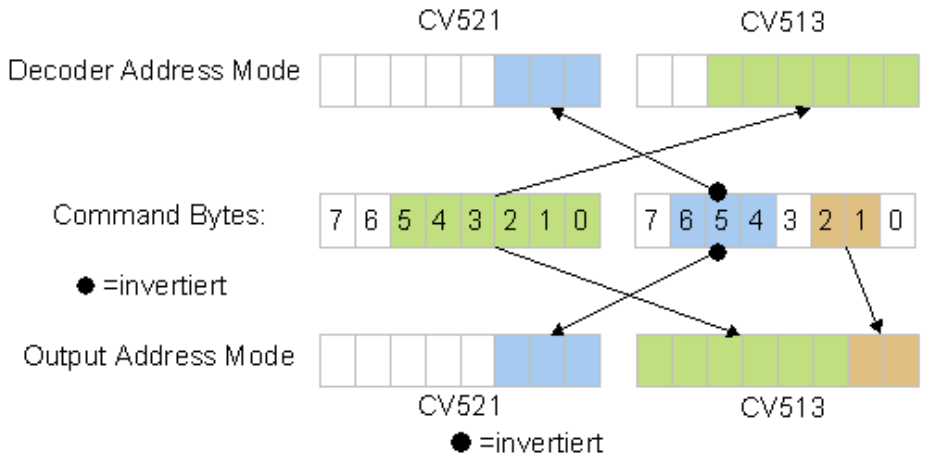


Abb. 11: Decoder-Addressing und Output-Addressing (Quelle: archiv)

3.6.5 Herstellerkennungen

Im folgenden sind einige Herstellerkennungen aufgeführt.

dez	hex	Hersteller
13	0x0D	Öffentliche und Selbstbaudekoder
18	0x12	JMRI
19	0x13	AMW (Arnold Hübsch)
20	0x14	t4t technology for trains
22	0x16	KAM Industries
24	0x18	MoBaTron.de (Clemens)
26	0x1A	MBTronik (Kurt Harders)
28	0x1C	Heljan
40	0x28	Kato
62	0x3E	Tams Elektronik
85	0x55	Uhlenbrock
97	0x61	Doehler & Haas
99	0x63	Lenz
101	0x65	Bachmann Trains
109	0x6D	Viessmann
111	0x6F	Haber & Koenig Electronics
113	0x71	QSI
115	0x73	Dietz Modellbahntechnik
117	0x75	cT Elektronik
123	0x7B	Massoth
129	0x81	Digitrax
131	0x83	Trix
145	0x91	Zimo
147	0x93	Umelec
151	0x97	ESU
155	0x9B	Fleischmann
157	0x9D	Kühn
159	0x9F	LGB
161	0xA1	Roco (Modelleisenbahn GmbH)
165	0xA5	MERG (DIY in UK)
186	0xBA	Brawa

3.7 Software einspielen bzw. Update der Software

3.7.1 Details

OpenDCC kann mit unterschiedlicher Software geladen werden. Hier ist beschreiben, wie die Software im Gerät "upgedated" (wunderbares Denglisch) werden kann. Keine Angst, das ist nicht so kompliziert, wie es auf dem ersten Blick aussieht.

- **Reprogrammierung**
  Selbstverständlich kann man in jeder Situation auch die beim Bau von OpenDCC (siehe Kapitel 4.2 auf Seite 121) benutzten Bootverfahren wie z.B. Ponyprog[3] benutzen.

- **Bootloader**
  Eleganter ist jedoch der Softwareupdate über den Bootloader. Das Gehäuse muß hierzu nicht geöffnet zu werden. Hierzu ist folgendes zu tun:

  1. **Vor dem Einschalten** von OpenDCC die **STOP-Taste** drücken und während dem Einschalten gedrückt halten. Statt der normalen Betriebsart wird nun ein Bootloader gestartet (erkennbar an der dauerleuchtenden CTRL-LED), welcher nun auf eine neue Software vom PC wartet und diese im Programmspeicher hinterlegt.
  2. update_opendcc.bat, opendcc.hex, opendcc.eep und avrosp.exe (siehe unten) in ein (beliebiges) Verzeichnis kopieren und aus der Eingabeaufforderung **update_opendcc COM1** (bzw. mit der richtigen Portnummer) starten. Wichtig: COM1 groß schreiben! Es erfolgt nun der Update.
  3. Falls zusätzlich noch das EEPROM geschriebern werden soll: update_opendcc_eeprom COM1 starten.
  4. OpenDCC neu einschalten - fertig!
  5. Wenn OpenDCC über USB (siehe Kapitel 3.10 auf Seite 109 angeschlossen ist, dann einfach die Portnummer des virtuellen Ports verwenden.

- **So funktioniert**
  Der Bootloader ist ein separates Programm, welches ganz an das Ende des Speichers geladen wird. Durch entsprechendes Programmieren der Konfigurationsbits (sog. Fuses) wird nun der AVR angewiesen, beim Reset zuerst den Bootloader anzuspringen. Dieser prüft die Stoptaste ab und springt auf den normalen Resetvektor 0x0000, wenn die Taste nicht gedrückt ist. Nähere Information finden sich bei Atmel[11] in den Application Notes

AVR109 und AVR911; Auf www.atmel.com darf man jedoch nicht nach diesen App-Notes suchen, sondern muß über das Menu zu den App-Notes navigieren, dann kann man die entsprechenden Quellen runterladen.

Um den Bootloader für OpenDCC zu **erzeugen**, müssen die geladenen Quellfiles angepasst werden. Hierzu ist das file preprocessor.xls zu öffnen und die Einstellungen gemäß Anleitung durchzuführen. Danach kann man den Bootloader mit "make" übersetzen. Unten ist der Download für ein bereits angepassten Bootloader.

Hinweise:
Das makefile nicht mit einem Editor bearbeiten, welcher TABs durch BLANKs ersetzt.
Der Linker rechnet in Bytes, d.h. hier wird die doppelte Zahl wie bei den Fuses eingetragen.

Damit der Bootloader auch angesprungen wird, sind diese Fuses und Einstellungen zu setzen:

- BOOTRST = 0 (0=programmed): damit springt der ATmega nicht nach 0x0000, sondern auf den Bootloader;
- BOOTSZ1 = 0 und BOOTSZ0 = 1: damit wird die Größe des Bootloaders festgelegt. Diese Einstellung bedeutet 1kWorte, d.h. der Bereich 0x0000 bis 0x3BFF ist für die Applikation (93,5%), der Bereich 0x3C00 bis 0x3FFF ist für den Loader.
  (Beide Bits = 0 würden 2kWorte bedeuten, d.h. der Bereich 0x0000 bis 0x37FF ist für die Applikation, der Bereich 0x3800 bis 0x3FFF ist für den Loader.)
- BASEADDR = 0x7800: Diese Anweisung im Makefile des Bootloaders sorgt dafür, das dieser auch an richtige Stelle geladen wird. (Hinweis: der Linker rechnet in Bytes, also: 0x7800 = 2*0x3C00)

- **So wird geladen**
  Auf der PC-Seite muß ein Programm geladen werden, das mit dem Bootloader reden kann. Hier gibt es mehrere Möglichkeiten:

  - avrosp:
    Dieses Tool verwende ich. Es ist bei der AppNote AVR911 dabei, der Aufruf sieht wie folgt aus:
    avrosp -dATmega32 -e -pf -vf -ifopendcc.hex -cCOM3
    Die Baudrate des Ports muß vorher richtig eingestellt sein:
    mode com3 baud=19200 parity=n data=8

- avrdude:
  Dieses Tool ist bei WinAVR dabei, der Aufruf sieht wie folgt aus:
  avrdude -c avr910 -p m32 -P com3 -U flash:w:opendcc.hex -b 19200

- Menüpunkt Tools in AVR Studio[5]; evtl. muß vorher in Options die Zahl der "Number of Comports to try" erhöht werden.

3.7.2 Unterlagen

Hex-File des Bootloaders: bootloader.hex Dieses File muß zur Erstinstallation mit dem ISP-Programmer in den Atmel[11] geladen werden. Bereits adaptierter Source Code hierzu: bootloader.zip

Tool zum Programm und EEPROM-Update von Atmel: AVROSP.EXE

Batch zum Programm-Update: update_opendcc.bat (falls *.bat nicht abrufbar: zip-file) *Hinweis: Einfach opendcc.hex, opendcc.eep, avrosp.exe und diesen Batch in ein Verzeichnis kopieren und den Batch starten. Der Batch erledigt auch die Einstellung des COM-Ports.*
Um einen Batch zu starten, muß man die Eingabeaufforderung bemühen: Leider steht diese nach dem Start auf dem 'home'-Verzeichnis. Einfacher geht es, wenn man dem Kontextmenu des Explorers das Starten der Eingabeaufforderung beibringt. Hier steht, wie es geht.

Batch zum EEPROM-Update: update_opendcc_eeprom.bat (falls *.bat nicht abrufbar: zip-file)

3.8 Notwendige Entwicklungsumgebung

3.8.1 Selbstübersetzen der Software?

Vorbemerkung: Neben der Benutzung der fertig compilierten Software bietet eine Open Source Software auch die Möglichkeit, das System für eigene Bedürfnisse zu optimieren. Nachfolgend sind die Werkzeuge beschrieben, die man dazu braucht. Neben der Veränderung von Grundeinstellungen und Konfigurationen (zumeist Konstanten in config.h oder config.c) kann man natürlich auf tiefer rumwühlen - da sollten dann aber schon Kenntnisse in C vorhanden sein.

- Zuerst mal Infos über die AVR Microcontroller einsammeln. Hier kann man sich zuerst bei Atmel AVR umsehen und auch das entsprechende Forum AVRFreaks oder Mikrocontroller.net ist ein guter Startpunkt. Auf diesen Seiten finden sich Application Notes, Beispiele, Datenblätter und jede Tools. Auf AVRFreaks gibt auch ein gutes Tutorial zum C-Compiler GCC . Außerdem gibt es im Netz der Netze auch jede Menge Open Source, z.B. einen USB-Endpunkt, Open Source RTOS oder auch einen Webring.

- **Entwicklungssoftware**
  Vom der populären GNU Compiler Collection (GCC) gibt es eine Portierung für den AVR (AVR-GCC). Wer Windows benutzt, findet ein komplettes Install bei Sourceforge.net (WinAVR). In diesem Paket ist alles enthalten: kompletter GCC Compiler mit Tools und einem Editor, den Programmers Notepad, der auch eine minimale IDE mitbringt. Für andere Betriebssystem empfiehlt sich ein Blick auf OpenAVR.

  Als Simulator kann ich AVR-Studio[5] von Atmel[11] empfehlen. WinAVR[2] läßt sich auch in AVR-Studio integrieren, so dass eine komplette integrierte Entwicklungsumgebung vorhanden ist. AVR-Studio simuliert auch die Peripherie des Prozessors mit.

  Achtung

  Der GCC-Compiler 3.4.5 enthält einen Bug, welcher z.B. die Funktionsaufrufe bei Lokomotiven stört. Dieser wurde mit der Version 4.1.1 (20070122) behoben. Nur läuft dieser Compiler nicht zusammen mit der alten IDE, man braucht AVRstudio 4.13. Und bevor man das installieren kann, braucht man den neuen Windows Installer 3.1. Also:

  1. Microsoft Windows Installer 3.1.4000.2435 Redistributable installieren.
  2. WinAVR 20070122 installieren
  3. AVRstudio 4.13 (bei http://www.atmel.no/beta_ware/) installieren

- **Hardware**

  Zusätzlich zur Zielhardware (ich verwende dafür das OpenDCC-Board ;-) braucht man für die Erstprogrammierung man noch ein Programmiertool. Ich verwende ponyprog, der entsprechende Stecker ist bereits vorgesehen, aber empfehlenswerter ist die Lösung mit einem zusätzlichen Adapter ; Selbstverständlich kann man auch andere Programmiertools verwenden, ein 6-poliger ISP-Stecker ist vorgesehen. Beispielsweise kann man den Atmel AVRISP programmer kaufen.
  Später wird dann Flash und das EEPROM mit dem Bootloader geladen

werden und der Programmierstecker kann entfallen.

Dies ist die für OpenDCC adaptierte Version (aus Appnote AVR911): bootloader.zip

- **Software**
  Allgemeines Vorgehen: Es wird zuerst in AVR Studio[5] ein neues Project angelegt, Prozessor ist ATmega32 mit 16MHz. Die folgenden Angaben werden im AVR Studio bei **Project →Configuration Options** eingestellt:

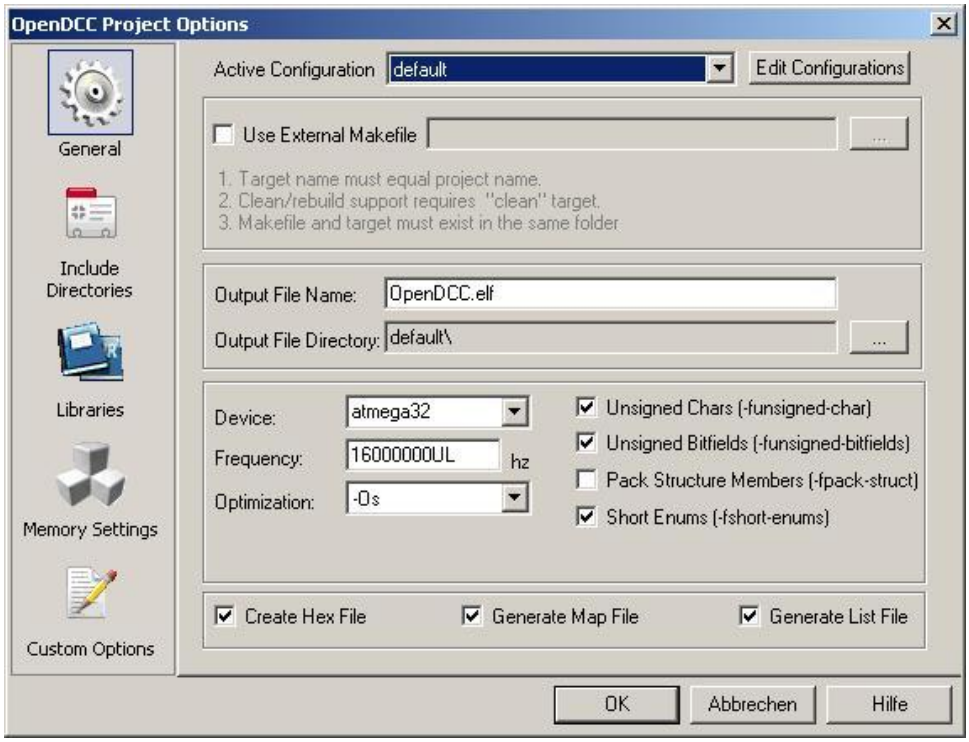


Abb. 12: Projekt Options (Quelle: archiv - Screenshot vom Programm AVRStudio der Fa. ATMEL www.atmel.com)

Als Optimierung ist -Os zu wählen, die Taktrate ist je nach Projekt einzustellen: für OpenDCC 16MHz, für die Decoder 8 oder 10MHz.
Hinweis: AVR Studio erwartet die Eingabe der Taktrate als Zahl (ohne UL), stellt (je nach Version) die eingegebene Zahl dann jedoch mit angehängtem UL (=unsigned long) dar.

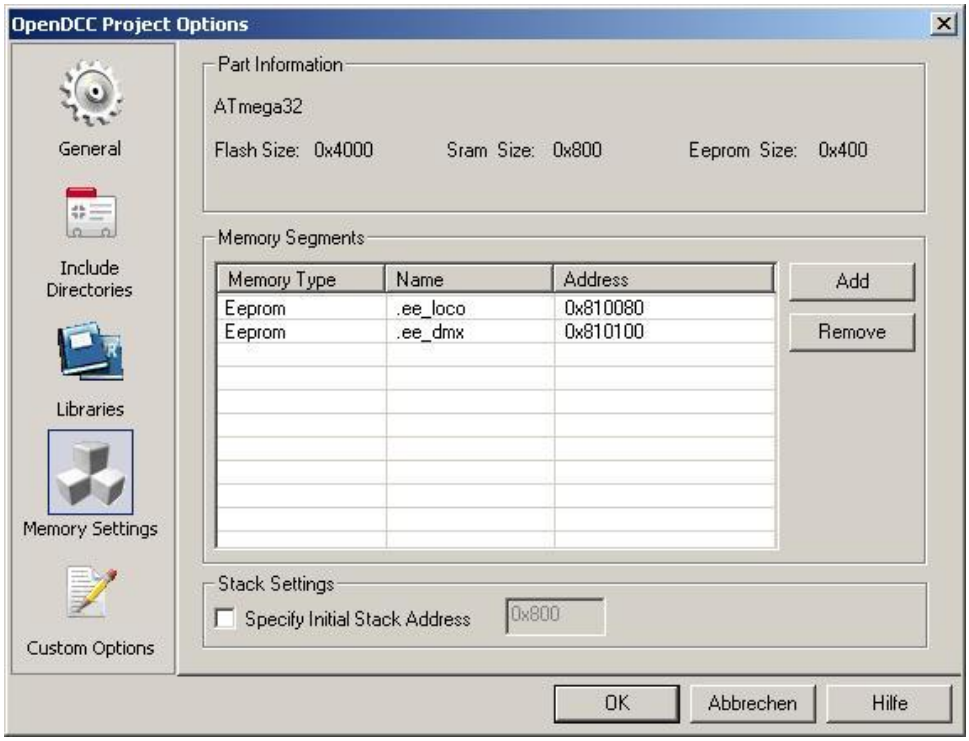


Abb. 13: Speicherbereiche (Quelle: archiv - Screenshot vom Programm AVRStudio der Fa. ATMEL www.atmel.com)

So werden die zusätzliche Speicherbereiche im EEPROM definiert. Diese Definition ist notwendig, um EEPROM Variablen an feste und vordefinierte Adressen zu binden. Leider werden diese Optionen nicht korrekt an WinAVR[2] weitergegeben - deshalb ist speziell bei den Decodern der gesamte Inhalt des EEPROM in einer Struktur abgelegt.

Nun wird das Softwarepaket wird in diese Arbeitsverzeichnis entpackt. Alle header und c-Files werden nun mit "add existing source files" hinzugefügt. Nun sollte mit BUILD (=Taste F7) übersetzt werden können.
Alle wesentlichen Einstellungen sind config.h zu finden.
Hinweise:
Einfacher geht es so: das im ZIP-File enthaltene **OpenDCC.APS** bei 'Project - Open' öffnen, dann sind alle Einstellungen vorhanden.
Das file dccout_without_feedback.c ist eigentlich ein backup von dccout.c und sollte nicht included werden.

3.9 Source Code

Source Code: OpenDCC_V014.zip inkl. HEX und .EEP

3.10 Installation des USB-Treibers

Schritt 1: Einspielen der USB-Konfigurationsdaten in OpenDCC

Damit die weiter unten beschriebenen Einstellungen auch zum Chip in OpenDCC passen, muss dieser vorher (einmalig) entsprechend konfiguriert werden. Das Konfigurieren des Adapters ist nur über die D2xx Treiber möglich, die man von FTDI[14] laden kann. Das Archiv von FTDI[14] wird in Verzeichnis entpackt.

Hinweis 11.07.2007: da gibt es einen neuen Kombitreiber von FTDI[14], mit dem geht es ohne die Umkonfigurieren. Einfach Treiber einspielen und mit mprog[4] Vendor-ID und Produkt-ID einspielen. Die nachfolgende Beschreibung ist daher veraltet.

1. Es dürfen keine VCP-Treiber[29] installiert sein. Sind schon den VCP-Treiber installiert, dann müssen diese zuerst wieder deinstalliert werden.
2. Jetzt OpenDCC mit dem PC verbinden. Eine neue Hardware wird erkannt, zu der es nun keinen Treiber gibt.
3. Für diese neue Hardware werden nun die D2XX-Treiber installiert. OpenDCC sollte nun als Eintrag unter den USB-Devices in der Systemsteuerung erscheinen.
4. Nach dem Installieren de D2XX-Treiber wird mprog[4] installiert und gestartet.
5. Im "mprog" [4] lädt man folgende Datei (EEPROM template: OpenDCC_V1.2.ept).

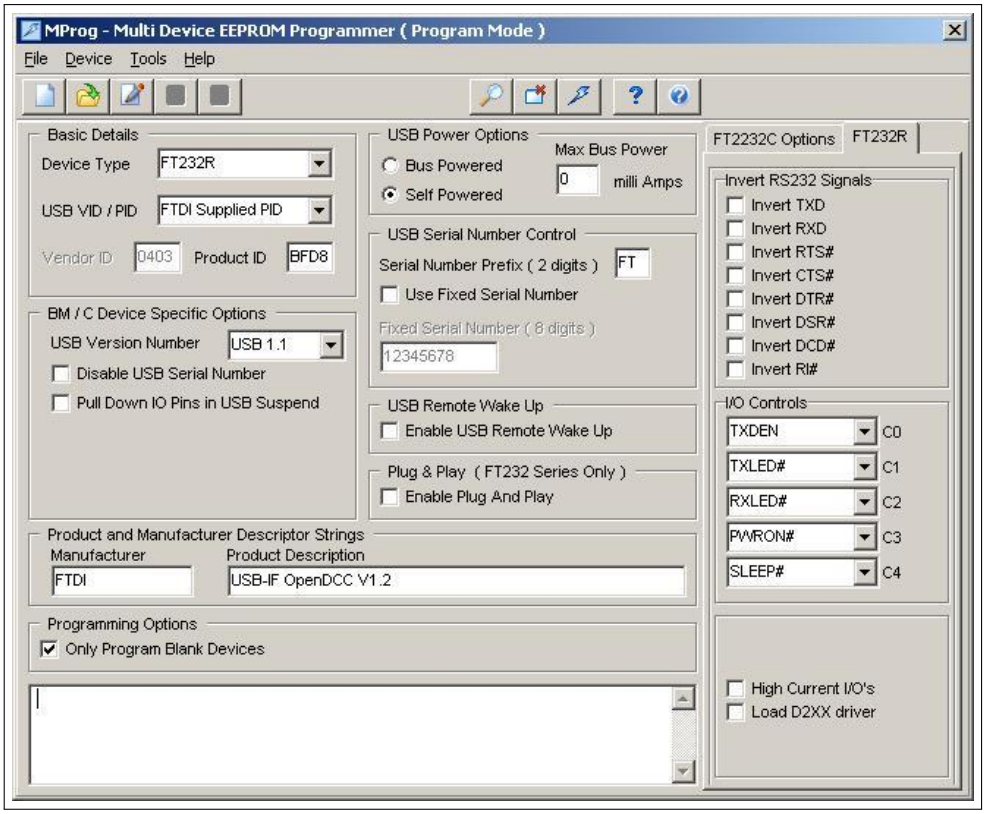


Abb. 14: mprog von FTDI (FTDI, http://www.ftdichip.com)[14]

Damit sind alle notwendigen Einstellungen bereits vorgenommen. Nun kann man mit der "Lupe" die verfügbaren Devices suchen und durch Drücken auf das Blitzsymbol wird der USB-Chip und das EEPROM konfiguriert.

6. Jetzt den D2XX Treiber wieder deinstallieren, damit der VCP[29] (siehe unten) installiert werden kann.

Schritt 2: Einstellung in der Hardware von OpenDCC

Um den USB-Port zu benutzen, müssen die entprechenden Lötbrücken (SJ = Solder Jumper) oder Steckbrücken (JP = Jumper) auf der Hardware geschlossen werden:

Layout V1.3

SJ2: offen
SJ3: geschlossen
SJ5: 2-3 geschlossen (2-3 ist die Seite zum USB-Chip hin)

Layout V1.4

JP5: 2-3 geschlossen
JP6: 2-3 geschlossen

Damit ist der Atmel[11] mit den USB-Chip und nicht mit dem RS232-Chip verbunden.

Schritt 3: Treiber am PC laden

Zur Nutzung der USB Schnittstelle von OpenDCC ist ein VirtuellerComPort (VCP) Treiber[29] nötig. Dieser emuliert eine serielle Schnittstelle im PC, auf die dann die Steuerungssoftware zugreifen kann. Der tatsächliche Datenverkehr wird aber über die USB-Verbindung hergestellt. Diesen VCP-Treiber[29] (und weitere Informationen) findet man bei FTDI[14]. Der auf OpenDCC eingesetzte USB Chip hat die Bezeichnung FD232RL.

Am USB-Bus hat jeder Teilnehmer eine VID und PID (VID: Vendor ID, PID: Product ID, ID=Kennzahl). OpenDCC hat eine eigene PID innerhalb der für FTDI[14] vergebenen VID bekommen:

VID:	0x0403
PID:	0xBFD8
Product Description:	'USB-IF OpenDCC V1.2'

Installation: das ZIP-Archiv von FTDI[14] wird es in ein beliebiges Verzeichnis entpackt. Nun müssen die VID und PID für OpenDCC in drei Dateien dieses Treibers eingetragen werden. Hierzu sind in den Dateien die folgenden kursiven, roten Zeilen zu ergänzen:

```
Ftdibus.inf
[FtdiHw]
%USB\VID_0403&PID_8372.DeviceDesc%=FtdiBus,USB\VID_0403&PID_8372
%USB\VID_0403&PID_6001.DeviceDesc%=FtdiBus,USB\VID_0403&PID_6001

%USB\VID_0403&PID_BFD8.DeviceDesc%=FtdiBus,USB\VID_0403&PID_BFD8

[ControlFlags]
ExcludeFromSelect=USB\VID_0403&PID_8372
ExcludeFromSelect=USB\VID_0403&PID_6001

ExcludeFromSelect=USB\VID_0403&PID_BFD8


[Strings]
Ftdi="FTDI"
DriversDisk="FTDI USB Drivers Disk"
USB\VID_0403&PID_8372.DeviceDesc="USB Serial Converter"
USB\VID_0403&PID_6001.DeviceDesc="USB High Speed Serial Converter"
```

```
USB\VID_0403&PID_BFD8.DeviceDesc="USB-IF OpenDCC V1.2"


WINUN="Software\Microsoft\Windows\CurrentVersion\Uninstall"
FtdiBus.SvcDesc="USB Serial Converter Driver"

Ftdiport.inf
[FtdiHw]
%VID_0403&PID_8372.DeviceDesc%=FtdiPort,FTDIBUS\COMPORT&VID_0403&PID_8372
%VID_0403&PID_6001.DeviceDesc%=FtdiPort232,FTDIBUS\COMPORT&VID_0403&PID_6001


%VID_0403&PID_BFD8.DeviceDesc%=FtdiPort232,FTDIBUS\COMPORT&VID_0403&PID_BFD8



[Strings]
FTDI="FTDI"
DriversDisk="FTDI USB Drivers Disk"
PortsClassName = "Ports (COM & LPT)"
VID_0403&PID_8372.DeviceDesc="USB Serial Port"
VID_0403&PID_6001.DeviceDesc="USB Serial Port"


VID_0403&PID_BFD8.DeviceDesc="USB-IF OpenDCC V1.2"


FtdiPort.SvcDesc="USB Serial Port Driver"
SerEnum.SvcDesc="Serenum Filter Driver"

Uninstaller.INI
=== In File Ftdiunin.ini ===

[Uninstall]


Device=VID_0403&PID_BFD8


InfFiles=FTDIBUS,FTDIPORT,FTSERMOU
Key=FTDICOMM

=== In File Ftdiun2k.ini ===

[Uninstall]


Device=VID_0403&PID_BFD8


Converter=FTDIBUS
Serial=FTSER2K
Key=FTDICOMM
```

Den Treiber mit den bereits durchgeführten Ergänzungen kann man auch gleich hier laden.

Schritt 4: Verbinden

Stellt man nun die Verbindung von OpenDCC und PC her, dann sollte OpenDCC korrekt in der Deviceübersicht in der Systemsteuerung erscheinen.

4 Hardwaredetails

4.1 Schaltungsbeschreibung

4.1.1 Schaltplan, Blattaufteilung

Blatt 1: ATmega, 5V Stabilisierung, JTAG und Programmierschnittstelle
Blatt 2: Booster, DCC und Programmiergleisausgang
Blatt 3: S88-Zusatzplatine
Blatt 4: Tastenplatine und LEDs, DMX-Ausgang
Blatt 5: RS232 und USB Interface

4.1.2 Schaltung

- **Prozessor:**
  Als Herz von OpenDCC pocht ein AVR ATmega32 von Atmel. Dieser Prozessor bietet 32kByte Flash Programmspeicher, 2kByte SRAM und 1kByte EEPROM, sowie Interruptcontroller, Timer und Schnittstellen "on chip". Der ATmega32 wird mit 16MHz getaktet, dies ist auch zugleich die Leitfrequenz, aus der sich alle anderen Timings ableiten. Beim Selbst-Compilieren muß daher unbedingt die richtige Taktrate angegeben werden (F_CPU in config.h bzw. im Project Setting).

- **Spannungsversorgung:**
  An X13 (X2) wird die stabilisierte 15V-Spannung des externen Netzteils eingespeist und zur Versorgung des Doppelvollbrückentreibers L6206 und des 5V Längsreglers 7805 verwendet. Leider gibt es bei Steckernetzteilen ein wenig Wildwuchs, was den Stecker angeht: Hier ist der Innenstift Plus, der Mantel Minus. Statt des Kleinspannungsstecker können auch Schraubklemmen bestückt werden.

  Sollten mehrere S88 Ausgänge verwendet werden, so wird der auf 5V entnommene Strom zu groß für den 7805. In diesem Fall kann statt des 7805 ein Schaltwandler PTH08080 von Texas Instruments bestückt werden. Diese kann bei einem Wirkungsgrad von 85% 2,5A liefern.
  Der externe S88-Eingang ist mit einer Polyfuse gegen Kurzschluß gesichert.

- **Kurzschlußüberwachung:**
  Der Vollbrückentreiber hat eine integrierte Kurzschlußüberwachung, diese wird über die exteren Widerstände R7 und R12 eingestellt. Ein Wert von 39k erlaubt einen Kurzschlußstrom vom ca. 500mA, bei 15k sind es 1.5A.

Beim Programmiergleis wird zusätzlich der Rückstrom aus der Vollbrücke über einen Strommeßwiderstand (0,5 Ohm, Parallelschaltung von zwei 1 Ohm Widerständen) geleitet. Der Spannungsabfall an diesem Widerstand wird für die ACK-Detection verwendet.

- **Gleisausgang, Überschwinger am Ausgangssignal:**
  Prinzipiell ist der Gleisanschluß mit verdrillten Leitungen auszuführen, um Einkopplungen des Ausgangssignals auf andere Schaltungsteile (z.B. Rückmelder, S88) möglichst zu vermeiden. Der Ausgang ist intern mit einer RC-Kombination kompensiert, um Signalüberschwinger zu vermeiden.
  Sollten am Gleis trotzdem noch Signalüberschwinger auftreten, so kann das zusätzliche RC-Glied etwas größer dimensioniert werden (größerer Kondensator).
  Die Signalüberschwinger haben Ihre Ursache in Reflexionen auf der Gleiszuleitung. Diese lassen sich vermeiden bzw. reduzieren wenn man den Ausgang zusätzlich durch ein RC-Glied filtert. Bewährt hat sich eine Kombination aus Längswiderstand und Querkondensator mit den Werten R3=R4=0,15Ohm und C9, C10=100nF bis 200nF. C10 sollte nicht über 200nF gewählt werden, weil sonst nach dem Polaritätswechsel des DCC-Signals die Kurzschlußschaltung anspricht. Der Widerstand (R3 bzw. R4) sollte eine max. Verlustleistung von 1W haben. Der Ruhestrom steigt dadurch allerdings um ca. 30-60mA (je nach Kondensator) an.

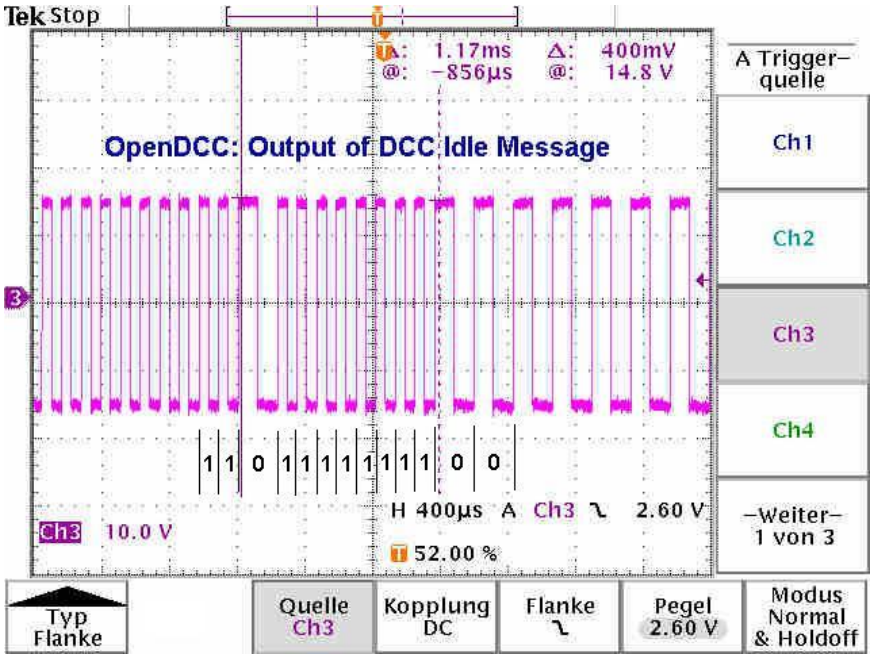


Abb. 15: Das Ausgangssignal von OpenDCC (Quelle: archiv)

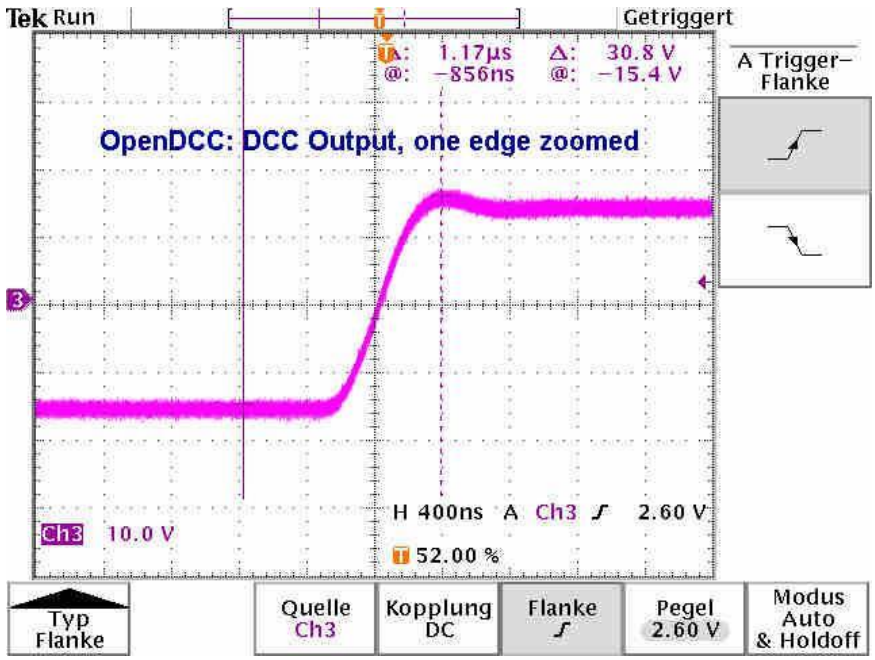


Abb. 16: Zoom des Ausgangssignals von OpenDCC (Quelle: archiv)

- **ACK Detection:**

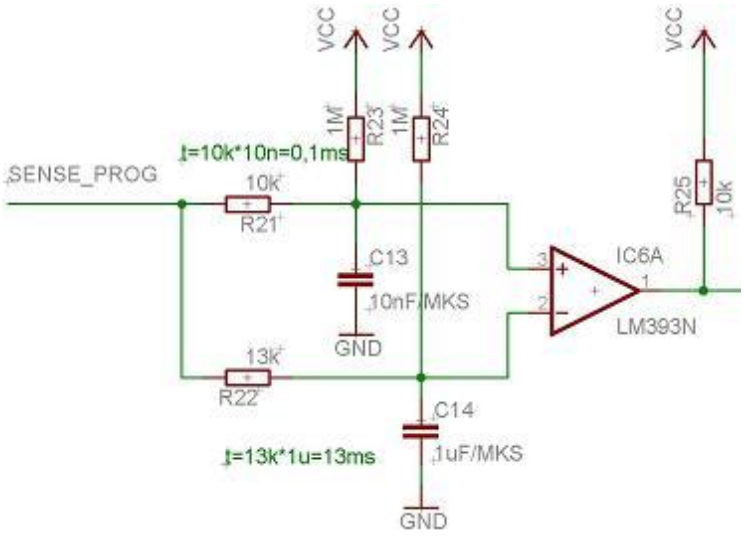


Abb. 17: Vorschlag "Acknowledge" (Quelle: Martin Pischky)

Gemäß RP 9.2.3 soll ein Lokdecoder im Service-Mode ein Acknowledge zurücksenden. Dies geschieht dadurch, dass die Stromaufnahme kurzzeitig

um 60mA erhöht wird. Diese Erhöhung verursacht am Sense Pin des Programmierausgangs einen Spannungsabfall. Nach einem Vorschlag von Martin Pischky wird der Spannungsabfall auf zwei verschiedene RC-Glieder geführt. Normalerweise ist die Spannung am negativem Eingang des Comparators um ca. 11mV höher. Bei einer Spannungsänderung braucht das untere Glied deutlich länger, um der Spannung zu folgen. Der Komparator detektiert daher einen kurzen positiven Puls beim Anstieg der Spannung. Dieser Puls löst einen Interrupt (aktiv low) am Prozessor aus.

Eine Hochpassschaltung nach ...
http://www.geocities.com/CapeCanaveral/Lab/2459/ACKDET02.HTM [6]
... wäre auch gegangen. Den internen Komparator des ATmega habe ich nicht genommen, weil dieser mit einem max. Offset von 40mV gesegnet ist.

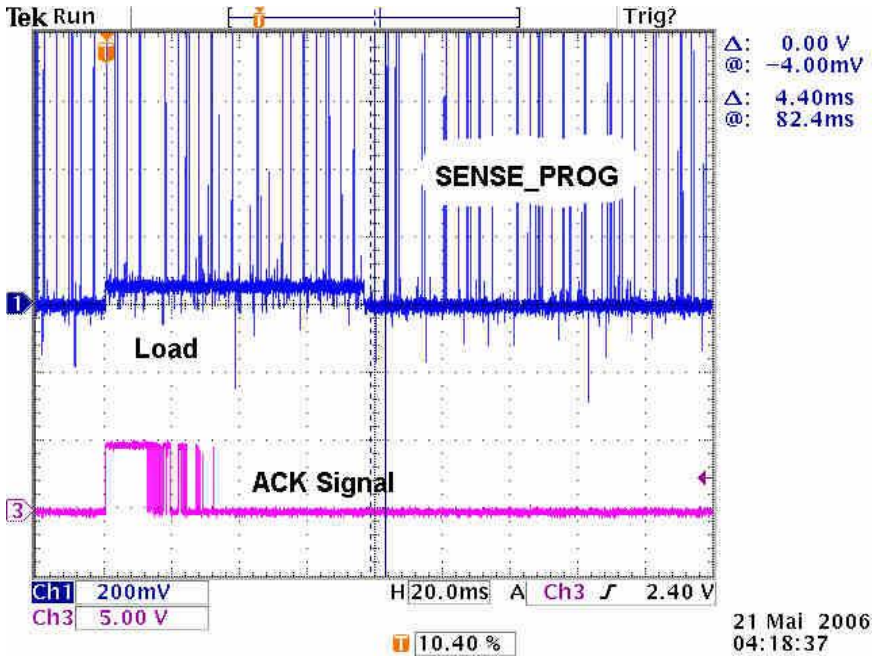


Abb. 18: 1. Darstellung von "Acknowledge detection" (Quelle: archiv)

[6] Copyright 1997 by Martin Pischky (martin.pischky@fernuni-hagen.de)
http://www.geocities.com/CapeCanaveral/Lab/2459/ACKDET02.HTM - 06.01.2008

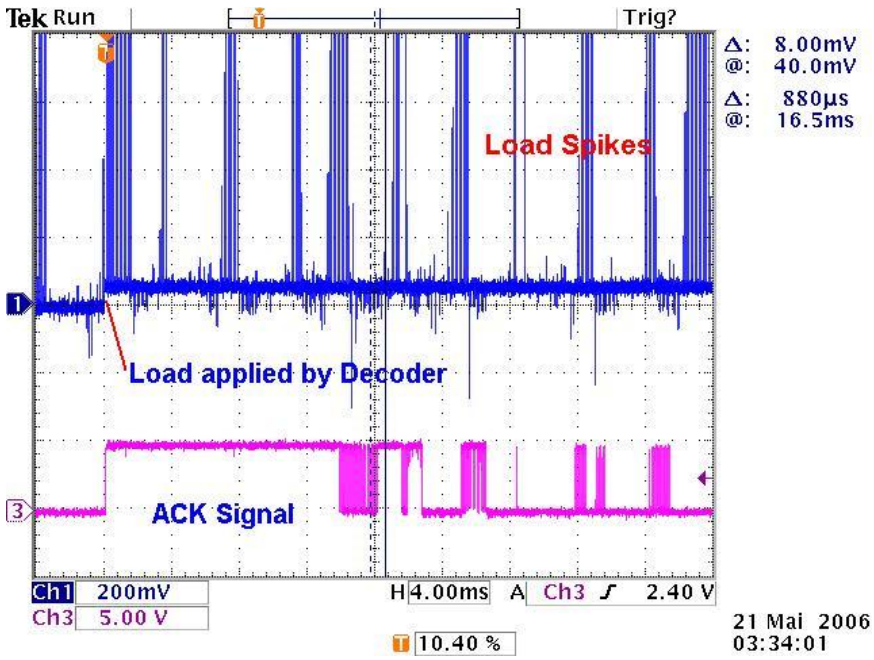


Abb. 19: 2. Darstellung von "Acknowledge detection" (Quelle: archiv)

Hier sind die Signale zu sehen; oben das Signal SENS_PROG. Bei jedem Wechsel des DCC-Signals fließt kurzzeitig ein hoher Strom. Diese werden durch einen (zusätzlichen) 10nF Kondensator am Eingang des ACK-Komparators 10nF zwischen Pin 2 und 3 weggefiltert. Unten das Signal ACK_DETECT, wobei deutlich der ca. 14ms lange ACK-Puls zu sehen ist; Wenn die o.g. RC-Glieder sich auf die Last neu einpegeln, kommt es zu Überlagerungen der Schaltpulse, deswegen ist die Rückflanke des ACK_DETECT Signals so "ausgefranst". ACK wird ab einer Stromerhöhung von 50mA erkannt (lt. Norm 60mA).

- **PC Schnittstelle:**
  Hier wird der übliche Max232 oder LT1081 zur Pegelwandlung verwendet. Neuere Varianten brauchen i.d.R. kleinere oder keine Kondensatoren mehr, in diesem Fall einfach die C's nicht bestücken. Im Layout sind neben den Pins 1 und 16 zwei Vias positioniert. Damit ist an dieser Stelle auch ein LT1080 bestückbar. Dieser ist an sich pinkompatibel, hat jedoch ein 18-poliges Gehäuse.
  Eine direkte USB-Schnittstelle (siehe Kapitel 3.10 auf Seite 109) ist alternativ über den Chip FT232RL möglich. Dieser Chip emuliert eine RS232-Schnittstelle am USB-Bus; das sekundäre Interface entspricht RS232, allerdings mit TTL-Pegeln, so dass eine direkte Verbindung zum ATmega möglich ist. Dieser Chip samt zugehöriger Steuerwiderstände ist auf der Platine in

SMT (SO28) ausgeführt. Sollte die USB-Schnittstelle verwendet werden, so ist der USB-Treiber von www.ftdichip.com zu verwenden die beiden Jumper JP5 und JP6 in Position USB zu stecken, ansonsten müssen beide Jumper in Position RS232 stecken. Wer nur ein Interface hat, kann die Jumper JP5 und JP6 durch Drahtbügel ersetzen.
Für die V1.3 sind diese Jumper noch als Lötbrücken ausgeführt, bitte Stromlauf und Bestückungszeichnung beachten.

- **Tastatur und LEDs:**
  Als LEDs sind 2mA Typen zu verwenden, diese können direkt mit dem ATmega angesteuert werden. Die LEDs kann man alternativ auf der Hauptplatine (dann nach vorne umgebogen) oder auf der Frontplatine bestücken. Bei den Tastern ist ein RC-Glied zum Entprellen vorgesehen; dieses braucht normalerweise nicht bestückt zu werden; Der notwendige Pullup wird mit ATmega programmiert und die Entprellung erfolgt per SW.

- **Boosterstufe:**
  Der verwendete H-Brückentreiber **L6206** könnte einen theoretischen Maximalstrom von 2,8A je Ausgang. Allerdings ist dies beschränkt durch die maximal mögliche Verlustleistung, welche abgeführt werden muß. Auf dem Layout sind Kühlflächen, damit kann (bei eingelötetem L6206) etwa ein Wärmeübergangswiderstand von 44K/W erreicht werden. Läßt man eine Erwärmung von 90 Grad zu, dann darf nicht mehr als 1,5A Strom entnommen werden.
  Die Wärmeableitung erfolgt im wesentlichen über die Groundpins, deshalb sollte in die Lochreihe quer zum L6206 ein entsprechendes Kühlblech (Kupferstreifen) eingelötet werden. Ersatzweise sind auch Lötstifte und ein kleiner Platinenstreifen möglich. STM nimmt für eine Kühlfläche mit 6cm^{2}Kupfer einen Wärmeübergang von 44 K/W an. Zusätzlich sollte man auf dem Baustein noch einen IC-Kühlkörper aufkleben, um die Wärmeabgabe weiter zu verbessern.
  Andere Gehäusearten (PowerSO36) hätten zwar bessere Wärmeableitung und damit mehr Leistung, aber sind auch teilweise schwierig manuell zu verlöten.
  Die interne Überstromabschaltung ist mit 15k Referenzwiderstand beschaltet, das aktiviert die Abschaltung bei ca. 1.5A. Bei kapazitiven Lasten (z.B. sehr lange Leitungen (⟩20m)) kann es sein, dass die Überstromschaltung bereits wegen des Umladens der Leitungskapazität anspricht. Hier dann unbedingt einen externen Booster nachschalten.

  Bei der Gehäusewahl ist auf eine ausreichende Luftführung zu achten.

Bitte beachten: *der Gleisausgang ist* ***nicht massebezogen****, sondern schaltet zwischen etwas über GND und etwas unter VCC hin und her. Beim Einsatz von Rückmeldemodulen auf potentielle Masseschleifen achten!*

- **S88:**
  Der S88-Bus ist mit Bustreibern 74AC244 ausgerüstet. Jeder S88-Strang hat einen eigenen Clocktreiber. Zur Vermeidung scharfer Flanken ist im Clockausgang ein RC-Glied vorgesehen, am Leitungsende sollte ein Pullup von etwa 330 Ohm gegen 5V vorgesehen werden.
  Es sind die übliche Steckleiste oder alternativ eine RJ45-Buchse bestückbar. Damit kann der S88-Bus mittels CAT5-Kabel verdrahtet werden. AB V1.4 ist die Belegung nach dem Standard S88-N (siehe Kapitel 7.2 auf Seite 166) verwendet.
  S88-N erlaubt auch die Übertragung von RAILDATA, hierzu wurde ein Buffer '125 hinzugefügt; um auch ein eventuelles Programmier-ACK auf S88-N zurück melden zu können wurde der RESET tristate schaltbar gemacht. Diese Optionen sind für den normalen S88-Betrieb nicht erforderlich.

  Hinweise:
  - in der V1.2 bzw. V1.3 (siehe Kapitel 7.1 auf Seite 161) war diese Kabelbelegung verwendet.
  - Adapter lassen sich relativ einfach mit MEB 8-8 (reichelt.de) erstellen. das ist eine RJ45-Buchse mit 70mm Kabelanschlüssen

- **Anschlußstecker (Typen):**
  Anschluß des Gleisausgangs mittels Wago 734-164, zugehöriger Stecker 734-104; (Bezug z.B. www.reichelt.de). Ersatzweise kann auch eine 4-polige Schraubklemme (RM 5,08) bestückt werden.

  Stecker für Power: üblicher DC-Kleinspannungsstecker, z.B. Bürklin 40F140, (0338-8230-00), Typ AMP 5202550 (-,74), Dazu passender Stecker ist 40F120. Ersatzweise kann auch hier eine Schraubklemme bestückt werden.

  Stecker für S88: RJ45: AMP 520251-4, Bürklin 73F558

4.1.3 Wunschliste an Verbesserungen

Natürlich kann eine einfache Box nicht alle Wünsche abdecken und die jeweiligen Vorlieben und Bedürfnisse sind verschieden. Folgende Wünsche an eine Weiter-

entwicklung sind bisher aufgetaucht:
- Support Loconet
- Ethernet, als SRCP Server

4.1.4 Unterlagen (Downloads)

Schaltplan:	V1.4	**opendcc_sch14.pdf**
	V1.3	opendcc_sch13.pdf
	V1.2	opendcc_sch12.pdf
Stückliste:		
Layout:	V1.4	**opendcc_brd14.pdf**
	V1.3	opendcc_brd13.pdf
	V1.2	opendcc_brd12.pdf
Datenblätter:	L6202	datenblatt_L6206.pdf

4.1.5 Hardware Change Log

from V1.3 to V1.4

- RJ45 according to S88-N.
- Buffer for raildata added.
- Reset tristate possible
- switch between USB and RS232 through jumper
- SJ4 on comp. side

from V1.2 to V1.3

- Rotated RJ45 connectors
- More holes on front panel keyboard
- Added C44 (10nF) at IC6
- Changed value of R19, R20 from 10k to 2k2

Den aktuellen Schaltplan gibt es im Kapitel 4.5 auf Seite 140 zu sehen.

4.2 Bauanleitung

4.2.1 Vorbemerkungen

Modellbahn ist VDE-technisch Spielzeug und da gelten strenge Schutzvorschriften. Siehe hierzu auch die Sicherheitshinweise des Fremo (Freundeskreis Europäischer Modellbahner eV.)[7] für die Modellbahn.

Norbert Martsch[8] bietet einen Bausatz an, er hat auch eine Bestellliste[9]bei reichelt.de hinterlegt.

4.2.2 Löthinweise

Zuerst mal der übliche Sums: Elektroniklötkolben (nicht über 300W;-), geeignetes Lötzinn (kein Tiffany-Bleilot), Dioden und Kondensatoren richtig polen (kleiner Strich oder + im Layout), ICs sind statisch gefährdet, kurzum: wer's nicht kann, soll nicht mit dieser Schaltung das Üben anfangen.

Andererseits gibt es bis auf den USB-Chip auch keine nennenswerten Schwierigkeiten. Die Werte der Widerstände und C's sind in der Regel unkritisch, im Schaltplan finden sich Hinweise zur Dimensionierung. (Ausnahmen: R37 muß 1% sein: 348 Ohm, R32 und R10 müssen Metallfilm sein). Bei LED's ist der lange Anschluß Anode (+) und der kurze Kathode (-).

Die IC5 (H-Bridge) sollte nicht gesockelt werden, da die Wärmeableitung im wesentlichen durch die Pins auf die Platine erfolgt.

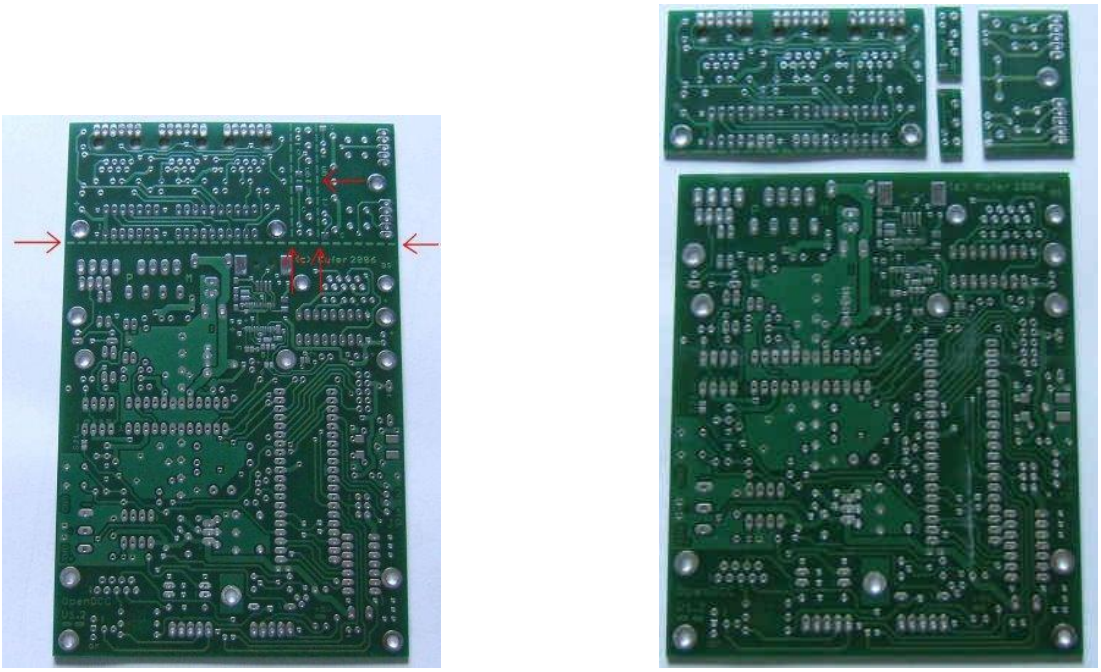
Der USB-Chip FT232RL kommt im 28-poligen Gehäuse mit 0,8mm Pitch. Hier wird zuerst nur ein Eckpin auf der Leiterplatte verzinnt, dann wird der Chip darüber gelegt. Nun kann dieses Pad und Pin erhitzt werden und der Chip exakt auf die Pads ausgerichtet werden. Nun kann man alle anderen Pins verlöten - feine Lötspitze und 0.5mm Lötzinn sind hilfreich ;-). Keine Panik, wenn zwei Pins zusammenkleben! Nicht lang rumbrutzeln, sondern abkühlen lassen und dann einfach mit dünner Lötsauglitze das überschüssige Zinn aufnehmen. Am Ende alle Lötstellen nochmal mit der Lupe kontrollieren.

[7] Freundeskreis Europäischer Modellbahner eV. - http://www.fremo.org/safety/vde_d.htm - 06.01.2008
[8] Norbert Martsch - http://www.norbert-martsch.de/ - 06.01.2008
[9] Norbert Martsch https://secure.reichelt.de/?;ACTION=20;LA=4000;AWKID=14794;PROVID=2084 gefunden und getestet am 06.01.2008

Bei der Montage des Quarzes kann fallweise ein Kurzschluss der SMD-Pads über das Metallgehäuse entstehen. Hier entweder einen Klebestreifen unterlegen oder mit etwas Abstand montieren.

4.2.3 Vorbereitung

Vor Beginn der Bestückung muss die Europlatine entlang der strichlierten Linien in fünf Teilplatinen zerlegt werden:



(a) mit Trennlinien (b) geteilte Platine

Abb. 20: Europlatine OPENDCC (Quelle: archiv)

Grundplatine: 120mm * 100mm
S88-Platine: 70*38 mm
Tastaturplatine: 38 * 23mm
2 LED-Platinchen: 17 * 5mm

Die Tastaturplatine bildet zusammen mit den 2 LED-Platinchen die Frontplatine, diese ist wie folgt angeordnet:

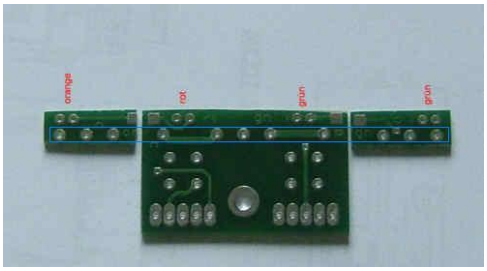
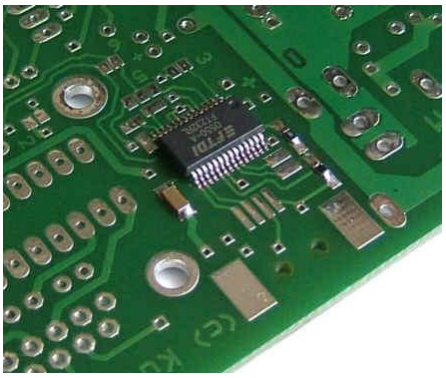


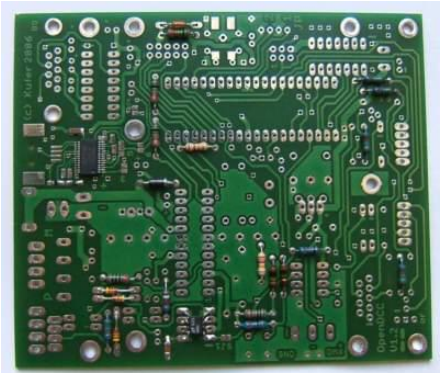
Abb. 21: Tastaturplatine (Quelle: archiv)

Nun werden mit kleinen Drahtstücken die Lötpad entsprechend der Buchstaben A-D miteinander verbunden, wobei darauf zu achten ist, dass das Raster einer möglichen Versteifungsschiene (blaue Linie) durchgehend ist. Dann ergeben sich identische Abstände für die LEDs. Diese Frontplatine wird später senkrecht im Gerät montiert (A-B = hinten, C-D = vorne).

4.2.4 Grundplatine



(a) SMD-Bauteile



(b) Rückseitenbauelemente

Abb. 22: Bestückung der Platine OPENDCC (Quelle: archiv)

Es sollten zuerst die SMD-Bauteile für den USB-Anschluß bestückt werden. Dann werden die Widerstände, Dioden, und Fassungen bestückt.

Anschließend werden IC's, Stecker und Kondensatoren bestückt. Die Rückwandstecker sollten zuvor an der Rückwand ausgerichtet werden.

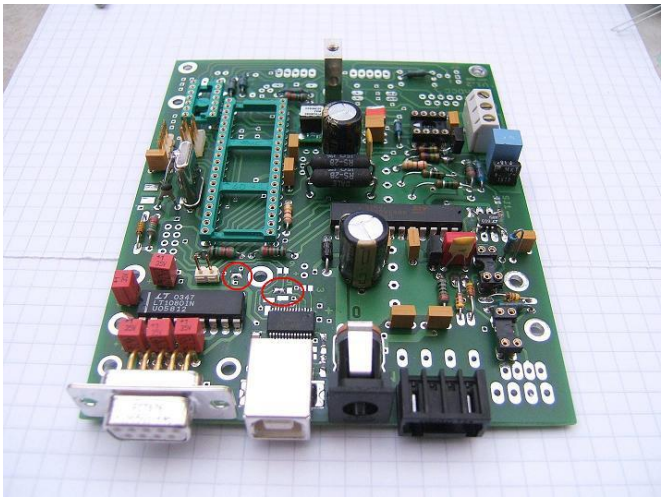


Abb. 23: Komplett bestückte Platine - Ansicht von hinten (Quelle: archiv)

Komplett bestückte Platine, Ansicht von hinten. Hinweis: hier sind beide PC-Schnittstellen bestückt, normalerweise ist nur eine Schnittstelle bestückt. Die jeweilige Schnittstelle wird per Jumper JP5 und JP6 ausgewählt. Diese Jumper haben drei Positionen und sind auf der Platine beschriftet. (Die beiden roten Markierung zeigen die Stellen der Lötbrücken bei der V1.3. Hier dargestellt: Betrieb über RS232).

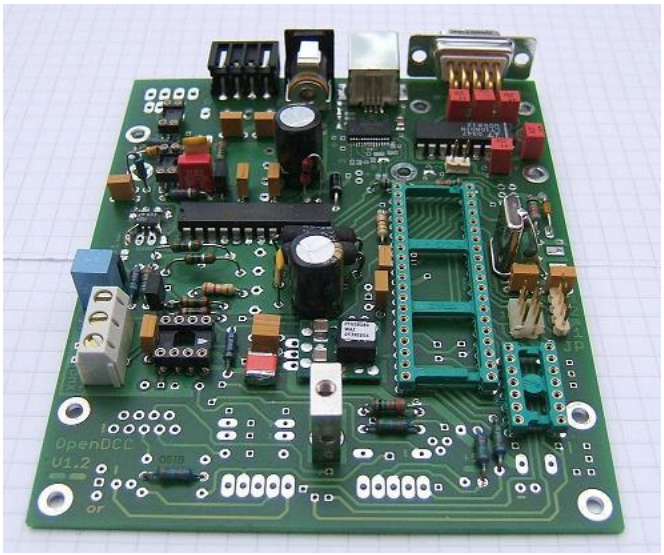


Abb. 24: Komplett bestückte Platine - Ansicht von vorne (Quelle: archiv)

Komplett bestückte Platine, Ansicht von vorne (die fehlenden Bauteile sind nur

für ponyprog-Betrieb erforderlich).

4.2.5 Frontplatine (Tasten, LED)

Zuerst wird die Frontplatine vorne aus ihren Teilplatinen zusammengelötet. An die Tastenplatine werden links und rechts die kleinen LED-Platinen angelötet, und zwar so, dass die entsprechenden Buchstaben A-D miteinander verbunden werden und das Raster der Versteifungsschiene (blauer Rahmen) durchgehend ist. Damit liegen alle LEDs auch im Raster von 22.86mm. Am besten diese Montage auf einem Stück Lochraster durchführen oder z.B. die Versteifung mit einer durchgehenden Stiftleiste (siehe Abbildung 25) machen.

Diese Frontplatine wird vorne mittig senkrecht montiert. Hierzu werden 5-polige Winkelstifte (oder auch Drahtreste von den Widerständen) verwendet; diese werden nur eingesteckt, noch nicht verlötet.

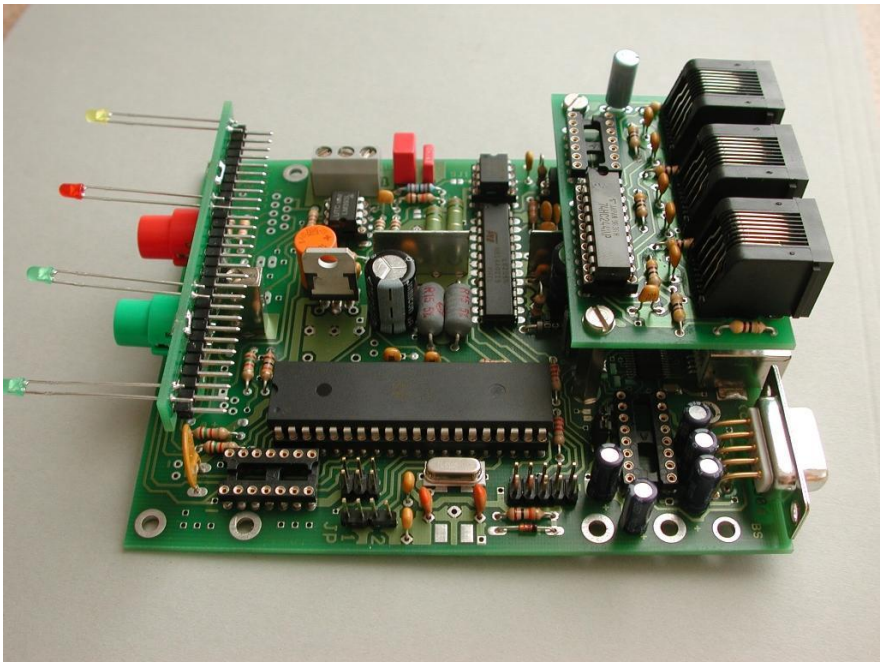
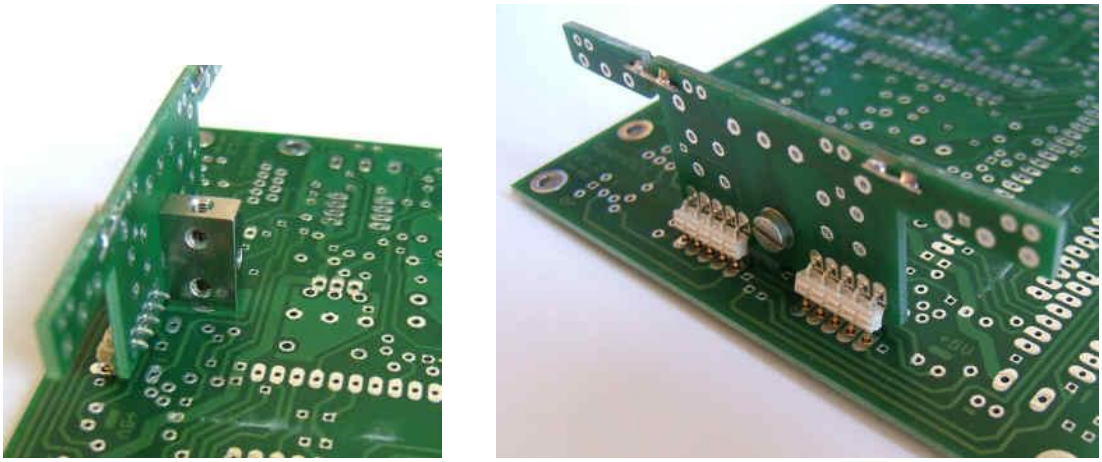


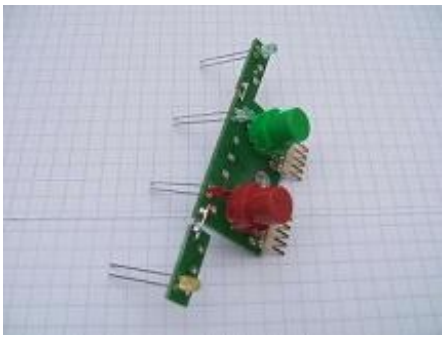
Abb. 25: Versteifung mit einer durchgehenden Stiftleiste (Quelle: archiv)



(a) Frontplattenplatine (b) Winkelstifte verlötet

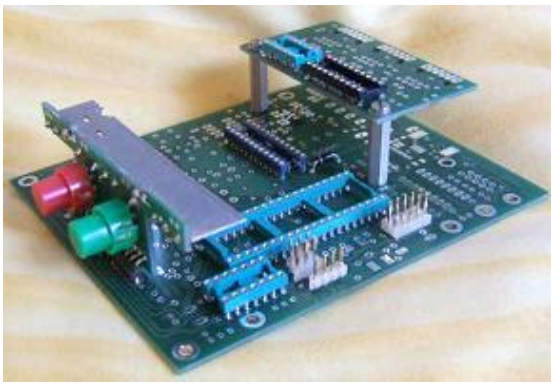
Abb. 26: Monage der Tastaturplatine (Quelle: archiv)

Auf der Frontplatte werden zuerst nur die Tasten bestückt, alle anderen Bauteil bleiben noch lose. Die Frontplatte wird nun mit einem Gewindewürfel (Bürklin, Best.Nr. 17H912) mit der Grundplatte verschraubt. Wenn man sich vergewissert hat, dass die Tasten genau zur Frontplatte ausgerichtet sind, dann können die Winkelstifte verlötet werden.
Tipp von Alex: statt des Gewindewürfels kann nach dem Ausrichten der Platine auch ein dicker Draht schräg durch die beiden Schraublöcher gelötet werden. Und der Würfel ist auch als "GB4xM3" für 0,50 bei http://www.qrpshop.de/ erhältlich. (Stand:12/2007)



(a) Frontplatine, die LEDs stecken noch lose in ihren Bohrungen.

(b) Erst bei einer Probemontage im Gehäuse werden die LED in die Bohrlöcher der Frontplatte geschoben und verlötet. Die Frontplatine könnte gegen Durchbiegen mit einem Blech- oder Platinenstreifen entlang der blauen Linie versteift werden.



(c) So werden die Platinen zusammengesetzt.

(d) So fügt sich der Aufbau ins Gehäuse ein.

Abb. 27: Probemontage im Gehäuse (Quelle: archiv)

4.2.6 Komplette Platine

(a) Ansicht von vorn

(b) Ansicht von rechts

(c) Ansicht von links

(d) Ansicht von hinten

Abb. 28: Endmontage im Gehäuse (Quelle: archiv)

4.2.7 Rückwand

An der Rückwand sind RS232 zum PC, Powereingang, USB, DCC Ausgang und Ctrl In-Out direkt auf der Platine bestückt. Auch hier sollte eine gebohrte Rückwand als Lehre zum genauen Ausrichten der Bauteile verwendet werden. Die S88 Schnittstellenplatine wird als Huckepackplatine mit Schraubbolzen M3*20mm montiert. Die Verbindung zum Grundboard erfolgt mittels 14-poligen DIL-Verbindern in Schneidklemmtechnik.

Achtung:

Es gibt zwei verschiedene Ausführungen dieser Stecker: Einmal ist der Pin 1 identisch zur Kabelader 1, bei der anderen Variante ist der Pin 14 auf die Ader

1 gelegt. Es ist egal, welcher Stecker verwendet wird, es muß nur beide mal der gleiche Typ sein. Sollte ein 14-poliger Stecker nicht verfügbar sein, so kann auch eun 16-poliger Stecker verwendet werden, es sind entsprechende Leerlöcher in der Platine vorgesehen.
Tipp zum Verpressen dieser Stecker: auf die Anschlußpins einen verlöteten Stapel von 3 Lochrasterplatinen legen und dann im Schraubstock verpressen.

4.2.8 Gehäuse

Die Platine samt montierten Huckepackplatinen kann in das Fischergehäuse eingeschoben werden bzw. beim Tekogehäuse auf die vorhandenen Schraubsockel montiert werden. Die Front- und Rückplatten werden gemaß Skizzen (siehe Kapitel 4.4 auf Seite 136) gebohrt und montiert.
Sollte die Platine auch als DMX-Schnittstelle verwendet werden, so ist der XLR-Stecker mittels Kabel an die dreipolige Schraubklemme anzuschliessen. Die Anschlußnummmer auf der Platine ist identisch zur Pinnummer der XLR-Buchse.
Hinweis: bei geplanter Lichtsteuerung für die Modellbahn empfehle ich eher den Decoder. evtl. entfällt in zukünftigen Softwareversion der DMX-Support.

4.2.9 Test, erstes Einschalten

Nach dem Bestücken alles nochmal genau optisch kontrollieren und Flußmittelreste mit Alkohol abwaschen. Der Prozessor wird einstweilen noch nicht bestückt.

Nun wird zum ersten Mal Spannung angelegt (aus einem geregelten Labornetzteil, Strombegrenzung auf 50mA).

Die 5V kontrollieren, dann kann ein erster Test der LEDs erfolgen: Hierzu einen Widerstand 220 Ohm mit Masse verbinden und mit dem anderen Ende nacheinander die Pins 28, 29, 35, 36 des Sockels für den Atmel auf GND ziehen. Die LED müssen leuchten. Nun werden die weiteren Bausteine bestückt.
Wenn alles bestückt ist, dann kann nochmal Spannung angelegt werden; die Ruhestromaufnahme bei nicht programmierten Prozessor sollte etwa 30mA sein, bei programmierten Prozessor etwa 50mA; Die Ruhestromaufnahme ist höher, wenn ein Längsregler für die 5V verwendet wird und wenn der DCC-Ausgang mit RC-Tiefpässen beschaltet ist (+60mA).

4.2.10 Prozessor erstmals programmieren

Nachdem die Hardware fertig aufgebaut ist, muß der Prozessor noch passend konfiguriert und mit der Firmware geladen werden. Dies erfolgt in drei Schritten:

1. Fuses setzen
2. Bootloader einspielen
3. Firmware-Update durchführen (siehe Kapitel 3.7 auf Seite 103)

Als erstes müssen die Fuses gesetzt werden.
Der Atmel[11] ATmega32 verfügt über mehrere Möglichkeiten der Taktung und verschiedene Bootmodi. Diese werden mit sog. Fuses gesetzt. Für den in OpenDCC eingesetzten Prozessor sind folgend Fusebits zu setzen:

		Fusebits
CKSEL3...0	= 1111	keine Häkchen, bedeutet Crystal mit 16MHz
CKOPT	= 0	Häkchen gesetzt, bedeutet voller Hub an XTAL2
SUT1..0	= 0 1	Häkchen bei SUT1, kein Häkchen bei SUT0 (Power Up Delay)
BOOTSZ1..0	= 0 1	Häkchen bei BOOTSZ1, kein Häkchen bei BOOTSZ0; Start bei 0x7800
BOOTRST	= 0	Häkchen gesetzt, bedeutet, das der Bootloader verwendet werden soll →siehe FW-Update (siehe Kapitel 3.7 auf Seite 103)
BODLEVEL	= 0	Häkchen gesetzt, bedeutet, dass der Brown-Out Detector bei 4V anspricht.
BODEN	= 0	Häkchen gesetzt, bedeutet, dass der Brown-Out eingeschaltet ist.

Es gibt drei Wege, um den ATmega[11] zum ersten Mal zu programmieren: JTAG, ISP (6-polig) und Ponyprog[3]. Hier wird nur der Bootloader eingespielt, das eigentliche Programm wird wie auch später ein eventueller Programmupdate über die normale Schnittstelle (USB oder RS232) eingespielt.

Wenn man über Ponyprog[3] oder ISP programmiert, dann muß der Jumper JP4 gezogen sein, weil sonst der RS232-Baustein und der Programmieradapter gegeneinander treiben. Nachfolgend ist die (Erst-)Programmierung mit Ponyprog[3] beschreiben.

Ein geeigneter Adapter für die ISP-Schnittstelle kann leicht selbst gebaut werden:

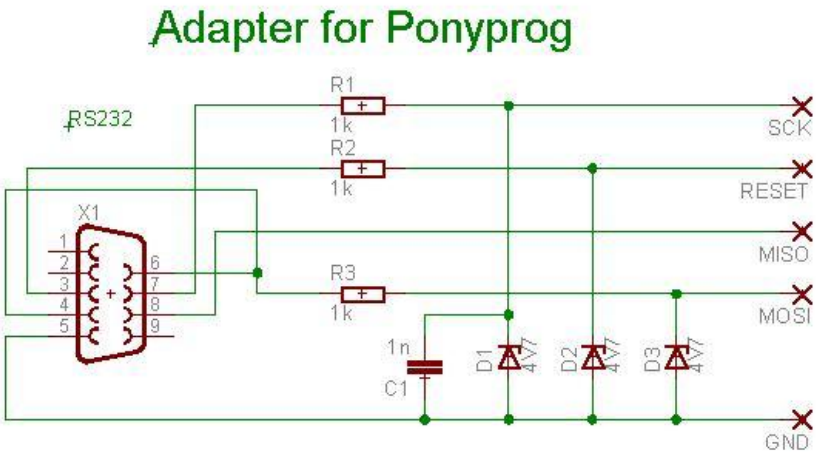


Abb. 29: Schaltung eines Prorammieradapters
(Quelle: Autor; Schaltung nach Ponyprog[20])

Hinweis: Bei diesem Adapter ist der Reset in der Ponyprog[3]-Configuration als invertiert zu markieren.
Dieser Adapter wird an der 6-poligen ISP-Schnittstelle angeschlossen und dann wird der Prozessor programmiert; anschließend kann der Adapter wieder abgezogen werden. Der Adapter kann auch andere Prozessoren programmieren.

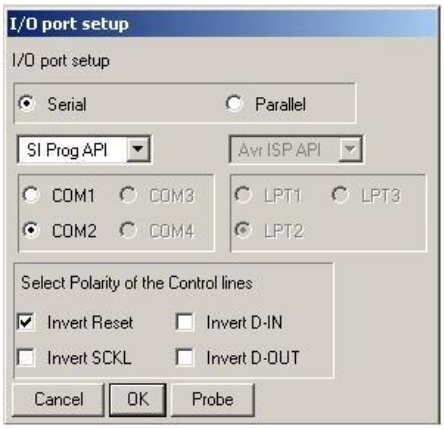


Abb. 30: IO-Port Setup(Quelle: Screenshot Ponyprog[3])

Zuerst muß die Schaltung der ISP-Pins Ponyprog[3] bekannt gemacht werden. Für OpenDCC[21] ist in den Einstellungen des Adapters der Reset als invertiert

zu markieren.

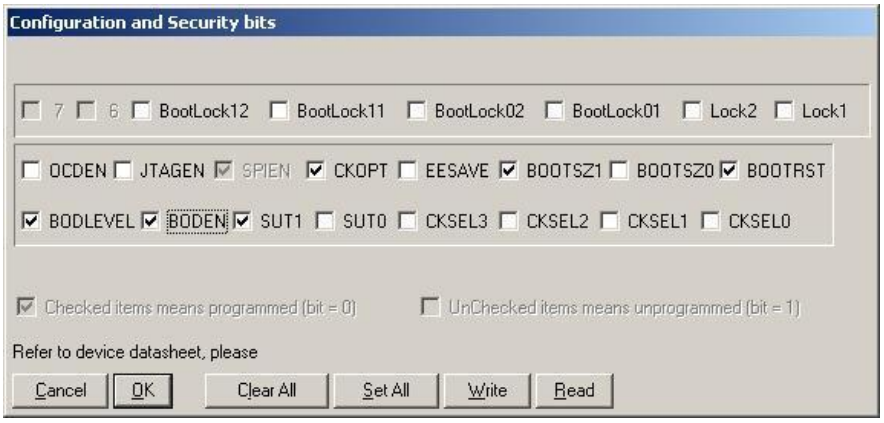


Abb. 31: Fusebits (bei aktiviertem Bootloader)(Quelle: Screenshot Ponyprog[3])

So werden die Fusebits (bei aktiviertem Bootloader) geschrieben.

Zum Laden des Bootloader wird das bootloader.hex-File geöffnet und dann mit *write program* an den Atmel[11] übertragen.

Hinweis: Wenn kein Bootloader gewünscht ist, dann muss der Haken bei BOOTRST entfallen. Das Programm und EEPROM kann dann direkt mit Ponyprog[3] eingeladen werden. (der Update funktioniert dann aber auch nicht ...)

Fuses für den Atmega644P (für Xpressnet)

								Wert/ Bedeutung
Lock Bit Byte								
7	6	5	4	3	2	1	0	
1	1	1	1	1	1	1	1	0xFF not locked
Extended Fuse Byte								
7	6	5	4	3	2	1	0	
-	-	-	-	-	BOD2	BOD1	BOD0	
1	1	1	1	1	1	0	0	0xFC 100 = 4.3V
Fuse High Byte								
7	6	5	4	3	2	1	0	
OCDEN	JTAGEN	SPIEN	WDTON	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST	
1	0	0	1	1	1	0	0	0x9C enter boot @0x7C00 (words)
Fuse Low Byte								
7	6	5	4	3	2	1	0	
CKDIV8	CKOUT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0	
1	1	0	1	0	1	1	1	0xD7 no div, crystal, BOD enabled

Der Bootloader ist 2048 Bytes bzw. 1024 Words groß.

4.2.11 USB programmieren

Zum Betrieb des USB-Chips muss dieser zuerst konfiguriert werden (siehe Kapitel 3.10 auf Seite 109) . (Dieser Schritt ist natürlich nur erforderlich, wenn die USB-Schnittstelle verwendet wird)

4.2.12 OpenDCC laden

Wenn jetzt soweit alles fertig ist, dann kann die Applikation geladen werden. Hierzu wird genauso wie beim normalen Einspielen der Software verfahren:

1. Während des Startens die STOP-Taste gedrückt.
2. Die gelbe LED leuchtet auf und OpenDCC wartet auf den Download des Programms (siehe Kapitel (3.7 auf Seite 103) und des EEPROM Speicherinhalts.
3. *Hinweis: Bei USB-Betrieb muss aber erst noch der USB-Chip konfiguriert werden, sonst gibt es die Verbindung PC-OpenDCC für den Update noch nicht.*

4.2.13 OpenDCC konfigurieren

Als letzter Schritt können (müssen aber nicht) jetzt noch persönliche Einstellungen vorgenommen werden. Diese sind in internen Konfigurationsvarablen (siehe Kapitel 3.3.1 auf Seite 80 abgelegt.
Es gibt mehrere Möglichkeiten, diese zu lesen und zu schreiben. Besonders komfortabel geht es mit Trainprogrammer[8] (siehe Kapitel 3.4 auf Seite 90.

4.2.14 Achtung, Fehlerhinweise zur Hardware Version 1.2

1. In der Platinenversion V1.2 sind die RJ45-Buchsen zum S88-Anschluß mit Ethernetkabeln für stehende Buchsen verdrahtet. Liegende Buchsen (die dann auch durch die Rückwand von außen steckbar sind) haben ein um 180 Grad gedrehtes Anschlußschema :-(Sollen diese Buchsen bestückt werden, so sind *alle* Anschlüsse zu den RJ45-Buchsen vor dem Bestücken mit einem Fräser durchzutrennen und mit Wrap- oder Fädeldraht um 180 Grad verdreht wieder herzustellen. Bitte auch darauf achten, dass Masse und VCC über der gesamten Adapterplatine verbunden bleibt. Leider ist das eine üble Bastelei ... (Diese Änderung ist nicht notwendig bei S88-Anschluß an den Pfostensteckern)

2. Wertänderungen:
   S88-Clockausgang: alle Widerstände 33 Ohm, C25, C28, C29 1nF →10nF Kondensatoren beim Quarz: C4, C5: 33pF →18pF zusätzlicher Kondensator (C44) 10nF zwischen IC6 (LM393) Pin 2 und Pin 3 R19 und R20 ändert sich von 10k auf 2k2. Grund: schnellere Erholung der Ausgangsstufe bei Abschalten nach Polaritätwechsel durch die Kapazität der Anlagenverdrahtung (war aufgetreten ab etwa 40m verdrillte Leitung)

Die obigen Änderungen sind in der Version 1.3 bereits eingearbeitet.

4.2.15 Was tun, wenn es nicht funktioniert?

Zuerst mal Ruhe bewahren :-)

Dann folgende Dinge kontrollieren:

1. Kontrolle der Bauelemente, ob überall das richtige Bauteil mit der richtigen Polung steckt.
2. Kontrolle aller Steckbrücken (JP...) und Lötbrücken (SJ...).
3. Sichtprüfung der Lötstellen und der Leiterbahnen - am besten mit der Optikerlupe.
4. Kontrolle der 5V und der Ruhestromaufnahme
5. Kontrolle ob die Signatur des Atmel gelesen werden kann: Atmega644P: 1E 96 0A
6. Kontrolle ob das Programmieren geht - und falls ja, mit Ponyprog[3] wieder zurücklesen.

7. Kontrolle von Reset und des Taktoszillators; an XTAL2 müssen 16MHz anliegen.
8. Wenn nach dem Programmieren des Bootloaders der Firmware-Update nicht funktioniert, dann die Firmware mal direkt laden und den Testmode aktivieren. Sind .hex und .eep geladen?
9. Stimmt die Position von Jumper 4? (für Programmierung mit Ponyprog[3] entfernen, für den Betrieb stecken)
10. Tipp von Jan: Bei einem ersten Test lief der Prozessor mit Quarz nicht an, als Fehler stellte sich hier ein Kurzschluss der SMD-Pads über das Metallgehäuse des normalen Quarzes heraus. Bitte einen Klebestreifen unterlegen.
11. Einschalten Testmode: Hierzu beide Jumper (JP1 und JP2) einstecken, es muß nun ein Lauflicht der 4 LEDs leuchten; Im Testmode sendet OpenDCC das empfangene Byte (mit 19200Baud) einfach wieder zurück - dies kann man mit einem Terminalprogramm (wie z.B. realterm) überprüfen. Dieser Testmode funktioniert auch bei USB!
12. Kontrolle, ob der RS232-Baustein auch die +/- 8-10V erzeugt. Ich hatte schon mal den Fall, dass ein Kondensator ab Tüte defekt war.
13. Kontrolle, ob das verwendete serielle Kabel auch die Handshake-Leitungen enthält.

Falls Sie beim Nachbau einen Fehler gemacht haben und diesen selbst gefunden haben, bitte trotzdem mitteilen, damit evtl. andere diesen Fehler vermeiden können.

4.3 Stückliste

4.3.1 Version 1.2, 1.3 und 1.4

Reichelt-Stückliste Stand: 07/2008
https://secure.reichelt.de/?;ACTION=20;LA=4000;AWKID=14794;PROVID=2084

Stückliste - Xpressnet-Extension

Bauteil	Wert	Device	Package	Option	Order-Code
C1	220uF/25V	CPOL-EUE2.5-7	E2,5-7		
C2	100uF/16V	CPOL-EUE2.5-7	E2,5-7		
C3	100nF	C-EU050-024X044	C050-024X044		
C4	100nF	C-EUC0805K	C0805K	not fitted	
C5	100nF	C-EUC0805K	C0805K	not fitted	
C6	100nF	C-EU050-025X075	C050-025X075		
C7	100nF	C-EU050-024X044	C050-024X044		
C8	100nF	C-EU050-024X044	C050-024X044		
C17	100nF	C-EU050-024X044	C050-024X044		
D3	1N4004	DIODE	DO41-10		
F1	1A	FUSE	TE5		
IC3	MAX485	alt.: MAX481CPA	DIL08		MAX 485 CPA
IC4	7812	REGULATOR	TO220		µA 7812
J1	RJ12-Plug	AMP 555077-1	SMD, Snap-In	AMP/TYCO	
JP1	2PIN PLUG	JP1EE	JP1E		BL 1X20G8 2,54
JP2	2PIN PLUG	JP1EE	JP1E	(see JP1)	
JP3	6PIN PLUG	JP3Q	JP3Q	(see JP1)	
LED7	green 2mA	LED3MM	LED3MM		
R1	120	R-EU_0207/10	0207/10		
R2	1k5	R-EU_0207/10	0207/10		
R3	4k7	R-EU_0207/10	0207/10		
R4	4k7	R-EU_0207/10	0207/10		
X12	JTAG	JP5Q	JP5Q		BL 1X20G8 2,54
IC5	ATMEGA644P-20PU	MICROCONTR.	DIL40	für Xpressnet	

4.4 Mechanik, Bohrmaße, Frontplatte

4.4.1 Details zur Mechanik

- **Maße:** Grundplatine 100 * 120mm, S88-Aufsteckplatine: 38 * 75mm, Tastatur: 38 * 23mm.
  (Alle Platinen befinden sich auf einer Europaplatine 160*100mm und müssen vor dem Bestücken entlang der gestrichelten Linien getrennt werden.)
- **Befestigungslöcher der Platine:**
  Grundplatine (Ansicht von vorne oben, (x,y[mm])):
  je ein Loch bei: A(4.5, 4.5), B(95.5, 4.5), C(4.5, 85), D(66, 85) und E(95.5, 85);
  Aufnahmen für das Teko-Gehäuse: T1(4.5, 22), T2(95.5, 22), T1(4.5, 98), T1(95.5, 98)

  S88-Aufsteckplatine: je ein Loch bei C'(7, 3) und D'(63.5, 3);
  Die Aufsteckplatine wird mit zwei Abstandssäulen 15mm zwischen C und C' bzw. D und D' auf die Grundplatine montiert. Die Achsmaße der der RJ45-Buchsen bzw. der Steckleisten für S88 liegen bei 15mm, 35mm und 55mm; Mit dem Versatz von 2.5mm durch das Aufschrauben ergeben sich dann: 17,5mm, 37,5mm und 57.5mm.

  Tastenplatine: mittig zwischen den Tastern, Abstand des Loches vom (gedachten) unteren Rand 7,5mm; damit kann die Tastenplatine mit Hilfe eines

Montagewürfels (Bürklin[25], Best.Nr. 17H912) auf der Grundplatine befestigt werden. Der elektrische Anschluß erfolgt mittels zweier abgewinkelter Stiftleisten 5-polig Typ Fischer SL (Bürklin[25], Best.Nr. 59F8228).
Die Tasten befinden sich 13,34mm (=7,5mm + 5,84mm (230mil)) über der Grundplatine (Achsmitte); Die LED sind weitere 400mil darüber: 23,5mm (=7,5+16(630mil)) über der Grundplatine, der Achsabstand der Tasten zueinander beträgt 22,86mm (900 mil); Die Tasten sind symmetrisch zur Grundplatine angeordnet.
An die Tastenplatine werden links und rechts kleine LED-Platinen angelötet, und zwar so, dass die entsprechenden Buchstaben A-D miteinander verbunden werden und das Raster der Versteifungsschiene durchgehend ist. Damit liegen die beiden weiteren LED liegen auch im Raster von 22.86mm. Es ergeben sich somit folgende Achsmaße für die LEDs: 15,71mm, 38,57mm, 61,43mm, 84,29mm.

- **Gehäuse:**
  Wenn man die Platine mit den direkt eingelöteten Steckverbindern verwenden möchte, dann braucht man ein Gehäuse mit einer Bautiefe 120mm. Geeignet hierfür ist z.B. das Alu-Strangpressgehäuse der Fa. Fischer Typ AKG 105 46 120 SA oder das Gehäuse Teko Cab 022, (Bürklin[25] 60H304 oder Reichelt[12]); der bei Reichelt angegebene Lochabstand von 86.4 mm ist falsch. Richtig ist die Info auf der Teko.it Seite mit 91mm; Bolzenhöhe ist 4mm.

4.4.2 Frontplatte, Rückwand:

- 1. Möglichkeit mit einem Gehäuse der Fa. Fischer.

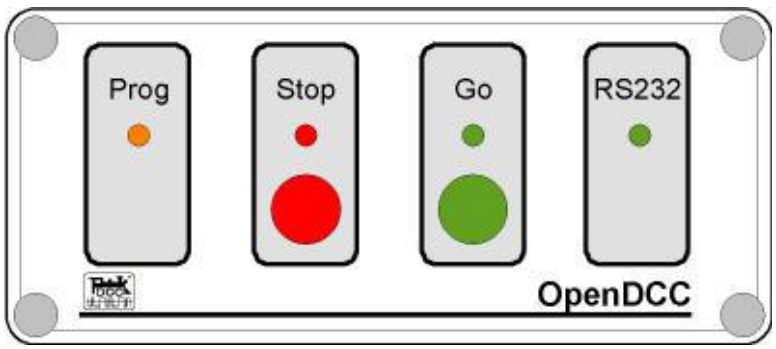


Abb. 32: Frontplatte - für Gehäuse Fa. Fischer (Quelle: Autor)

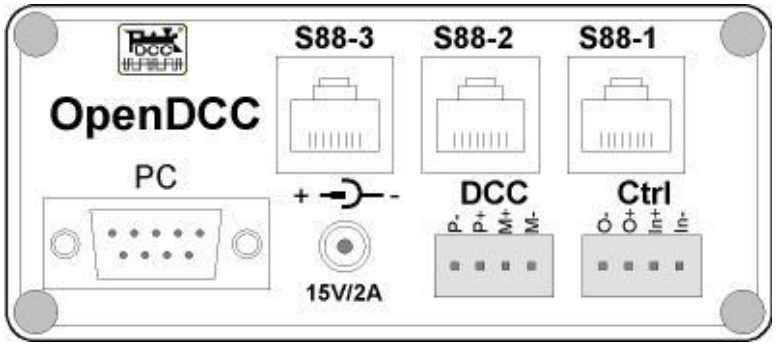


Abb. 33: Rückplatte - für Gehäuse Fa. Fischer (Quelle: Autor)

- 2. Möglichkeit mit einem Gehäuse der Fa. Teko.

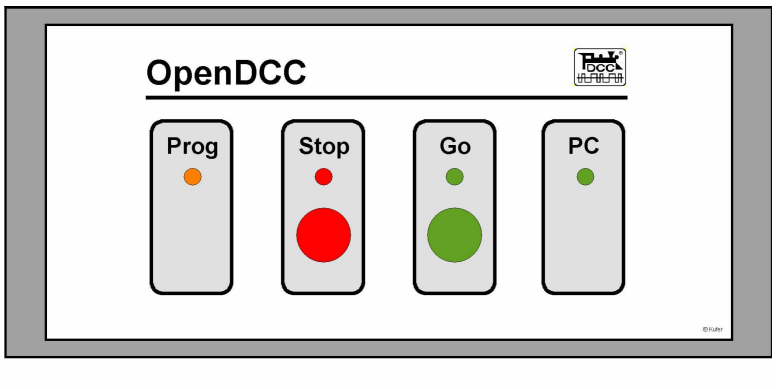


Abb. 34: Frontplatte - für Gehäuse Fa. Teko (Quelle: Autor)

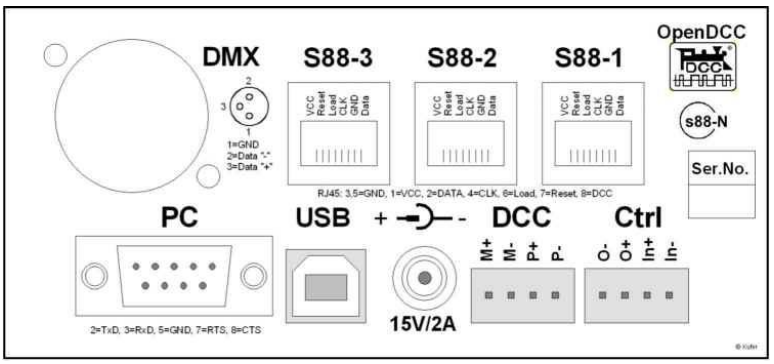


Abb. 35: Rückplatte - für Gehäuse Fa. Teko (Quelle: Autor)

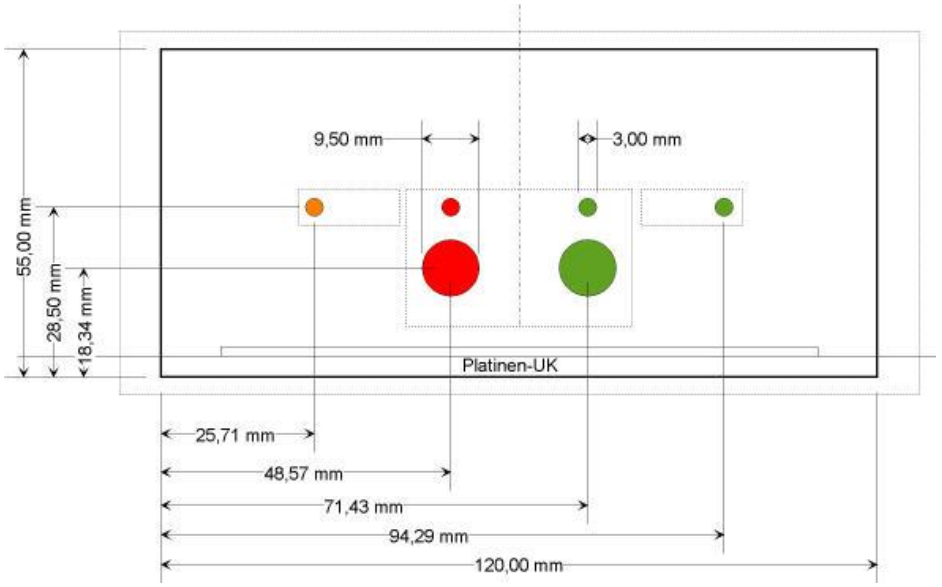


Abb. 36: Maßskizze Frontplatte für Gehäuse Fa. Teko (Quelle: Autor)

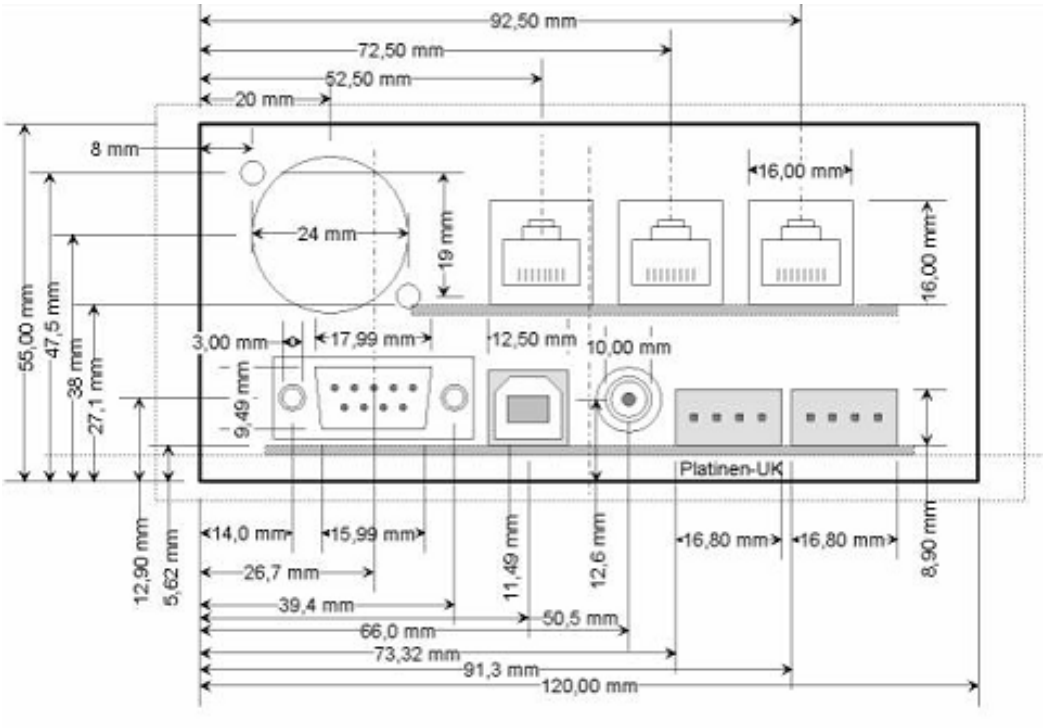


Abb. 37: Maßskizze Rückwand für Gehäuse Fa. Teko (Quelle: Autor)

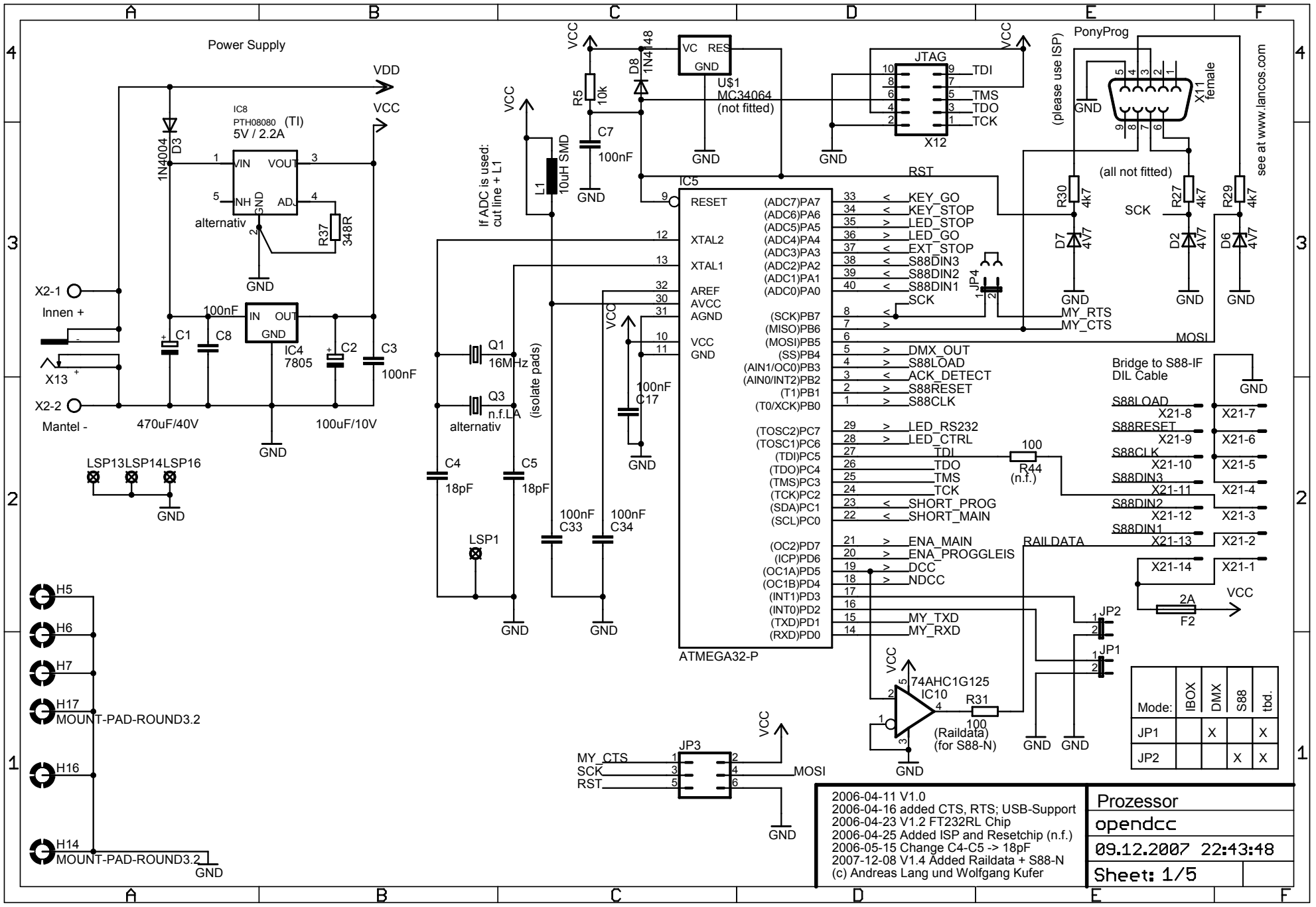
4.4.3 Unterlagen

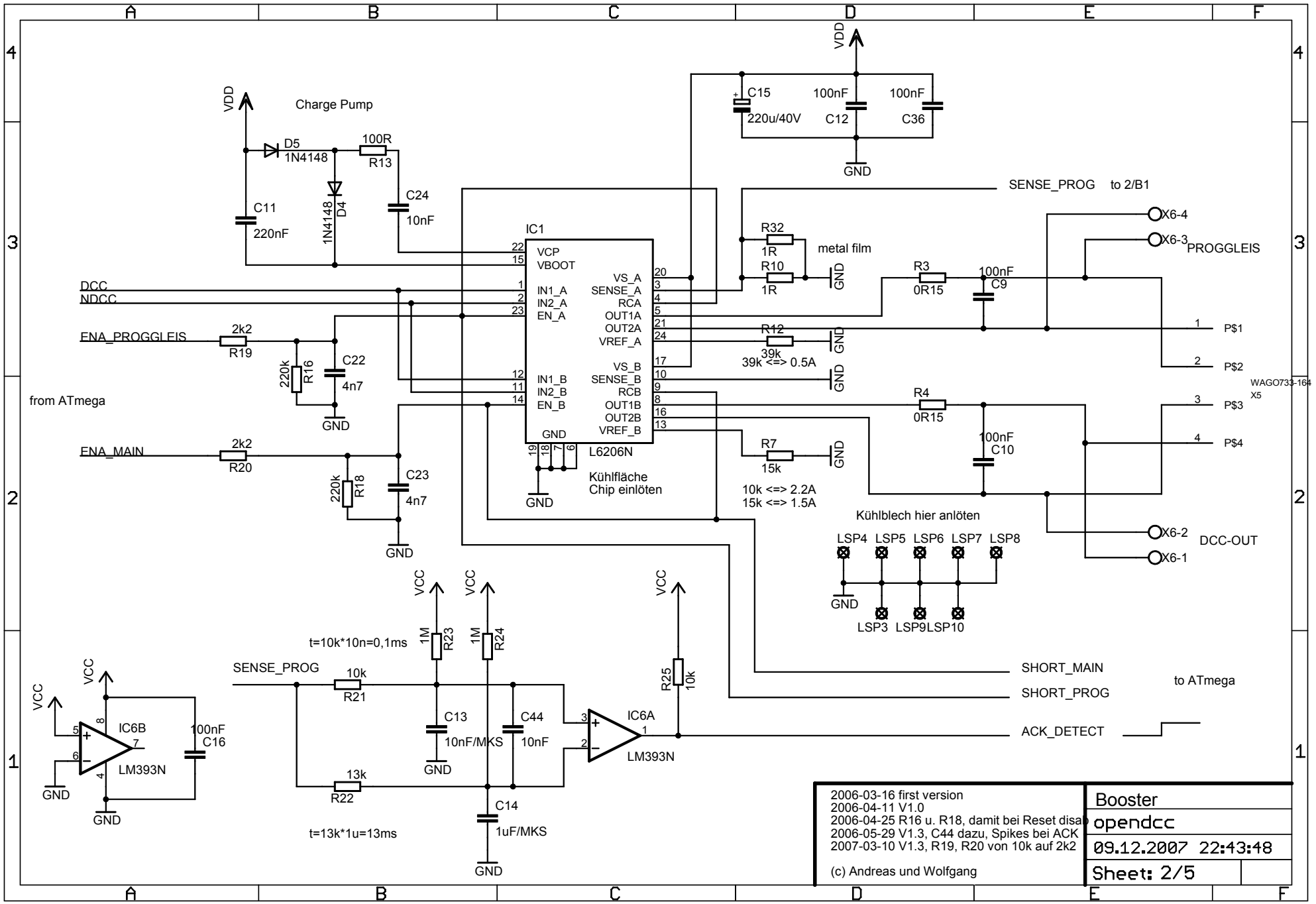
Frontplattenentwurf_Teko_V1.4.dsf
(MicroGraphx Designer) (für V1.4, neue Beschriftung S88-N)

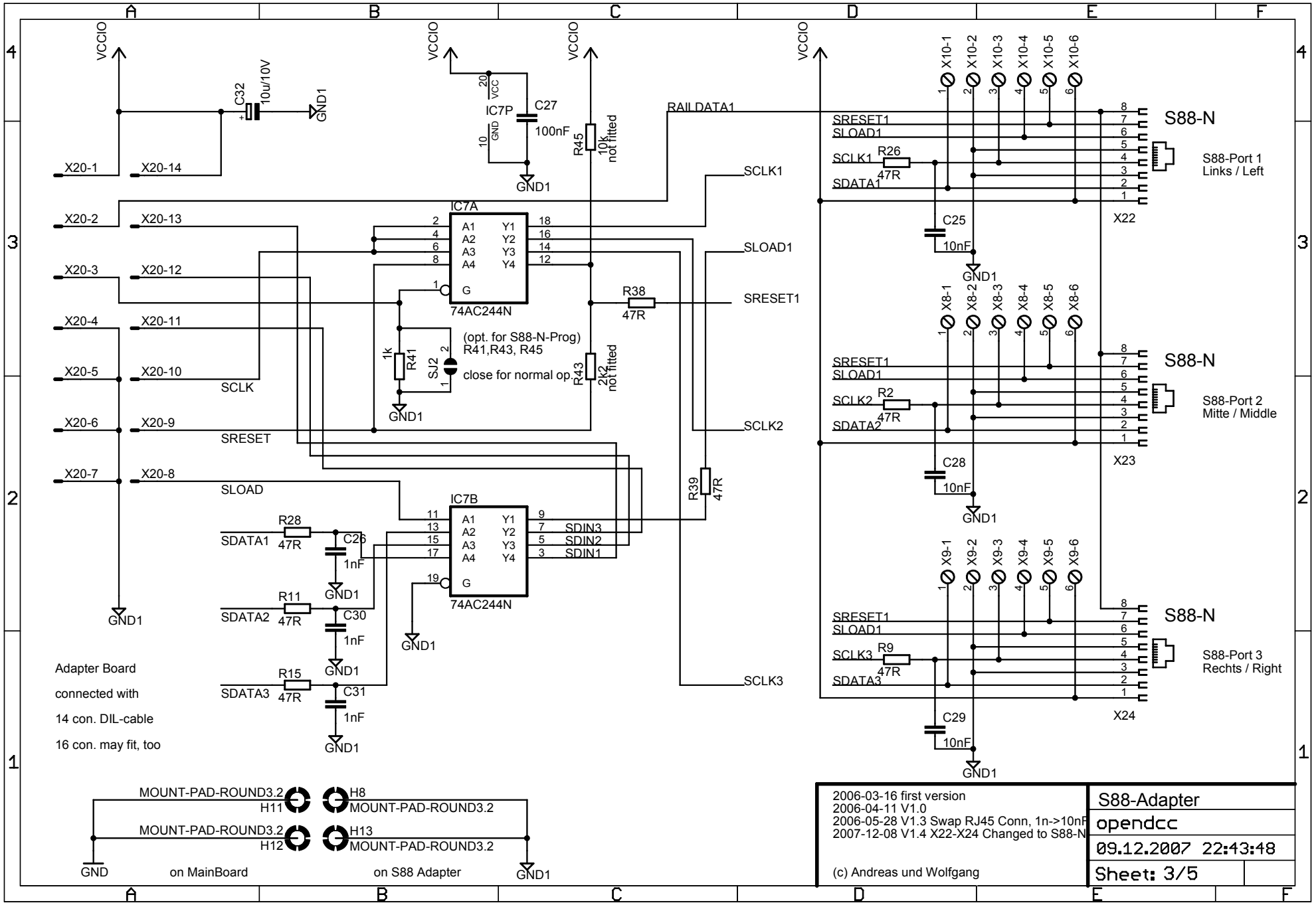
Frontplattenentwurf_Teko_V1.1.dsf
(MicroGraphx Designer) (für V1.1 bis 1.3)

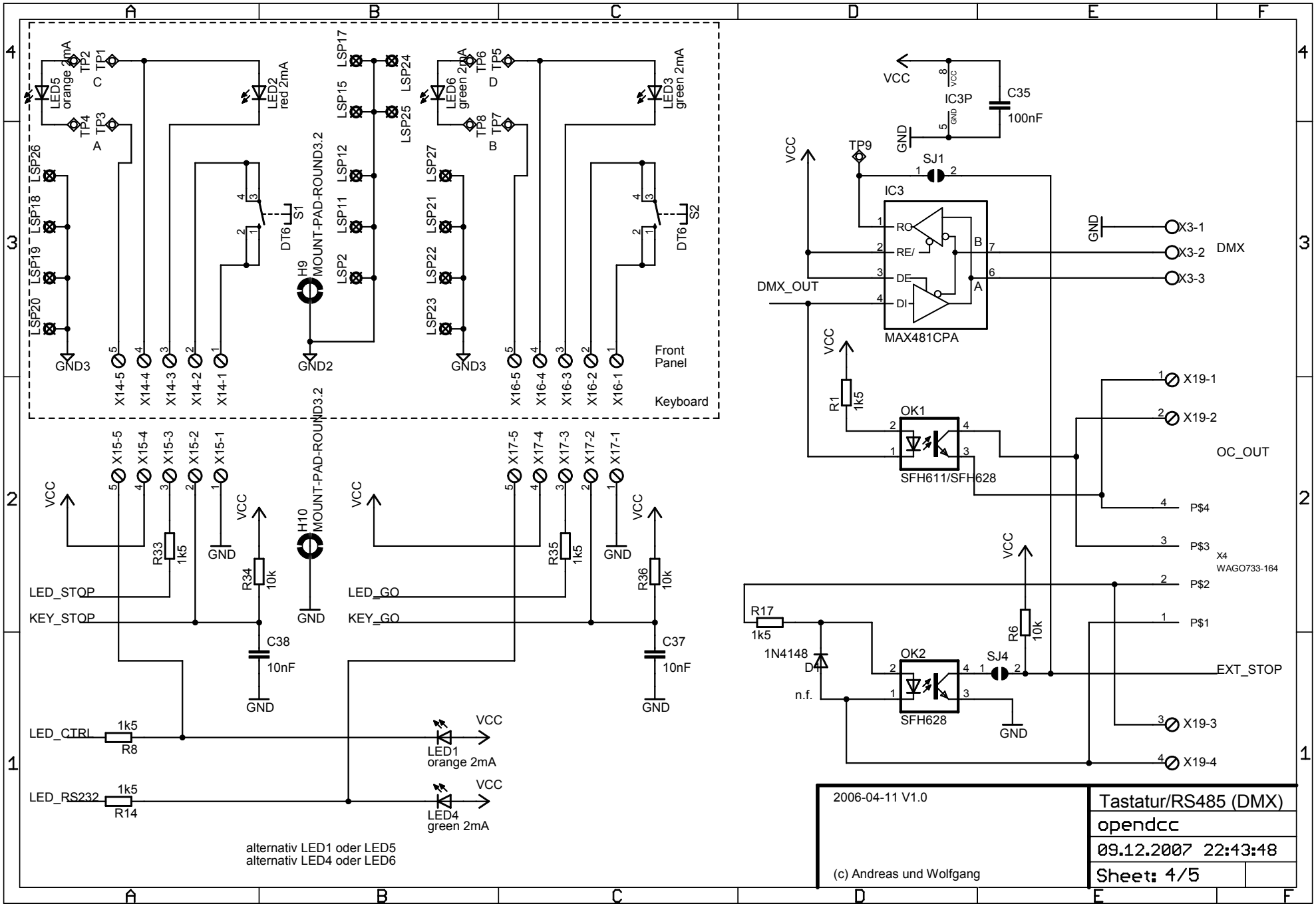
4.5 Schaltplan

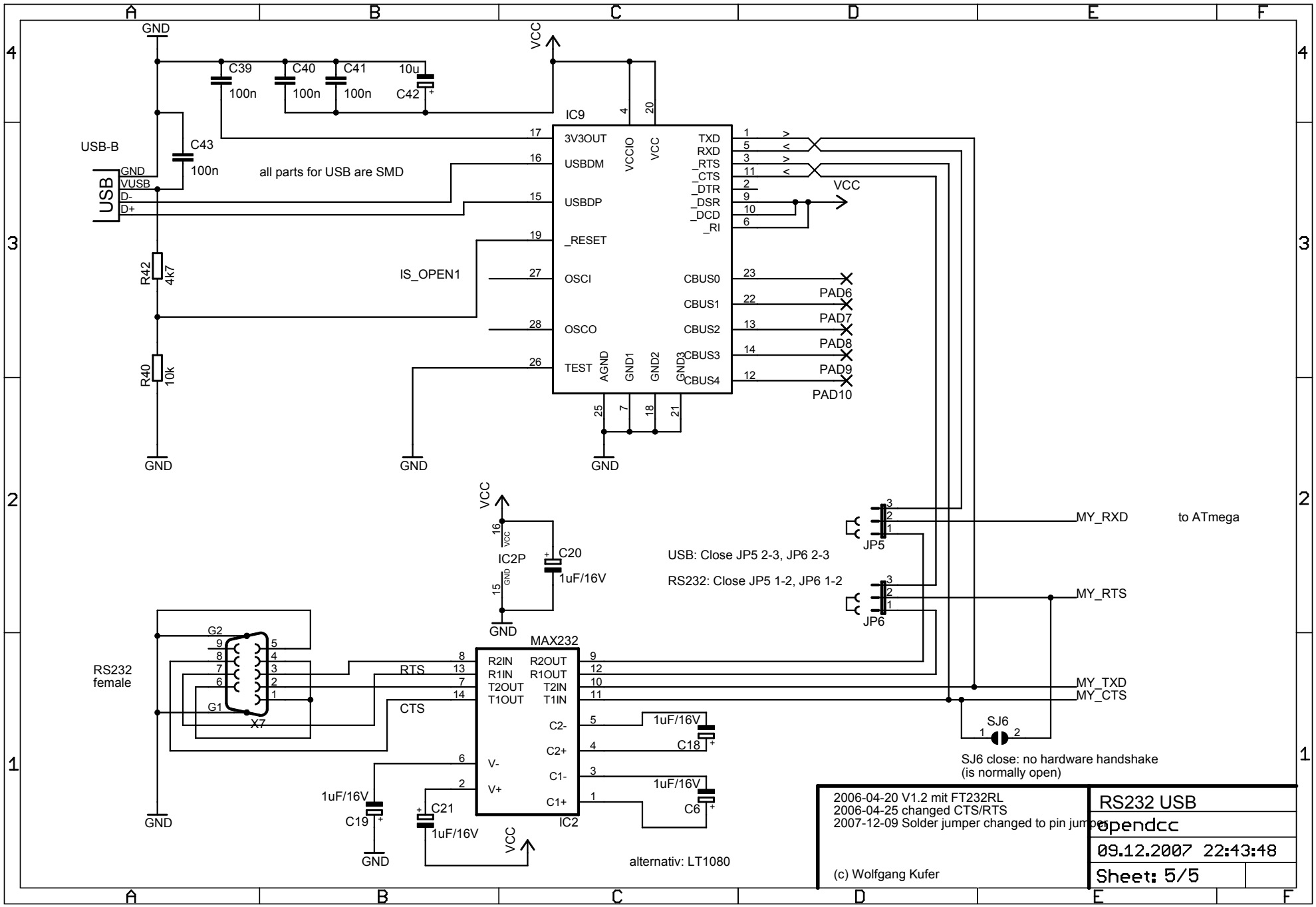
Abgebildet ist der derzeitige Schaltplan.


Power Supply
VDD
VCC
IC8
PTH08080 (TI)
5V / 2.2A
1N4004
D3
VIN
VOUT
NH
GND
ADJ
alternativ
R37
348R
X2-1
Innen +
X2-2
Mantel -
X13
100nF
C1
C8
IC4
7805
C2
C3
100nF
470uF/40V
100uF/10V
LSP13
LSP14
LSP16
H5
H6
H7
H17
MOUNT-PAD-ROUND3.2
H16
H14
MOUNT-PAD-ROUND3.2
If ADC is used:
cut line + L1
L1
10uH SMD
R5
10k
D8
1N4148
C7
100nF
U$1
MC34064
(not fitted)
VC
RES
GND
JTAG
TDI
TMS
TDO
TCK
X12
PonyProg
(please use ISP)
X11
female
(all not fitted)
see at www.lancos.com
IC5
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
Q1
16MHz
Q3
n.f.
alternativ
(isolate pads)
C4
18pF
C5
18pF
C17
100nF
C33
100nF
C34
100nF
LSP1
ATMEGA32-P
(ADC7)PA7
(ADC6)PA6
(ADC5)PA5
(ADC4)PA4
(ADC3)PA3
(ADC2)PA2
(ADC1)PA1
(ADC0)PA0
(SCK)PB7
(MISO)PB6
(MOSI)PB5
(SS)PB4
(AIN1/OC0)PB3
(AIN0/INT2)PB2
(T1)PB1
(T0/XCK)PB0
(TOSC2)PC7
(TOSC1)PC6
(TDI)PC5
(TDO)PC4
(TMS)PC3
(TCK)PC2
(SDA)PC1
(SCL)PC0
(OC2)PD7
(ICP)PD6
(OC1A)PD5
(OC1B)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
KEY_GO
KEY_STOP
LED_STOP
LED_GO
EXT_STOP
S88DIN3
S88DIN2
S88DIN1
SCK
MY_RTS
MY_CTS
MOSI
DMX_OUT
S88LOAD
ACK_DETECT
S88RESET
S88CLK
LED_RS232
LED_CTRL
TDI
TDO
TMS
TCK
SHORT_PROG
SHORT_MAIN
ENA_MAIN
ENA_PROGGLEIS
DCC
NDCC
MY_TXD
MY_RXD
RST
R30
4k7
R27
4k7
R29
4k7
D7
4V7
D2
4V7
D6
4V7
JP4
R44
100
(n.f.)
Bridge to S88-IF
DIL Cable
S88LOAD
S88RESET
S88CLK
S88DIN3
S88DIN2
S88DIN1
RAILDATA
X21-8
X21-7
X21-9
X21-6
X21-10
X21-5
X21-11
X21-4
X21-12
X21-3
X21-13
X21-2
X21-14
X21-1
2A
F2
JP2
JP1
74AHC1G125
IC10
R31
100
(Raildata)
(for S88-N)
JP3
MY_CTS
SCK
RST
MOSI
Mode:
IBOX
DMX
S88
tbd.
JP1
JP2
2006-04-11 V1.0
2006-04-16 added CTS, RTS; USB-Support
2006-04-23 V1.2 FT232RL Chip
2006-04-25 Added ISP and Resetchip (n.f.)
2006-05-15 Change C4-C5 -> 18pF
2007-12-08 V1.4 Added Raildata + S88-N
(c) Andreas Lang und Wolfgang Kufer
Prozessor
opendcc
09.12.2007 22:43:48
Sheet: 1/5


VDD
Charge Pump
D5
1N4148
100R
R13
C11
220nF
1N4148
D4
C24
10nF
C15
220u/40V
100nF
C12
100nF
C36
GND
IC1
VCP
VBOOT
VS_A
SENSE_A
RCA
OUT1A
OUT2A
VREF_A
VS_B
SENSE_B
RCB
OUT1B
OUT2B
VREF_B
IN1_A
IN2_A
EN_A
IN1_B
IN2_B
EN_B
GND
L6206N
Kühlfläche
Chip einlöten
SENSE_PROG to 2/B1
R32
1R
metal film
R10
1R
R3
0R15
100nF
C9
X6-4
X6-3
PROGGLEIS
DCC
NDCC
ENA_PROGGLEIS
2k2
R19
220k
R16
C22
4n7
R12
39k
39k <=> 0.5A
P$1
P$2
WAGO733-164
X5
from ATmega
ENA_MAIN
2k2
R20
220k
R18
C23
4n7
R4
0R15
100nF
C10
P$3
P$4
R7
15k
10k <=> 2.2A
15k <=> 1.5A
Kühlblech hier anlöten
LSP4
LSP5
LSP6
LSP7
LSP8
LSP3
LSP9
LSP10
X6-2
X6-1
DCC-OUT
VCC
t=10k*10n=0,1ms
1M
R23
1M
R24
R25
10k
SENSE_PROG
10k
R21
C13
10nF/MKS
C44
10nF
IC6A
LM393N
SHORT_MAIN
SHORT_PROG
ACK_DETECT
to ATmega
IC6B
LM393N
100nF
C16
13k
R22
C14
1uF/MKS
t=13k*1u=13ms
2006-03-16 first version
2006-04-11 V1.0
2006-04-25 R16 u. R18, damit bei Reset disab
2006-05-29 V1.3, C44 dazu, Spikes bei ACK
2007-03-10 V1.3, R19, R20 von 10k auf 2k2
(c) Andreas und Wolfgang
Booster
opendcc
09.12.2007 22:43:48
Sheet: 2/5


VCCIO
C32
10u/10V
GND1
X20-1
X20-14
X20-2
X20-13
X20-3
X20-12
X20-4
X20-11
X20-5
X20-10
X20-6
X20-9
X20-7
X20-8
SCLK
SRESET
SLOAD
IC7P
C27
100nF
R45
10k
not fitted
IC7A
74AC244N
R38
47R
(opt. for S88-N-Prog)
R41,R43, R45
close for normal op.
R41
1k
SJ2
R43
2k2
not fitted
IC7B
74AC244N
R28
47R
C26
1nF
SDATA1
R11
47R
C30
1nF
SDATA2
R15
47R
C31
1nF
SDATA3
SDIN3
SDIN2
SDIN1
R39
47R
RAILDATA1
SCLK1
SLOAD1
SRESET1
SCLK2
SCLK3
X10-1
X10-2
X10-3
X10-4
X10-5
X10-6
SRESET1
SLOAD1
SCLK1
R26
47R
SDATA1
C25
10nF
S88-N
S88-Port 1
Links / Left
X22
X8-1
X8-2
X8-3
X8-4
X8-5
X8-6
SRESET1
SLOAD1
SCLK2
R2
47R
SDATA2
C28
10nF
S88-N
S88-Port 2
Mitte / Middle
X23
X9-1
X9-2
X9-3
X9-4
X9-5
X9-6
SRESET1
SLOAD1
SCLK3
R9
47R
SDATA3
C29
10nF
S88-N
S88-Port 3
Rechts / Right
X24
Adapter Board
connected with
14 con. DIL-cable
16 con. may fit, too
MOUNT-PAD-ROUND3.2
H11
H8
MOUNT-PAD-ROUND3.2
MOUNT-PAD-ROUND3.2
H12
H13
MOUNT-PAD-ROUND3.2
GND
on MainBoard
on S88 Adapter
2006-03-16 first version
2006-04-11 V1.0
2006-05-28 V1.3 Swap RJ45 Conn, 1n->10nF
2007-12-08 V1.4 X22-X24 Changed to S88-N
(c) Andreas und Wolfgang
S88-Adapter
opendcc
09.12.2007 22:43:48
Sheet: 3/5


LED5 orange 2mA
LED2 red 2mA
LED6 green 2mA
LED3 green 2mA
TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP8
C
A
D
B
LSP17
LSP24
LSP15
LSP25
LSP12
LSP27
LSP26
LSP18
LSP11
LSP21
LSP19
LSP2
LSP22
LSP20
LSP23
S1
S2
DT6
H9 MOUNT-PAD-ROUND3.2
GND3
GND2
X14-5
X14-4
X14-3
X14-2
X14-1
X16-5
X16-4
X16-3
X16-2
X16-1
Front Panel
Keyboard
H10 MOUNT-PAD-ROUND3.2
X15-5
X15-4
X15-3
X15-2
X15-1
X17-5
X17-4
X17-3
X17-2
X17-1
VCC
GND
R33 1k5
R34 10k
R35 1k5
R36 10k
LED_STOP
KEY_STOP
LED_GO
KEY_GO
C38 10nF
C37 10nF
LED_CTRL
LED_RS232
R8 1k5
R14 1k5
LED1 orange 2mA
LED4 green 2mA
alternativ LED1 oder LED5
alternativ LED4 oder LED6
IC3P
C35 100nF
TP9
SJ1
IC3
RO
RE/
DE
DI
B
A
MAX481CPA
DMX_OUT
X3-1
X3-2
X3-3
DMX
R1 1k5
OK1
SFH611/SFH628
X19-1
X19-2
OC_OUT
P$4
P$3
X4
WAGO733-164
P$2
P$1
R17 1k5
1N4148
D1
n.f.
OK2
SFH628
SJ4
R6 10k
EXT_STOP
X19-3
X19-4
2006-04-11 V1.0
(c) Andreas und Wolfgang
Tastatur/RS485 (DMX)
opendcc
09.12.2007 22:43:48
Sheet: 4/5


GND
VCC
C39
100n
C40
100n
C41
100n
10u
C42
IC9
USB-B
C43
100n
all parts for USB are SMD
USB
GND
VUSB
D-
D+
3V3OUT
USBDM
USBDP
_RESET
OSCI
OSCO
TEST
VCCIO
VCC
AGND
GND1
GND2
GND3
TXD
RXD
_RTS
_CTS
_DTR
_DSR
_DCD
_RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
R42
4k7
R40
10k
IS_OPEN1
PAD6
PAD7
PAD8
PAD9
PAD10
IC2P
C20
1uF/16V
JP5
JP6
USB: Close JP5 2-3, JP6 2-3
RS232: Close JP5 1-2, JP6 1-2
MY_RXD
to ATmega
MY_RTS
MY_TXD
MY_CTS
SJ6
SJ6 close: no hardware handshake
(is normally open)
RS232
female
X7
G1
G2
RTS
CTS
MAX232
R2IN
R1IN
T2OUT
T1OUT
R2OUT
R1OUT
T2IN
T1IN
C2-
C2+
C1-
C1+
V-
V+
IC2
C18
C6
C19
C21
alternativ: LT1080
2006-04-20 V1.2 mit FT232RL
2006-04-25 changed CTS/RTS
2007-12-09 Solder jumper changed to pin jumper
(c) Wolfgang Kufer
RS232 USB
opendcc
09.12.2007 22:43:48
Sheet: 5/5

4.6 Layout

Das vollständige Dokument als Download siehe Kapitel 4.1.4 auf Seite 120.

5 Nachrüstung Xpressnet™

5.1 Einleitung

Mit Xpressnet™ [15] lassen sich Eingabegeräte, PC-Interfaces und stationäre Decoder bzw. Rückmelder verbinden. Viele handelsübliche Eingabegeräte (u.a. auch die Multimaus von Roco [16]) basieren auf diesem Bus. Mit dem nachfolgend beschriebenen Adapter lassen sich Xpressnetgeräte an OpenDCC anschließen.

5.2 Übertragungstechnik

Die Übertragung erfolgt mit RS485 im Halbduplex, das ist eine differentielle Übertragungstechnik auf zwei Leitungen, diese sind an der Zentrale mit 120 Ohm abgeschlossen. Es ist linienförmige und sternförmige Topologie möglich, Netz- und Ringstrukturen sind nicht erlaubt. Die max. Ausdehung ist mit 1000m angegeben, lt. Lenz sollen dabei aber dann linienförmige Strukturen und ein Abschluß am weitest entfernten Teilnehmer vorgesehen werden.
Die Sender sind kurzzeitig kurzschlussfest, so dass kurzzeitiges Gegensenden zweier Sender nicht zu Defekten führt. Auch können Geräte während des Betriebes ab- und angesteckt werden.

Es wird mit 9 Bit, keinem Paritätsbit und einem Stopbit übertragen, die Übertragungsrate beträgt 62500 Baud. RS485 könnte mehr (bis zu 10MBit auf kurzen Strecken), allerdings wurde die Geschwindigkeit wegen leichterer Verarbeitung in den Endgeräten und Freiheit bei der Netztopologie (Stichleitungen) auf 62,5 kBaud reduziert.

Aus elektrischen Gründen können bis zu 32 Teilnehmer angeschlossen werden. Jeder Teilnehmer verursachte eine Last am Bus (sog. Unit Loads, eine hypothetische Lastgrösse). Diese Teilnehmerzahl spiegelt sich auch in der Protokollebene (s.u.) wieder. Allerdings sind mittlerweile viele Bustransceiver auch mit geringeren Buslasten verfügbar, so dass inzwischen elektrisch auch mehr Teilnehmer möglich sind.

5.3 Physikalische Ausführung, Stecker

Es gibt zwei verschiedene Steckerausführungen:

- **DIN-Rundstecker**
  Xpressnet, DIN Stecker

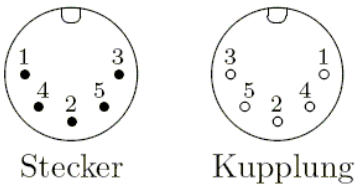


Abb. 38: Xpressnet - DIN Stecker (Quelle: archiv)

Pin	Farbe	Name	Beschreibung
1	gelb	L	Power, +12V
2	-	-	offen
3	schwarz	M	Masse, GND
4	grün	A	RS485 +
5	rot	B	RS485 -

- **Modularstecker**
  Dieser Stecker ist der heute gebräuchliche Anschluß. Es wird RJ12 (wie RJ11, aber alle 6 Pins belegt) verwendet, die Pinbelegung ist gemäß folgender Tabelle.

Abb. 39: Xpressnet, RJ12 Stecker (Quelle: archiv)

Pin	Farbe	Name	Beschreibung
1	weiß	C	Gleissignal, DCC (optional)
2	schwarz	M	Masse, GND
3	rot	B	RS485 -
4	grün	A	RS485 +
5	gelb	L	Power, +12V
6	blau	D	Gleissignal, DCC (optional)

Das Gleissignal C und D ist optional und dient z.B. der Versorgung von rückmeldefähigen Decodern oder Boostern. Es wird ein übliches 4-adriges oder 6-adriges Telefonkabel verwendet. Die Leitungen A und B sind verdrillt auszuführen.

- Buszuteilung, Arbitrierung Das Bustiming wird von der Zentrale vorgegeben. Die Zentrale sendet entweder Buszuteilungen (sog. Token) oder Nachrichten und schaltet dann fallweise für eine bestimmte Zeit die Busrichtung um. Jede Übertragung von der Zentrale wird mit einem besonderen Byte, dem sog. **Callbyte** begonnen. Dieses Callbyte enthält den Adressat der Nachricht (=client, Wert 1..31) sowie 2 Bits, welche den Typ der Nachricht bezeichen. Beim Callbyte ist immer das Bit 8 (also das MSB) gesetzt, daran können die Clients den Beginn einer Nachricht erkennen (framing). Nachrichten an den Client 0 sind an alle adressiert (=Broadcast).

  Zur Vergabe des Busses sendet die Zentrale in beliebiger Reihenfolge Tokens an die Busteilnehmer (Clients). Ein Token besteht nur aus dem Callbyte. Dieses enthält die Adresse des Clients (1..31), 0x40, und das Bit 8 (also das MSB) ist gesetzt. Nach dem Empfang eines Tokens darf der angesprochene Client innerhalb eines Zeitfensters von 80µs eine (und nur eine) Anforderungsnachricht abschicken, die fallweise von der Zentrale beantwortet wird.

  Die übertragenen Nachrichten werden in Paketen zusammengefaßt, diese bestehen aus:

Xpressnet Nachrichten		
1 Byte	1 - 14 Bytes	1 Byte
Header	Nutzinhalt	Prüfsumme
Kennung und Länge des Paketes	z.B. Lokadresse und Fahrstufe	XOR über Header und Nutzinhalt

- **Probleme und Mängel**

- Zahl der Busteilnehmer beschränkt.
  Xpressnet™ kann 31 Busteilnehmer ansprechen - auch wenn inzwischen RS485 da schon mehr zuläßt, so ist es doch auf der Protokollebene auch auf 31 beschränkt. Und das kann dann doch schnell zusammenkommen: ein paar Bediengeräte, PC-Interface, Rückmelder, Railcom - Detektoren ...

- Adressraum für Zubehördekoder und Rückmelder
  Xpressnet™ kann nur 1024 verschiedene Zubehördekoder ansprechen, dies ist eine deutliche Einschränkung gegenüber der DCC-Norm, wo 2040 Adressen zur Verfügung stehen. Darüber hinaus ist dieser Adressraum in der unteren Hälfte gleichzeitig von Rückmeldern belegt (es wird im Protokoll nicht unterschieden). Zudem ist die Bitzuordnung der Abfragebefehle etwas "unglücklich", so dass man eigentlich nur der Empfehlung von Lenz folgen kann und bis 64 die Weichen, danach die Rückmelder einordnet. Folge: es sind nur noch 256 Weichen sinnvoll ansprechbar.

- Erweiterte Zubehördekoder
  Für Signaldecoder und auch für Mehrwegweichen ist in DCC ein erweiterter Zugriff vorgesehen, in welchem direkt die Signalbegriff oder der Abzweig gestellt wird. Xpressnet kann diese Art von Befehlen nicht ansprechen.

- Programmieren für Zubehördekoder
  Xpressnet™ kann Zubehördekoder nicht per PoM ansprechen - erstaunlich bei einer Firma, welche technologische Führerschaft bei DCC für sich reklamiert.

5.4 Schaltung des Adapters

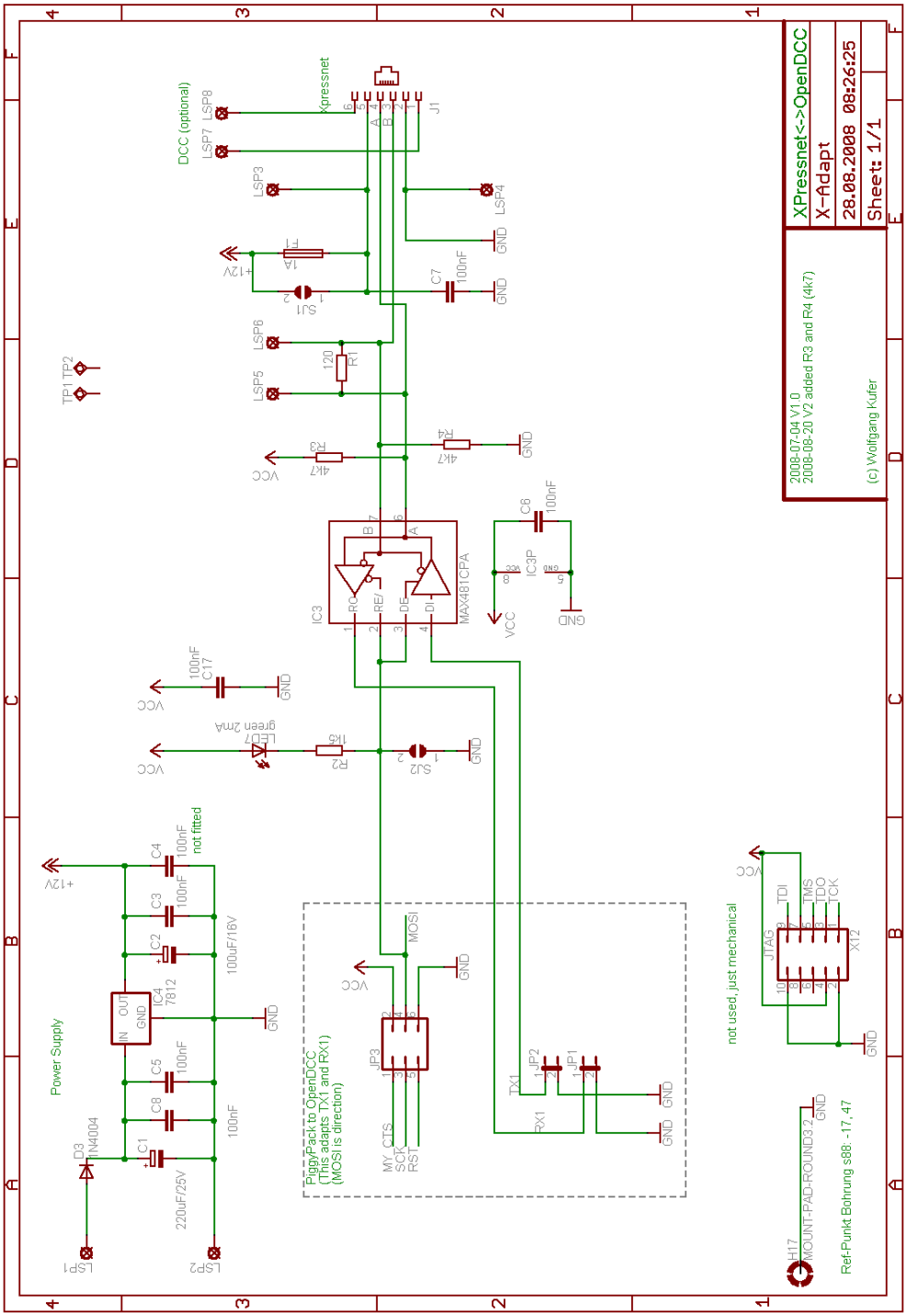


Abb. 40: Schaltung des Xpressnet -Adapter (Quelle: archiv)

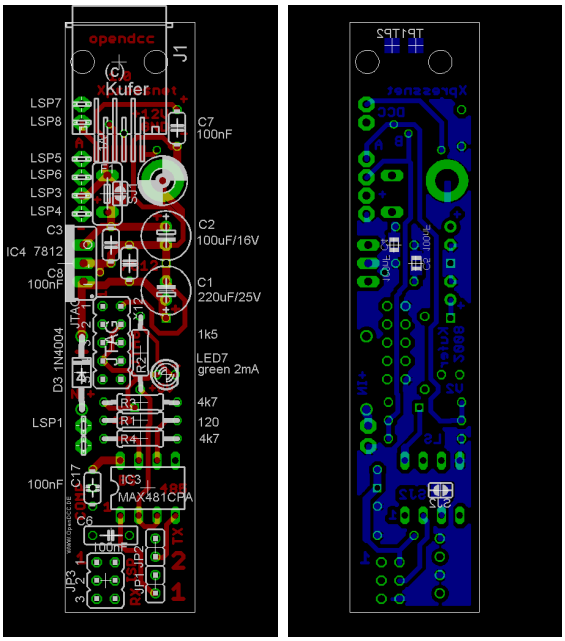
Die Schaltung besteht aus zwei Teilen:

- **Stromversorgung:**
  Xpressnetgeräte brauchen einen Stromversorgung von 12V. Diese wird mittels eines üblichen Längreglers aus der Versorgung der OpenDCC-Zentrale gewonnen. Laut Spec darf ein Xpressnet-Client 20mA Strom entnehmen. Bei ausgedehnten Xpressnet™ Installationen ist dieser Regler fallweise zu kühlen oder es müssen die 12V mit einem leistungsfähigem Netzteil extern erzeugt werden. In diesem Fall darf die Sicherung F1 nicht bestückt werden. SJ1 bleibt normalerweise offen, nur wer es gerne ohne Sicherung mag, kann SJ1 schließen ...

- **RS485 Empfänger und Sender:**
  Xpressnet™ wird im Halbduplex über eine RS485 Verbindung übertragen. Baudrate ist 62,5kBaud, es werden 9Bit ohne Parity mit einem Stopbit übertragen. Der Baustein Max485 (oder Max481) übernimmt die Daten von der seriellen Schnittstelle des Prozessors und wandelt sie in das differentielle R485 Signal um. Eine LED zeigt die Senderichtung an.
  Sollte die Schaltung als reiner Empfänger benutzt werden, so ist durch Schließen der Lötbrücke SJ2 der Empfänger dauerhaft freizuschalten. Als Sende-Empfangschip eignet sich Max485 oder auch LTC485. Empfohlen ist auf alle Fälle die Verwendung eines flankenbegrenzten (slew rate limited) tranceivers, dieser hilft bei Störstrahlproblemen und bei Reflexionen.

  Ich habe bei Tests mit verschiedenen Transcievern festgestellt, daß die differentielle RS485 Leitungen ev. einen undefinierten Zustand einnehmen, wenn weder Zentrale noch Slave senden. Zur Abhilfe empfiehlt sich ein Pullup von 4k7 gegen 5V auf der Leitung A und ein Pulldown von 4k7 auf der Leitung B.

5.5 Layout



(a) Bestückungsseite (b) Lötseite

Abb. 41: Adapterplatine (Quelle: archiv)

Der Adapter ist 18 x 82 mm groß.

5.6 Einbau und notwendige Änderungen

Abb. 42: eingebauter Xpressnet™ Adapter (Quelle: archiv)

Der Adapter ist so konzipiert, dass er auf die Steckstifte für JP1 und JP2 sowie der Bootschnittstelle aufgesteckt werden kann. Einzig die Stromversorgung (+15V) des Längsreglers für Xpressnet™ muß mit einer separaten Leitung vom Stromversorgungsstecker der Zentrale nachverdrahtet werden. Ich verwende lange Stifte (für Wire-Wrap), um den notwendigen Leiterplattenabstand von 20mm zu erreichen. Die Xpressnetschnittstelle ist dann direkt von der Rückwand her zugänglich, hierzu muß ein entsprechender Ausschnitt erstellt werden.

Notwendige Änderungen:
Um die erforderliche zweite serielle Schnittstelle zu erhalten, muß der Prozessor Atmega32 von OpenDCC durch einen **Atmega644P** ersetzt werden. Wichtig ist der Suffix P (Picopower), nur dieser Typ enthält zwei USARTs.

Natürlich muß dieser neue Prozessor mit der entsprechenden Software (Kapitel 3.1.2 auf Seite 40) geladen werden.

5.7 Test, erstes Einschalten

Nun wird zum ersten Mal Spannung angelegt, dabei wird die Ausgangsspannung am Adapter kontrolliert. Erst danach dürfen Eingabegeräte angeschlossen werden. Die Roco multiMaus® [16] meldet Err13, wenn sie keine Verbindung zur Zentrale hat.

5.8 Steckdosen für Xpressnet™

Für die Installation in einer Anlage empfehlen sich die folgend beschriebenen Einbausteckdosen.

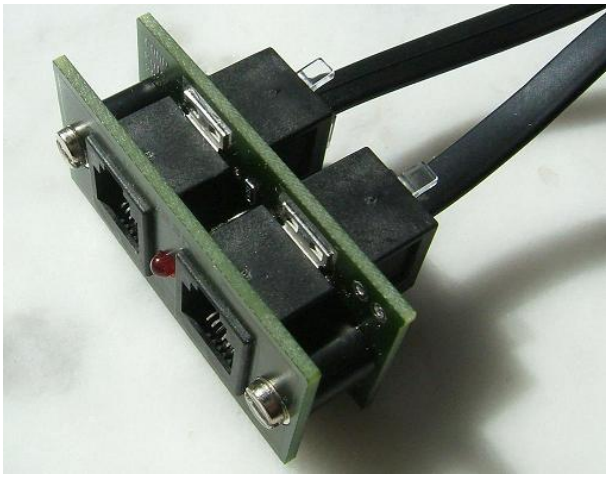


Abb. 43: Steckdosen für Xpressnet™ (Quelle: archiv)

Für Handregler auf Xpressnet-Basis braucht man an der Anlage Steckdosen, um den Handregler an der jeweils geeigneten Stelle einstöpseln zu können. Xpressnet kann einfach verlegt werden, normalerweise werden übliche Telefonleitungen verwendet. Die hier vorgestellt Schaltung verwendet vier RJ12 Buchsen, je zwei von der Bestückungsseite und von der Lötseite.

5.9 Schaltplan der Steckdosen

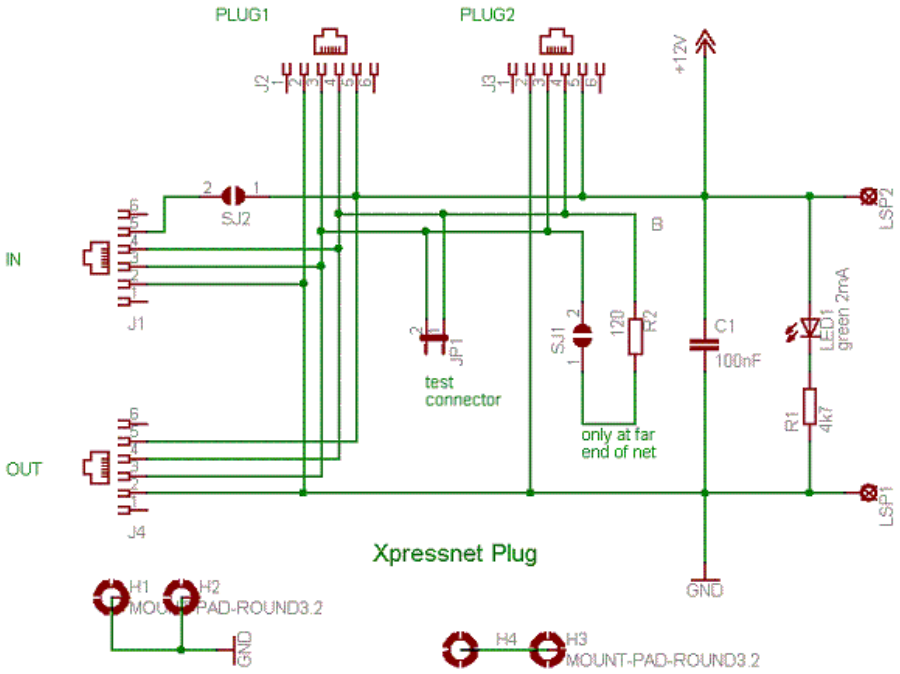
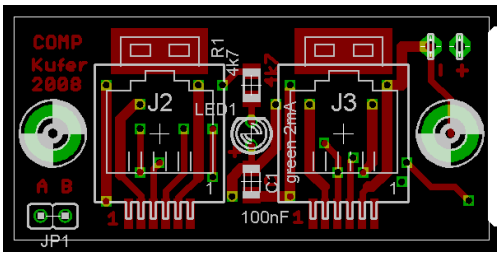


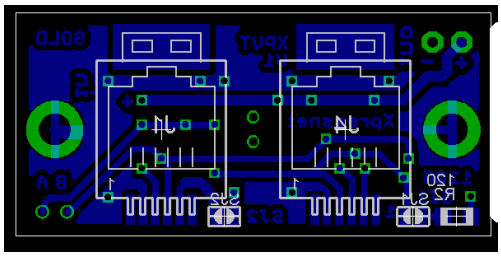
Abb. 44: Schaltplan der für Xpressnet™ (Quelle: archiv)

Der Abschlußwiderstand 120 Ohm ist nur bei ausgedehnten Anlagen am Ende des Netzes erforderlich, SJ1 bleibt daher i.d.R. offen.
Normalerweise wird das Xpressnet von der Zentrale versorgt (SJ2 geschlossen), sollte eine externe Versorgung notwendig sein, so wird SJ2 offen gelassen und die +12V können separat eingespeist werden.

5.10 Layout der Steckdosen



(a) Bestückungsseite



(b) Lötseite

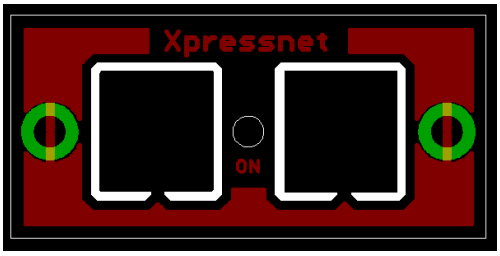
Abb. 45: Steckdosenplatine (Quelle: archiv)

5.11 Bestückung der Steckdosen

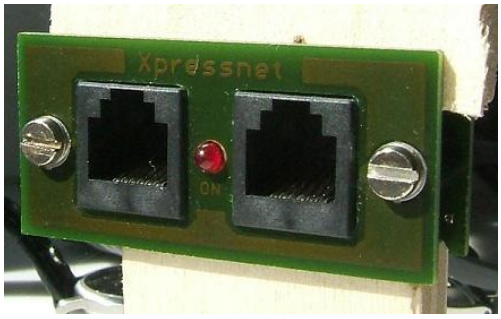
Hierzu gibt es nich viel zu sagen - die Buchsen sind SMD Bauteile von Tyco/AMP, dadurch konnte ein kompaktes Layout erreicht werden. Zuerst sollten die SMD Bauteile oben bestückt werden, dann unter Zuhilfenahme des Abdeckrahmens die Buchsen oben und die LED; Dann erst die Buchsen unten.

5.12 Einbau der Steckdosen

Mit dem gleichen Maßen wie die Platine (23 * 49mm) habe ich auch eine Frontplatte (als Leiterplatte) erzeugt, damit kann die Steckdose dann einfach in Außenwand der Anlage geschraubt werden und es ergibt sich ein sauberes Gesamtbild.



(a) Frontansicht



(b) Foto der Frontansicht

Abb. 46: Steckdosenplatine fertig (Quelle: archiv)

5.13 Stückliste der Steckdosen

Bauteil	Wert	Device	Package	Beschreibung
C1	100nF	C-EUC1206	C1206	Kondensator
J1	1-338086-3	RJ12 PLUG	SMD MOUNT	AMP Verbinder
J2	1-338086-3	RJ12 PLUG	SMD MOUNT	AMP Verbinder
J3	1-338086-3	RJ12 PLUG	SMD MOUNT	AMP Verbinder
J4	1-338086-3	RJ12 PLUG	SMD MOUNT	AMP Verbinder
LED1	green 2mA	LED3MM	LED3MM	LED
R1	4k7	R-EU_R1206	R1206	Widerstand
R2	120	R-EU_R1206	R1206	Widerstand
PCB				X-Verteiler

6 Abschalten der Booster

An OpenDCC lassen sich Booster und externe Nothalttaster anschließen. Damit kann von jeder Stelle der Anlage ein Halt ausgelöst werden bzw. bei ausgelöstem Halt auch externe Booster abgeschaltet werden. Die entsprechende Kontrollleitung ist als Ringleitung über alle Taster zu verbinden.

6.1 Schaltung

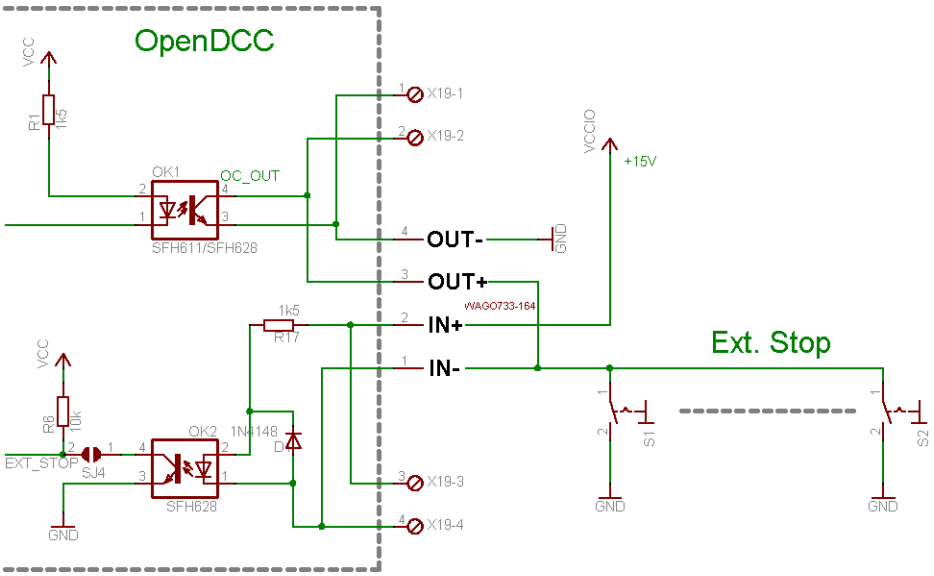


Abb. 47: Kontrolle der Booster und externes Abschalten (Quelle: archiv)

Wenn über OK2 ein Strom fließt, so wird die Zentrale abgeschaltet - sie geht in den Zustand STOP. Dieser wird durch den aktiven Ausgang OK1 angezeigt. Weitere Taster werden so angeschlossen, wie im Schaltbild angegeben.

6.2 Erforderliche Einstellungen

Damit OpenDCC diese Ein- und Ausgänge so behandelt, muß CV36 (Kapitel 3.3.2 auf Seite 88) auf 1 programmiert werden. SJ4 muß geschlossen sein.

6.3 Einschränkungen

Die Optokoppler können für verschiedene Aufgaben verwendet werden. Logischerweise muß man sich entscheiden für welche Aufgabe - deshalb ist der externe Nothalt nicht zusammen mit der Weichenrückmeldung (Kapitel 3.5 auf Seite 93) verwendbar.

7 Hintergrundinformationen

7.1 Umsetzer für S88 nach RJ45

Beschreibung (V1.2 und V1.3):

Um die potentiellen Übertragungsprobleme bei S88 zu vermeiden, entstand eine kleine Platine für S88 nach RJ45 Umsetzung. Diese kann beim GBM16XS der Fa. Blücher[9] und bei der Intellibox eingesteckt werden. OpenDCC kann von vornherein alternativ mit den S88-Stiften oder der RJ45-Buchse bestückt werden. (OpenDCC kann sowohl die Intellibox als auch das HSI88 emulieren).

Mittlerweile gibt es eine Normierung dieser Umsetzung: siehe S88-N (Kapitel 7.2 auf Seite 166)

Adapter (OpenDCC-Belegung (bis Hardware V1.3)):

Die beiden Stecksystem haben folgende Pinbelegung:

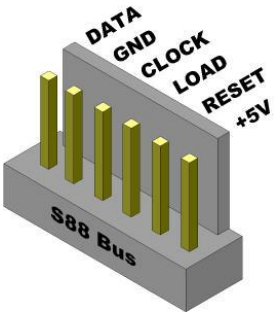


Abb. 48: Pinbelegung OpenDCC (Quelle: Autor)

- **S88:**

Pin	Name	Beschreibung
1	DATA	Auslesedaten
2	GND	Masse für Signale und Versorgungsspannung
3	CLOCK	Taktsignal für die Synchronisation
4	LOAD	Lade die Informationen in den Bus
5	RESET	Zurücksetzen der Eingangsspeicher
6	+5V	Versorgungsspannung für die Rückmeldemodule

- **RJ45:**

  Paare sind auf den Pins 1,2 - 3,6 - 4,5 - 7,8; d.h. immer "farbe"-weiß und "farbe" sind gepaart.
  übliche Farbkodierung (TIA568A):

  1 - ws/gn = weiß-grün
  2 - gn = grün
  3 - ws/or = weiß-orange
  4 - bl = blau
  5 - ws/bl = weiß-blau
  6 - or = orange
  7 - ws/br = weiß-braun
  8 - br = braun

  Pin 1 am Stecker ist von vorne gesehen, Kontakte oben, rechts

  Wenn man die empfindlichen Signale (CLOCK, LOAD, RESET) jeweils mit Ground paart, so ergibt sich folgende, sinnvolle **Kabelumsetzung** (OpenDCC-Belegung):

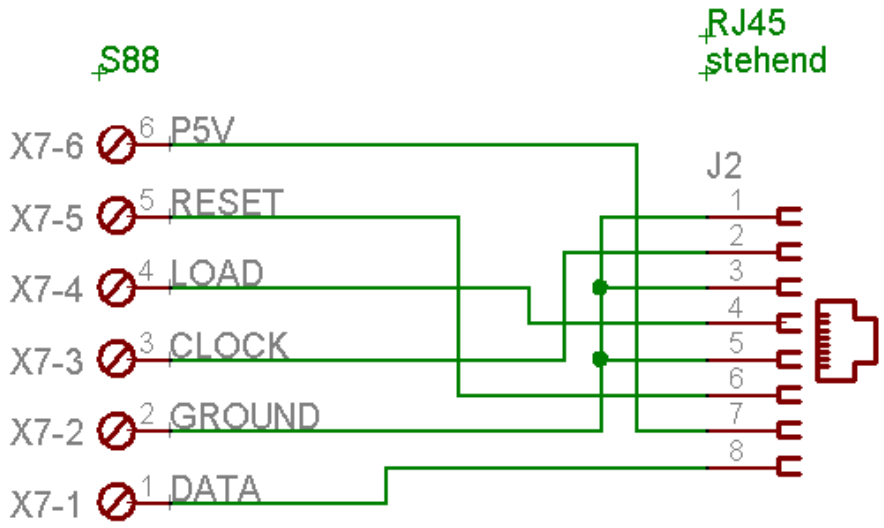
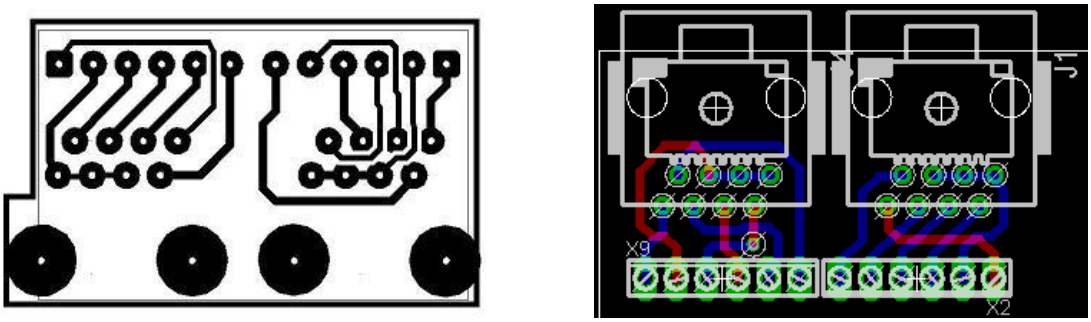


Abb. 49: Kabelumsetzung S88 - RJ45 (Quelle: Autor)

Pin S88	Name	Beschreibung	Pin RJ45	Farbe
1	DATA	Auslesedaten	8	br
2	GND	Masse für Signale und Versorgungsspannung	1	ws/gn
2	GND		3	ws/or
2	GND		5	ws/bl
3	CLOCK	Taktsignal für die Synchronisation	2	gn
4	LOAD	Lade die Informationen in den Bus	4	bl
5	RESET	Zurücksetzen der Eingangsspeicher	6	or
6	+5V	Versorgungsspannung für die Rückmeldemodule	7	ws/br

Hinweis: Es gibt mittlerweile eine Normung: s88-N (Kapitel 7.2 auf Seite 166). OpenDCC verwendet ab V1.4 die neue Norm. Adapter nach alter und neuer Norm können gemischt eingesetzt werden, wenn darauf geachtet wird, dass an *einem* RJ45-Kabel vorn und hinten die gleiche Norm Verwendung findet.

- **Layout:**



(a) Einseitig (b) Zweiseitig

Abb. 50: Planinenadapter (Quelle: archiv)

- **Bauteile:**

Folgende Bauteile können verwendet werden (Nummer gemäß reichelt.de):

- MEBP 8-8 (Doppelbuchse zum Kabel verlängern)
- MEBP 8-8S (Platinenstecker)
- BL 1X20W8 2,54 (S88-Buchse; diese ist 20 polig, da kann man 3 Buchsen mit 6 Polen daraus machen)
- Kabel: diese gibt es auch bei Reichelt, nach RJ45 suchen
- MEB 8-8 (RJ45 Buchse mit Kabeln)

- **Bilder:**

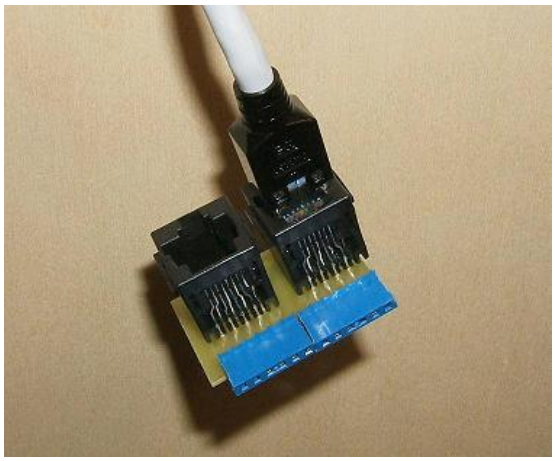


Abb. 51: Adapter mit Kabel (Quelle: Autor)

Abb. 52: Die Rückmelder (GBM16XS) der Hauptwendel mit eingesteckten Adaptern (Quelle: Autor)

7.2 S88 über Netzwerkkabel

- Beschreibung (ab V1.4):

Der S88-Bus hat sich als preiswerter Rücklesebus für die Modellbahn etabliert. Das Prinzip ist einfach: der S88-Bus ist ein serielles Schiebe-Register mit parallelem Load-Eingang.

Um Störung bei der Übertragung (Kapitel 7.1 auf Seite 161) zu minimieren, wird die Übertragung mittels preiswerter Netzwerkkabel durchgeführt. Diese bietet gute Eigenschaften: verdrillt, geschirmt und sicheres, einfaches Steck-system.

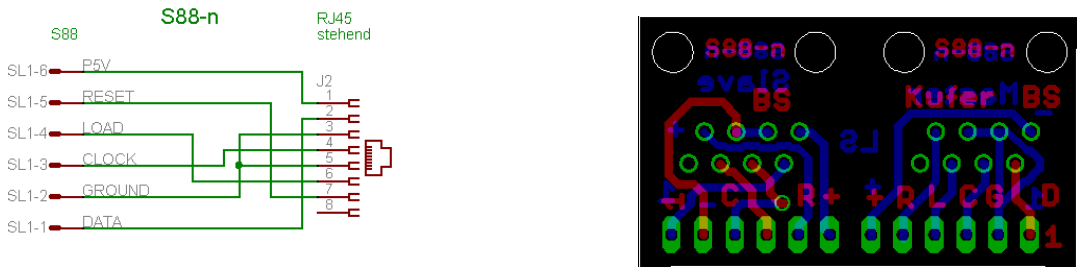
Die Vorteile CAT-5 Kabel und RJ-45 Steckverbinder haben auch andere Anbieter erkannt. Leider gibt es bisher verschiedenen Pinbelegungen. Mitte 2007 wurde eine Normung durchgeführt - zukünftige Entwicklungen sollen die Normbelegung s88-N verwenden. Mittlerweile haben schon einige Anbieter Produkte angekündigt, u.a. Tams Elektronik, Blücher Elektronik, rocrail und moba-digital.net.
Auch OpenDCC unterstützt ab Version 1.4 s88-N.

- Pinbelegung:

s88-N: Normung S88 auf Netzwerkkabel (CAT-5, RJ45)

Pin S88	Name	Beschreibung	Pin RJ45	Farbe
1	DATA	Auslesedaten	2	gn
2	GND	Masse für Signale und Versorgungsspannung	3	ws/or
2	GND		5	ws/bl
3	CLOCK	Taktsignal für die Synchronisation	4	bl
4	LOAD	Lade die Informationen in den Bus	6	or
5	RESET	Zurücksetzen der Eingangsspeicher	7	ws/br
6	+5V	Versorgungsspannung für die Rückmeldemodule	1	ws/gn
-	RAILDATA	Gleissignal	8	br

- Adapter:



(a) Schaltplan Adapter s88-N (b) Layout Adapter s88-N

Abb. 53: Adapter (Quelle: archiv)

- Bauteile:

  Folgende Bauteile können verwendet werden (Nummer gemäß reichelt.de):

  - MEBP 8-8 (Doppelbuchse zum Kabel verlängern)
  - MEBP 8-8S (Platinenstecker)
  - BL 1X20W8 2,54 (S88-Buchse; diese ist 20 polig, da kann man 3 Buchsen mit 6 Polen daraus machen)
  - Kabel: diese gibt es auch bei Reichelt, nach RJ45 suchen
  - MEB 8-8 (RJ45 Buchse mit 70mm Kabel)

Eine Lösung ohne Adapter ist mit Modular-Einbaubuchsen 8-8 mit Anschlusskabel (Reichelt MEB 8-8) möglich. Nachfolgende Abbildung zeigt ein Beispiel in Verbindung mit dem Gleisbesetztmelder von Ratschmeier.

Abb. 54: Gleisbesetztmelder von Ratschmeier (Quelle: Autor)

- Übersicht und Vergleich bisheriger Belegungen:

  Übersicht bisheriger Belegungen (Anbieter/ Norm)

Pin RJ45	digital-bahn.de	opendcc.de	railway-lauf.de	iek.de	s88-N
1	+12V	GND	DATA	DATA	+12V/+5V
2	DATA	CLK	GND	DATA	DATA
3	GND	GND	GND	GND	GND
4	CLK	PS	CLK	CLK	CLK
5	GND	GND	PS	PS	GND
6	PS	RESET	RESET	RESET	PS
7	+12V	+5V	+5V	+5V	RESET
8	RAILDATA	DATA	+5V	+5V	RAILDATA

  Hierzu folgende Anmerkungen:

  – Die bisherigen Belegungen von railway-lauf.de und iek.de sind elektrisch ungünstig, weil die empfindlichen Leitungen nicht mit statischen Leitungen gepaart wurden. Beide sind wohl aus dem Wunsch nach einem einfachem Layout entstanden. Zudem ist bei IEK der GND nicht verstärkt.

- Eine mögliche höhere Versorgung auf 12V und entsprechende Verschiebung der Schaltschwelle bringt mehr Spielraum, damit sich Störungen nicht auswirken können. Deswegen wurde neben der 5V Versorgung auch die 12V zugelassen. Der Anwender muß beachten, dass Nur wenn alle angeschlossenen Module 12V-tolerant sind, dürfen die 12V von Zentrale oder vom s88-Booster verwendet werden. Allerdings birgt die 12V auch die Gefahr von Anwendungsfehlern, so dass meiner Meinung nach nur 5V verwendet werden sollte.
- Bei der Belegung von digital-bahn.de wurde die Resetleitung weggelassen, was fallweise Kompatibilitätsprobleme bringt. In der s-88-N ist diese enthalten.
- Die optimale GND Versorgung der Belegung von opendcc wurde zugunsten RAILDATA (d.h. das Gleissignal) etwas abgeschwächt.

7.3 Analyse des S88-Busses

- **Beschreibung:**

  Der S88-Bus hat sich als preiswerter Rücklesebus für die Modellbahn durchgesetzt. Das Prinzip ist einfach: der S88-Bus ist ein serielles Schiebe-Register mit parallelem Load-Eingang.

  Weitere Teilnehmer dieses Busses werden durch einfaches Kaskadieren angeschlossen, so entsteht ein langes Schieberegister, in dem alle auszulesenden Bits in einer langen Kette liegen. Diesem Vorteil des einfachen Aufbaus stehen allerdings Nachteile gegenüber: Es ist keine Adressvorgabe der Rückmelder möglich und die Übertragung erfolgt vollkommen ungeschützt, d.h. es gibt weder Parity, Prüfsumme oder CRC.

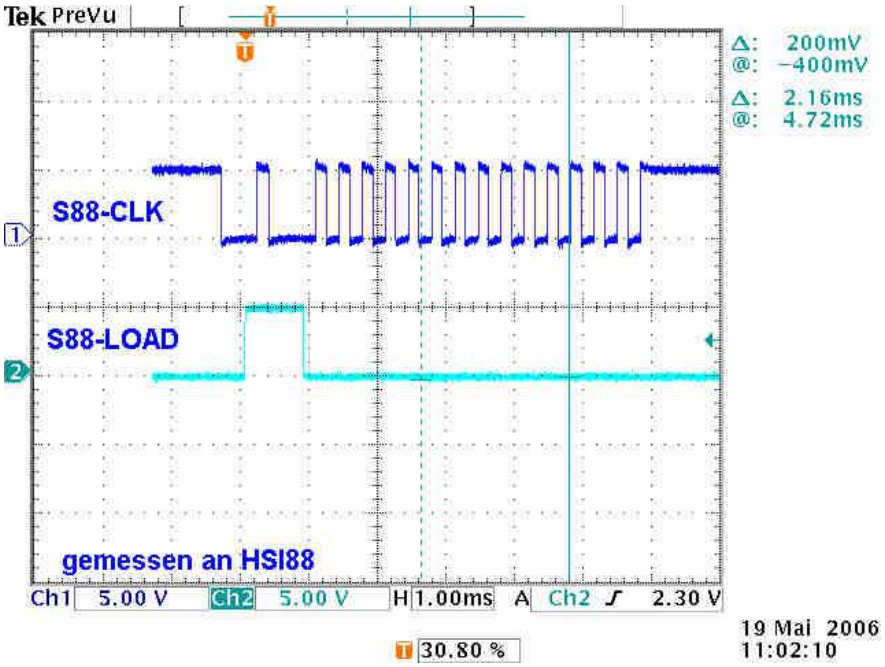


Abb. 55: Auslesevorgang - Signalverlauf auf dem Bus (Quelle: Autor)

Die LOAD-Leitung geht auf 1, darauf erfolgt ein Schiebtakt; alle Register in der Scheibekette übernehmen bei diesem Takt die Information an ihren Paralleleingängen. Als nächstes erfolgt ein RESET-Puls (auch aktiv high), dieser löscht die den Paralleleingang vorgeschalteten Latches, welche damit wieder bereit für die Übernahme neuer Information sind. Die Latches speichern auch kurze Signale bis zur nächsten Abfrage.

Nun wird die Schieberegister (mit LOAD = 0) geschoben. Dadurch dass jeweils der Datenausgang eines S88-Moduls mit dem Dateneingang des nächsten verbunden ist, kommen so die beim ersten Takt geladenen Zustände der Latches nach und nach zur Zentrale.

Leider gilt dieser an sich triviale Bus als relativ störempfindlich; das liegt an der elektrisch ungenügenden Ausführung: Es werden ungeschirmte Flachbandleitungen verwendet, die dann auch noch ohne Leitungsabschluß betrieben werden. Der S88-Bus wird zudem noch recht hochohmig betrieben. Kein Wunder, dass es manchmal zu Einkopplungen und Reflexionen kommt.

- **Lösung:**

  Also sind folgende Maßnahmen zu ergreifen: **Abschluß**, räumliche **Trennung** und **geschirmte Kabel**.

  - Beim Thema Abschluß muß man die verwendetete Zentrale betrachten, was diese leisten kann. Bei OpenDCC sind TTL-Bustreiber verbaut, welche 24mA können; Hier kann man am Ende(!) der Kette mit 220Ohm gegen 5V eine deutliche Reduzierung von Clocküberschwingern erreichen.
  - Gegen Einkopplung hilft zuerst mal räumliche Trennung: die S88-Leitungen haben nichts direkt neben dem Gleissignal oder gedimmten Netzleitungen verloren. Auch sollte die Gleisverdrahtung **generell verdrillt** ausgeführt werden, damit reduziert sich das Abstrahlfeld erheblich.
  - Neben der räumlichen Trennung ist auch auf entsprechende saubere elektrische Trennung zu achten. Es dürfen keine Anlagenströme (beabsichtigt oder unbeachsichtigt) über den S88-Bus laufen. Bitte hierzu die Verdrahtungshinweise beim Booster beachten.
  - Einen noch besseren Schutz vor Einkopplung bieten geschirmte Kabel, welche sehr preiswert als Patchkabel (CAT5) für Computer zu kaufen sind. Innerhalb des CAT5-Kabels sind 4 Paare, die jeweils verdrillt sind. Benutzt man die bei S88-N verwendete Kabelbelegung, so kommt Schirmung und Verdrillung gleichermaßen zum Zug - man erreicht optimale Verhältnisse.

7.4 Das Timing am S88-Bus

- **Ausgangssituation:**

  Leider gibt es keine offizielle Spezifikation für den S88-Bus, daher versuche ich hier, anhand der bisher gebräuchlichen Chips und der Verdrahtungssituation eine sinnvolle Spezifikation zu geben. Ziel ist eine weitgehende Kompatibilität der Module zu erreichen. Leider sträuben sich manche Firmen (wie z.B. ESU) Details bekannt zu geben, man ist also auf Messungen an Einzelexemplaren angewiesen.

  – **Analyse:**
  Im Orginalinterface sind HEF4014 verbaut, diese haben (bei 5V) eine tsetup bis zu 35ns und eine thold bis 30ns. Die Durchlaufzeit tpd beträgt 260ns. (Quelle: 4014-Datenblatt von Philips[10])

  Es sind Leitungen angeschlossen, diese bringen sowohl Laufzeit als auch Kapazitätsbelag mit: übliche Kabel haben ca. t_{pd} von 6,6ns/m, was bei einer angenommenen Maximallänge von 30m in Summe 200ns ergibt. Der Kapazitätbelag wirkt zusammen mit einem Ausgangswiderstand als Tiefpass, z.B. verursachen 44pF/m bei einem Quellwiderstand von 250 Ohm eine t_{rc} von 11ns/m.

  Das Steuertiming aller bekannten Zentralen (IB, HSI88, M*) arbeitet mit extrem langsamer Bitsteuerung per Software, die jeweiligen Signalwechsel werden nach einigen 10 µs vorgenommen.

  Clockraten bisheriger Interfaces

Zentrale	t(high_min)	t(low_min)	Comment
Tams	100 µs	100 µs	wenig Jitter, Scan alle 4,9ms
Littfinski HSI88	150 µs	150 µs	wenig Jitter
Intellibox	30 µs	25 µs	starker Jitter, im Mittel etwa 70 µs
OpenDCC	20 µs	20 µs	Jitter, Timing per CV einstellbar
ECoS	25 µs	25 µs	inoffzielle, nicht kontrollierte Angabe.*1)

  **1)*
  ESU hat eine entsprechende Auskunft mit 'kopierschutzrechtlichen

Gründen' (hört, hört) verweigert - offenbar wissen sie es einfach selbst nicht genau.

Darauf haben sich auch einige Implemenierungen eingestellt - der S88 wird oft über einen Microcontroller realisiert und je nach Anbieter sind die entsprechenden Routinen unterschiedlich schnell. (Blücher scheint mind. 140 µs Zykluszeit zu fordern, railway-lauf.de mind. 200 µs).

- **Problemfälle:**
  - Mischung von verschiedenen S88-Bausteinen:
    Kombiniert man aktuelle, schnelle Bausteine mit langsamen 4014 Chips, so kann es zu Verletzungen der thold kommen.
  - Impedanzfehler:
    Die S88-Leitungen laufen *ziemlich* fehlangepasst, es kann zu Reflexionen und in Folge zu Doppeltaktungen bzw. falsch gelesenen Daten kommen.

- **Normiertes Timing:**

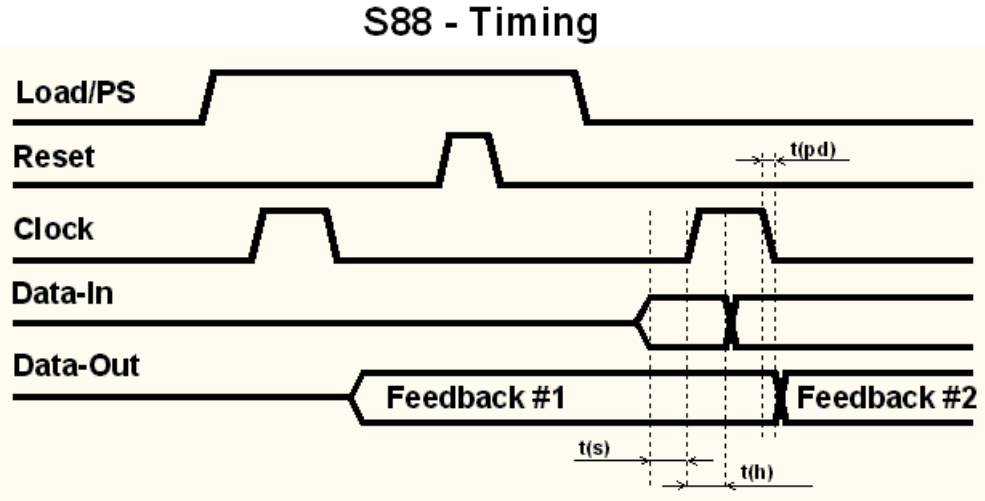


Abb. 56: Normiertes Timing (Quelle: Autor)

  - Am Eingang eines Moduls:
    t_{setup} ⟨50ns, t_{hold} ⟨50ns, jeweils bezogen auf die steigende Clockflanke; durch das langsame Timing bleibt noch Margin für eventuelle Dämpfungsglieder in der Clock und Datenleitung.
  - Am Ausgang eines Moduls:
    t_{pd} ⟩1 µs, damit das nächste Modul sicher ohne Setup/Holdverletzungen

schalten kann. Zusätzlich muß t_{pd} ⟨14 µs (bezogen auf die negative Clockflanke) gelten. Das ergibt zusammen mit der u.g. Zykluszeit für tlow ein rechtzeitiges Schalten vor dem Einlesen in die Zentrale. Damit ergibt sich ein maximaler Lesetakt von ca. 30kHz, was bei weitem reichen dürfte. Es wird empfohlen, den Modulausgang jeweils erst mit der negativen Clockflanke zu schalten.

– Clock:
Hier empfiehlt sich eine Beschränkung der Flankensteilheit, um Reflexionen zu vermeiden. t_{rise} ⟩300ns.

t_{cycle} ⟩30 µs und t_{high} ⟩15 µs und t_{low} ⟩15 µs; Für S88-N Kompatibilität wird dieses Timing gefordert. Damit sollten sich auch per Mikrocontroller implementierte Module realisieren lassen.
Ältere Module brauchen fallweise ein langsameres Timing. Die Zentrale soll hierzu eine Einstellmöglichkeit bieten.

7.5 S88-Tastatur

- **Einleitung:**

Es gibt auf der Modellbahn eine Reihe von Zubehör und bewegten Objekten, diese sollen oft sowohl automatisch als auch durch einen lokalen Taster ausgelöst werden. Dieses Zubehör wird per Decoder gesteuert, die automatisierte Auslösung übernimmt der Computer. Der manuelle Tastendruck muß nun auch über den Computer geleitet werden, hierzu habe ich einen extrem einfachen S88-Rückmeldebaustein entworfen, welcher entweder acht direkt angeschlossene Taster oder eine kleiner Zehnertastatur abfrägt, entprellt und mit einstellbarer Haltezeit am s88-Bus ausgibt.
Auch die Rückmeldung eines Boosterausfalls kann damit einfach erfolgen.

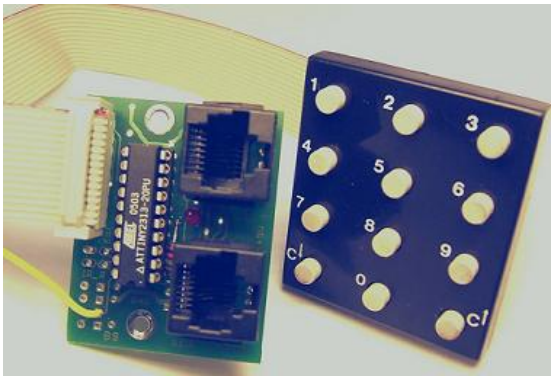


Abb. 57: Tastaturplatine (Quelle: Autor)

Die Schaltung ist als kleine Platine nur mit einem ATtiny2313 aufgebaut und damit *sehr* preiswert. Das S88-Interface ist entweder als Stiftleiste oder mit mit Netzwerkkabeln gemäß der Norm S88-N anschließbar.

- **Schaltung:**

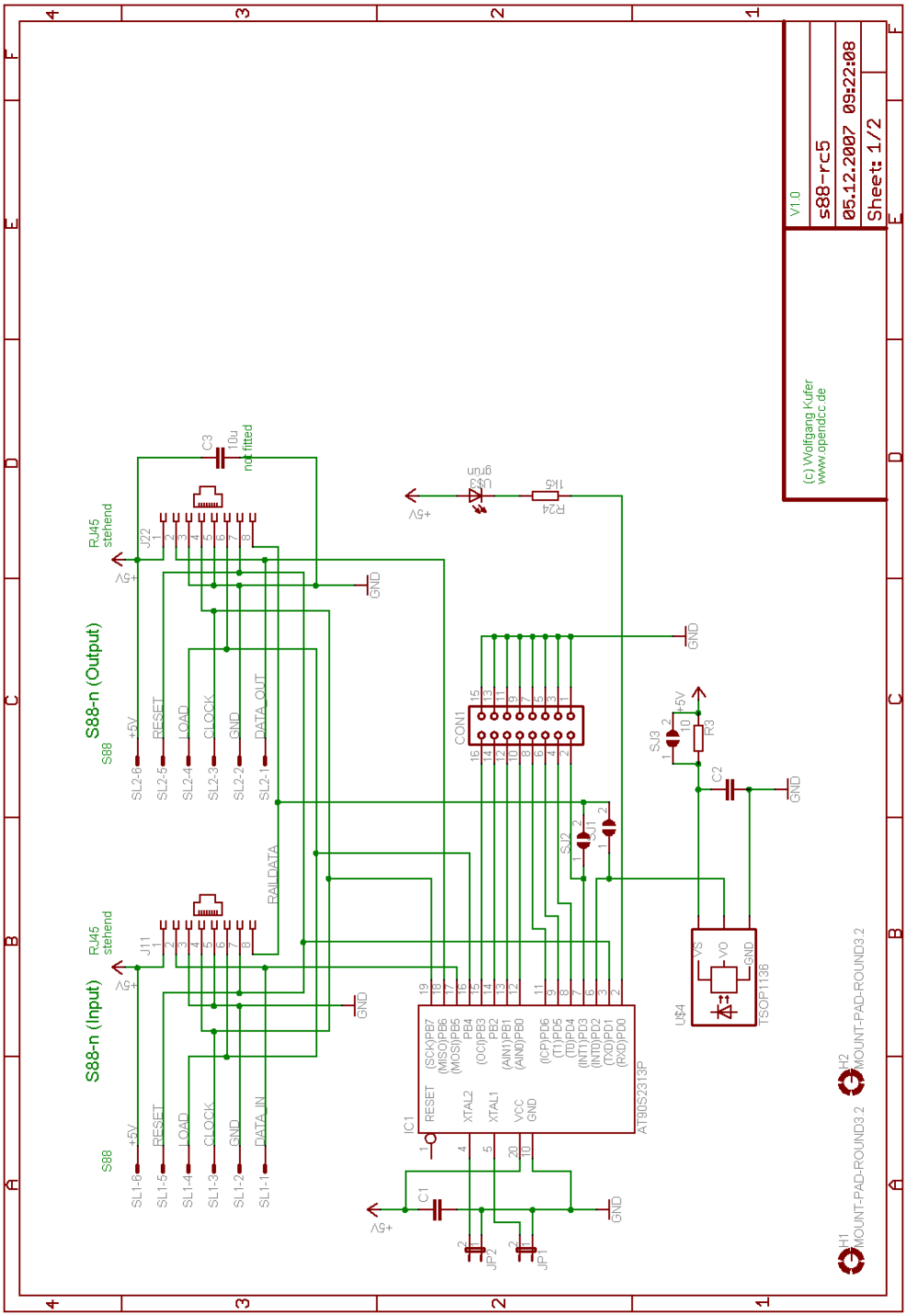


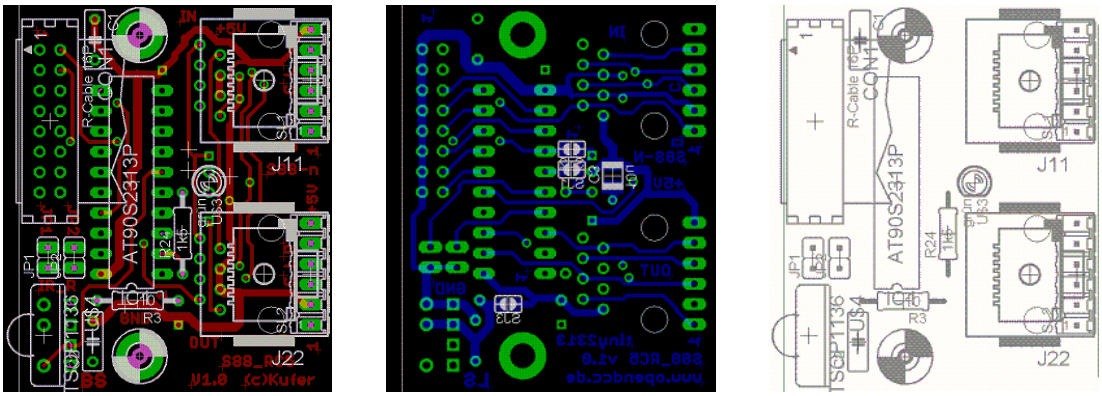
Abb. 58: Schaltung der Tastaturplatine (Quelle: Autor)

Der Schaltplan ist relativ einfach: die gesamte Logik wird mit Hilfe eines ATtiny2313 abgebildet. Dieser läuft mit einem internen Takt von 8MHz.

Der S88-Anschluß erfolgt mit Hilfe des USI (Universal Serial Interface), das ist ein Schieberegister auf dem Chip. Dieses Schieberegister wird in den S88-Datenstrom eingefügt. Der Anschluß nach außen kann in der Norm S88-N oder alternativ mit 6 Stiften erfolgen.

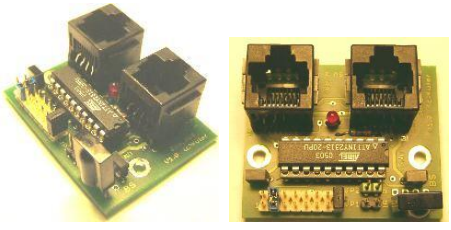
Zusätzlich ist noch ein IR-Empfänger vorgesehen, damit können Befehle von einer Fernbedienung eingelesen werden. Hierzu wird die Samplingtechnik verwendet.

Der Anschluß der Taster erfolgt mittels eines 16-poligen Flachbandkabels, das ist immer abwechselnd GND - Taste belegt. Man kann daher einfach das Kabel in Paare spleißen und so z.B. Taster auf einem Stellpult anschließen. Die Taster schalten jeweils gegen Masse, der notwendige Pullup wird innerhalb des Attiny durch Programmierung realisiert. Es ist keinerlei Schutzbeschaltung vorgesehen, man darf also nur Taster oder Optokopplerausgänge anschließen, keinesfalls irgendwelche Rückleitungen vom Gleis, diese würden den Prozessor zerstören.



(a) Bestückungsseite (b) Lötseite (c) Bauteile

Abb. 59: Tastaturplatine (Quelle: archiv)



(a) (b)

- **Anschluß:**

  Die Verbindung zum S88 Bus erfolgt einfach mit Netzwerkabeln in der Norm S88-N (5V). Wenn ein Lesetakt von S88-Bus erkannt wird, dann wird ganz kurz die LED angesteuert - diese leuchtet dadurch bei anliegendem Lesetakt schwach. Wenn eine Taste betätigt wird, so wird diese von der Software entprellt und mit einer Haltezeit von etwa 1s auf den S88-Bus ausgegeben. Solange eine Taste betätigt ist, wird die LED voll angesteuert, damit ist eine direkte Kontrolle des Tasters und der Verbindung zum Taster möglich.

  An den Baustein können entweder Standardtaster oder eine Tastatur von Pollin angeschlossen werden. Dies wird durch den JUMPER 1 unterschieden.

  - **Anschluß von Standardtastern oder Optokopplerausgängen**
    Der Baustein kann 8 Taster einlesen, diese sind jeweils gegen GND zu schalten. GND ist jeweils paarweise am Stecker angelegt, so dass recht einfach durch Spleißen des Flachbandkabels die Taster angeschlossen werden können, z.B. Taster 1 muß zwischen Ader 1 und 2 des Kabels angeklemmt werden. Auch ein Einzelanschluß mit Stiften (wie auf PC-Motherboards üblich) ist möglich.

    Der Baustein belegt in diesem Fall 8 Rückmelderadressen, Jumper 1 ist nicht gesteckt.

    Anschluß Taster

GND	1	2	Key 1	S88 Melder 1
GND	3	4	Key 2	S88 Melder 2
GND	5	6	Key 3	S88 Melder 3
GND	7	8	Key 4	S88 Melder 4
GND	9	10	Key 5	S88 Melder 5
GND	11	12	Key 6	S88 Melder 6
GND	13	14	Key 7	S88 Melder 7
GND	15	16	Key 8	S88 Melder 8

  - **Anschluß einer Minitastatur**
    Der Baustein kann eine kleine Zehnertastur einlesen - diese sind oft recht preiswert zu erstehen.
    Die Tastatur hat 12 Tasten, diese sind in 2 Spalten zu 5 bzw. 7 Zeilen organisiert. Es sind also 9 Leitungen erforderlich: 8 Leitungen werden von Standardstecker verwenden, zusätzlich wird die Anschlußleitung des IR-Empfängers verwendet (dieser darf dann natürlich nicht bestückt

sein).
Zur Ansteuerung dieser Tasten werden die Abfragepulse von Prozessor erzeugt.

Der Baustein belegt in diesem Fall 16 Rückmelderadressen, davon sind allerdings nur die untersten 12 Adressen belegt, wegen der leichteren Zuordnung in der PC-Software ist die Länge des Bausteins aber auf 16 eingestellt. Jumper 1 ist gesteckt.

Anschluß Zehnertastatur Samwell

Verbindung IR-Receiver			Pin 1
n.c.	1	2	Pin 2
n.c.	3	4	Pin 6
n.c.	5	6	Pin 7
n.c.	7	8	Pin 8
n.c.	9	10	Pin 9
n.c.	11	12	Pin 10
n.c.	13	14	Pin 11
n.c.	15	16	Pin 12

Die Tasten sind dann wie folgt den S88-Meldeadressen zugeordnet:

Tastenbelegung Zehnertastatur Samwell

Taste	1	2	3	4	5	6	7	9	0	Cv	C^				
S88-Melder	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16

- **Erläuterungen zur Software:**
  - **Tasten einlesen:**
    Alle Zustände von Ports, an denen Taster angeschlossen werden können, werden von der Einleseroutine in einen Vektor geschrieben. Dieser Vektor wird dann mit Hilfe einer Statusmaschine bitweise bearbeitet. Damit wird Entprellen in beide Schaltrichtungen und zusätzlich ein Memory von einer Sekunde für den aktivierten Zustand realisiert. Solange der Ausgangsvektor (current_message) der Statusmaschine gedrückte Tasten anzeigt, wird die lokale LED zur Kontrolle angesteuert.
  - **Übergabe an den S88 Bus:**
    Der S88-Bus wird über die USI-Schnittstelle des Prozessors verwaltet. Diese Schnittstelle bietet 8 Bits, welche mit einem externen Takt

geschoben werden. Am Ende des Schiebens erfolgt ein Interrupt (USI_OVERFLOW).
Damit die Bitlänge des Moduls frei programmierbar ist, wird jedoch die Schnittstelle so programmiert, das bei jeder Clockflanke ein Interrupt erfolgt. Bei steigender Flanke wird das eingetaktete Bit wird in ein Schieberegister (shift_reg) innerhalb der SW übernommen, bei fallender Flanke wird das nächste ausgebende Bit in den Ausgangsbuffer gelegt. Die Länge des Softwareregisters wird mittels der Variable shift_size festgelegt. Damit kann diese Routine gleichermaßen für Rückmeldelängen von 4 bis 32 Bit verwendet werden.
Bei jedem Interrupt erfolgt ein kurzes Ansteuern der LED, damit läßt sich der Ausleseprozess kontrollieren.

- **Unterlagen:**

  Download: S88_Keyboard_V0.02.zip

8 GNU-Lizenz

8.1 Deutsch - nicht rechtsverbindlich

Deutsche Übersetzung[23] der GNU General Public License
Erstellt im Auftrag der SuSE GmbH [suse@suse.de]
von Katja Lachmann Übersetzungen [na194@fim.uni-erlangen.de], überarbeitet von Peter Gerwinski [peter.gerwinski@uni-essen.de] (31. Oktober 1996)

Diese Übersetzung wird mit der Absicht angeboten, das Verständnis der GNU General Public License (GNU-GPL) zu erleichtern. Es handelt sich jedoch nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.

Die Free Software Foundation (FSF) ist nicht der Herausgeber dieser Übersetzung, und sie hat diese Übersetzung auch nicht als rechtskräftigen Ersatz für die Original-GNU-GPL anerkannt. Da die Übersetzung nicht sorgfältig von Anwälten überprüft wurde, können die Übersetzer nicht garantieren, dass die Übersetzung die rechtlichen Aussagen der GNU-GPL exakt wiedergibt. Wenn Sie sichergehen wollen, dass von Ihnen geplante Aktivitäten im Sinne der GNU-GPL gestattet sind, halten Sie sich bitte an die englischsprachige Originalversion.

Die Free Software Foundation möchte Sie darum bitten, diese Übersetzung nicht als offizielle Lizenzbedingungen für von Ihnen geschriebene Programme zu verwenden. Bitte benutzen Sie hierfür stattdessen die von der Free Software Foundation herausgegebene englischsprachige Originalversion.

This is a translation of the GNU General Public License into German. This translation is distributed in the hope that it will facilitate understanding, but it is not an official or legally approved translation.

The Free Software Foundation is not the publisher of this translation and has not approved it as a legal substitute for the authentic GNU General Public License. The translation has not been reviewed carefully by lawyers, and therefore the translator cannot be sure that it exactly represents the legal meaning of the GNU General Public License. If you wish to be sure whether your planned activities are permitted by the GNU General Public License, please refer to the authentic English version.

The Free Software Foundation strongly urges you not to use this translation as the official distribution terms for your programs; instead, please use the authentic English version published by the Free Software Foundation.

GNU General Public License (engl. Original)
GNU General Public License
Deutsche Übersetzung der Version 2, Juni 1991

Jeder hat das Recht, diese Lizenzurkunde zu vervielfältigen und unveränderte Kopien zu verbreiten; Änderungen sind jedoch nicht gestattet.

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion! Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die GNU General Public License, die allgemeine öffentliche GNU-Lizenz, ebendiese Freiheit garantieren. Sie soll sicherstellen, dass die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der Free Software Foundation herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Werk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden. (Ein anderer Teil der Software der Free Software Foundation unterliegt stattdessen der GNU Library General Public License, der allgemeinen öffentlichen GNU-Lizenz für Bibliotheken.)

Die Bezeichnung "freie" Software bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, dass Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen - und dass Sie wissen, dass Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie - kostenlos oder gegen Bezahlung - Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, dass auch sie den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüberhinaus sicherstellen, dass jeder erfährt, daß für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, daß die Empfänger wissen, daß sie nicht das Original erhalten haben, damit von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-Patente bedroht. Wir möchten die Gefahr ausschließen, dass Distributoren eines

freien Programms individuell Patente lizensieren - mit dem Ergebnis, dass das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, dass jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muß oder überhaupt nicht lizenziert werden darf.

Es folgen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung: Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung

Paragraph 0
Diese Lizenz gilt für jedes Programm und jedes andere Werk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, dass das Werk unter den Bestimmungen dieser General Public License verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Werk als "das Programm" bezeichnet; die Formulierung "auf dem Programm basierendes Werk" bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Werk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als "Bearbeitung" eingestuft.) Jeder Lizenznehmer wird im folgenden als "Sie" angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Werk darstellt (unabhängig davon, dass die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

Paragraph 1
Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, dass Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluß veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und desweiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie für das Programm anbieten.

Paragraph 2
Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Werk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, dass zusätzlich alle folgenden Bedingungen erfüllt werden:

(a)

Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung

hinweist.

(b)

Sie müssen dafür sorgen, dass jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

(c)

Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, dass es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, dass es keine Gewährleistung gibt (oder anderenfalls, daß Sie Garantie leisten), und daß die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muß der Benutzer darauf hingewiesen werden, wie er eine Kopie dieser Lizenz ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muß Ihr auf dem Programm basierendes Werk auch keine solche Meldung ausgeben).

Diese Anforderungen betreffen das veränderte Werk als Ganzes. Wenn identifizierbare Abschnitte des Werkes nicht von dem Programm abgeleitet sind und vernünftigerweise selbst als unabhängige und eigenständige Werke betrachtet werden können, dann erstrecken sich diese Lizenz und ihre Bedingungen nicht auf diese Abschnitte, wenn sie als eigenständige Werke verbreitet werden. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen verbreiten, das ein auf dem Programm basierendes Werk darstellt, dann muß die Verbreitung des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf die Gesamtheit ausgedehnt werden - und damit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Werke in Anspruch zu nehmen oder zu beschneiden, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Werken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt ein einfaches Zusammenstellen eines anderen Werkes, das nicht auf dem Programm basiert, zusammen mit dem Programm oder einem auf dem Programm basierenden Werk auf ein- und demselben Speicher- oder Vertriebsmedium das andere Werk nicht in den Anwendungsbereich dieser Lizenz.

Paragraph 3
Sie dürfen das Programm (oder ein darauf basierendes Werk gemäß Paragraph 2) als Objectcode oder in ausführbarer Form unter den Bedingungen von Paragraph 1 und 2 vervielfältigen und verbreiten - vorausgesetzt, dass Sie außerdem eine der folgenden Leistungen erbringen:

(a)

Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muß. Oder:

(b)

Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen - zu nicht höheren Kosten als denen, die durch den physikalischen Kopiervorgang anfallen -, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:

(c)

Liefern Sie das Programm zusammen mit dem schriftlichen Angebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

Unter dem Quelltext eines Werkes wird diejenige Form des Werkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet "der komplette Quelltext": Der Quelltext aller im Programm enthaltenen Module einschließlich aller zugehörigen Modulschnittstellen-Definitionsdateien sowie der zur Compilation und Installation verwendeten Skripte. Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft - es sei denn, diese Komponente selbst gehört zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programms oder des Objectcodes dadurch erfolgt, dass der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quelltext als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objectcode zu kopieren.

Paragraph 4
Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

Paragraph 5
Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterze-

ichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Werke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Werk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Werkes.

Paragraph 6
Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Werk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

Paragraph 7
Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluß, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluß.

Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

Paragraph 8
Wenn die Verbreitung und/oder die Benutzung des Programms in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnitt-

stellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geographische Begrenzung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, so dass die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

Paragraph 9
Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, dass es dieser Lizenz in einer bestimmten Versionsnummer oder "jeder späteren Version" ("any later version") unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.
Paragraph 10
Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software Foundation steht, schreiben Sie an die Free Software Foundation; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Werke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern. Keine Gewährleistung

Paragraph 11
Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, "wie es ist", ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich - aber nicht begrenzt auf - Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

Paragraph 12
In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich - aber nicht beschränkt auf - Datenverluste, fehlerhafte Ve-

rarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.
Ende der Bedingungen
Anhang: Wie Sie diese Bedingungen auf Ihre neuen Programme anwendbar machen

Wenn Sie ein neues Programm entwickeln und wollen, dass es von größtmöglichem Nutzen für die Allgemeinheit ist, dann erreichen Sie das am besten, indem Sie es zu freier Software machen, die jeder unter diesen Bestimmungen weiterverbreiten und verändern kann.

Um dies zu erreichen, fügen Sie die folgenden Anmerkungen zu Ihrem Programm hinzu. Am sichersten ist es, sie an den Anfang einer jeden Quelldatei zu stellen, um den Gewährleistungsausschluß möglichst deutlich darzustellen; außerdem sollte jede Datei mindestens eine "Copyright"-Zeile besitzen sowie einen kurzen Hinweis darauf, wo die vollständige Lizenz gefunden werden kann.

```
[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
Copyright (C) 19[yy]  [Name des Autors]

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 2 of
the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Auf Deutsch:

```
[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
Copyright (C) 19[jj]  [Name des Autors]

Dieses Programm ist freie Software. Sie können es unter
den Bedingungen der GNU General Public License, wie von der
Free Software Foundation herausgegeben, weitergeben und/oder
modifizieren, entweder unter Version 2 der Lizenz oder (wenn
Sie es wünschen) jeder späteren Version.

Die Veröffentlichung dieses Programms erfolgt in der
Hoffnung, dass es Ihnen von Nutzen sein wird, aber OHNE JEDE
GEWÄHRLEISTUNG - sogar ohne die implizite Gewährleistung
```

```
der MARKTREIFE oder der EIGNUNG FÜR EINEN BESTIMMTEN ZWECK.
Details finden Sie in der GNU General Public License.

Sie sollten eine Kopie der GNU General Public License zusammen
mit diesem Programm erhalten haben. Falls nicht, schreiben Sie
an die Free Software Foundation, Inc., 675 Mass Ave, Cambridge,
MA 02139, USA.
```

Fügen Sie auch einen kurzen Hinweis hinzu, wie Sie elektronisch und per Brief erreichbar sind.
Wenn Ihr Programm interaktiv ist, sorgen Sie dafür, dass es nach dem Start einen kurzen Vermerk ausgibt:

```
Gnomovision version 69, Copyright (C) 19[yy]  [Name des Autors]
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type
‘show w’. This is free software, and you are welcome to redistribute
it under certain conditions; type ‘show c’ for details.
```

Auf Deutsch:

```
Gnomovision Version 69, Copyright (C) 19[jj]  [Name des Autors]
Für Gnomovision besteht KEINERLEI GARANTIE; geben Sie ‘show w’
für Details ein. Gnomovision ist freie Software, die Sie unter
bestimmten Bedingungen weitergeben dürfen; geben Sie ‘show c’
für Details ein.
```

Die hypothetischen Kommandos ‘show w’ und ‘show c’ sollten die entsprechenden Teile der GNU-GPL anzeigen. Natürlich können die von Ihnen verwendeten Kommandos anders heißen als ‘show w’ und ‘show c’; es könnten auch Mausklicks oder Menüpunkte sein - was immer am besten in Ihr Programm paßt. Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber (wenn Sie als Programmierer arbeiten) oder Ihre Schule einen Copyright-Verzicht für das Programm unterschreiben lassen. Hier ein Beispiel; ändern Sie bitte die Namen:

```
Yoyodyne, Inc., hereby disclaims all copyright interest
in the program "Gnomovision" (which makes passes at compilers)
written by James Hacker.

[Unterschrift von Ty Coon], 1 April 1989
Ty Coon, President of Vice
```

Auf Deutsch:

```
Die Yoyodyne GmbH erhebt keinerlei urheberrechtlichen
Anspruch auf das Programm "Gnomovision" (einem Schrittmacher
für Compiler), geschrieben von James Hacker.

[Unterschrift von Ty Coon], 1. April 1989
Ty Coon, Vizepräsident
```

Diese General Public License gestattet nicht die Einbindung des Programms in proprietäre Programme. Ist Ihr Programm eine Funktionsbibliothek, so kann es sinnvoller sein, das Linken proprietärer Programme mit dieser Bibliothek zu gestatten. Wenn Sie dies tun wollen, sollten Sie die GNU Library General Public License anstelle dieser Lizenz verwenden.

8.2 English

http://www.gnu.org/licenses/gpl.html

9 Literaturverzeichnis

[1] AVR-GCC C Compiler Version 3.2 20020612. *http://www.avrfreaks.net, Atmel Norway.* 20.03.2007 gefunden: 07.01.2008.

[2] WinAVR Version 20070122. *http://winavr.sourceforge.net/.* gefunden: 07.01.2008.

[3] Ponyprog V 2.07c BETA. *Claudio Lanconelli, http://www.lancos.com/ppwin95.html.* 48022, Lugo. gefunden: 07.01.2008.

[4] MProg 3.0a. *EEPROM Programming Utility, FTDI, http://www.ftdichip.com/Resources/Utilities.htm#MProg.* 2007 gefunden: 12.01.2008.

[5] AVRStudio Version 4.13. *Atmel Corporation, 2325 Orchard Parkway, San Jose, CA 95131 USA.* gefunden: 06.01.2008.

[6] P.f.u.Sch. Version 5.11. *Dipl.Ing. Ewald Sperrer, http://www.stp-software.at.* Weissenberg 23, 4053 Haid. gefunden: 07.01.2008.

[7] Traincontroller V 5.8. *Jürgen Freiwald, http://www.freiwald.com/seiten/traincontroller.htm.* Kreuzberg 16 B, D-85658 Egmating. gefunden: 07.01.2008.

[8] TrainProgrammer V 5.8. *Jürgen Freiwald, http://www.freiwald.com/seiten/traincontroller.htm.* Kreuzberg 16 B, D-85658 Egmating. gefunden: 12.01.2008.

[9] Elektronikentwicklung Blücher. *http://www.bluecher-elektronik.de, Barstraße 23, Berlin, D - 10713.* gefunden: 26.09.2008.

[10] NXP Semiconductors Netherlands B.V. *http://www.nxp.com/, High Tech Campus 60, AG Eindhoven, The Netherlands 5656.* gefunden: 25.09.2008.

[11] Atmel Corporation. *http://www.atmel.com/, 2325 Orchard Parkway, San Jose, CA 95131 USA.* gefunden: 06.01.2008.

[12] Reichelt Elektronik e. Kfr. *Online-Shop, http://www.reichelt.de/.* gefunden: 13.01.2008.

[13] TrackONE V 1.1.0 (Beta) Der Gleisbildeditor für Windows. *Michael Reukauff, http://www.reukauff.de/TrackONE/.* 30.12.2002 gefunden: 07.01.2008.

[14] FTDI. *Future Technology Devices International Limited, http://www.ftdichip.com/.* 2007 gefunden: 12.01.2008.

[15] Lenz Elektronik GmbH. *http://www.lenz-elektronik.de/, Hüttenbergstrasse 29, Giessen, D 35398.* gefunden: 23.09.2008.

[16] Roco Modelleisenbahn GmbH. *http://www.roco.co.at/, Plainbachstraße 4, Bergheim, A-5101.* gefunden: 23.09.2008.

[17] Uhlenbrock Elektronik GmbH. *http://www.uhlenbrock.de/, Mercatorstrasse 6, Bottrop, D 46244.* gefunden: 23.09.2008.

[18] SpDrS60 grafischer SRCP-Client. *Guido Scholz, http://spdrs60.sourceforge.net/.* 02.08.2007 gefunden: 07.01.2008.

[19] Intellibox. *Multi Protokoll Digitalsystem, Uhlenbrock Elektronik GmbH.* 2002 gefunden: 12.01.2008.

[20] Ponyprog ISP. *Claudio Lanconelli, http://www.lancos.com/prog.html#avrisp.* 48022, Lugo. gefunden: 12.01.2008.

[21] Wolfgang Kufer. *OpenDCC - Eine Zentrale für DCC, http://www.opendcc.de/index.html*, 12.11. 2006. gefunden: 06.01.2008.

[22] Littfinski DatenTechnik (LDT). *Dipl. Ing. Peter Littfinski, http://www.ldt-infocenter.com/.* gefunden: 07.01.2008.

[23] GNU General Public License. *GNU Deutsche Übersetzung, S.u.S.E. GmbH, http://oswaldism.de/gpl-ger.html.* 1996 gefunden: 12.01.2008.

[24] NRMA. *National Model Railroad Association, Inc., http://www.nmra.org/.* 1995-2007 gefunden: 07.01.2008.

[25] Bürklin OHG. *Online-Shop, http://www.buerklin.com/.* 1998 - 2007 gefunden: 13.01.2008.

[26] SRCP: Model Railroads and Internet. *http://srcpd.sourceforge.net/.* 28 Oct 2007 gefunden: 07.01.2008.

[27] Railware. *Andrea Hinz, http://www.railware.com/.* Dieffler Straße 18a, 66701 Beckingen. gefunden: 07.01.2008.

[28] Rocrail Model Railroad Control System. *http://www.rocrail.net/.* 03.01.2008 gefunden: 07.01.2008.

[29] VCP-Treiber. *Virtual COM port (VCP), FTDI, http://www.ftdichip.com/Drivers/VCP.htm.* 2007 gefunden: 12.01.2008.

[30] NEM 609 Richtlinien zur elektrischen Sicherheit bei Modellbahnausstellungen. *Verband der Modelleisenbahner und Eisenbahnfreunde Europas (MOROP), Bern (CH) - Fondée.* gefunden: 06.01.2008.

Stichwortverzeichnis

mailto:post@walther-andreas.de