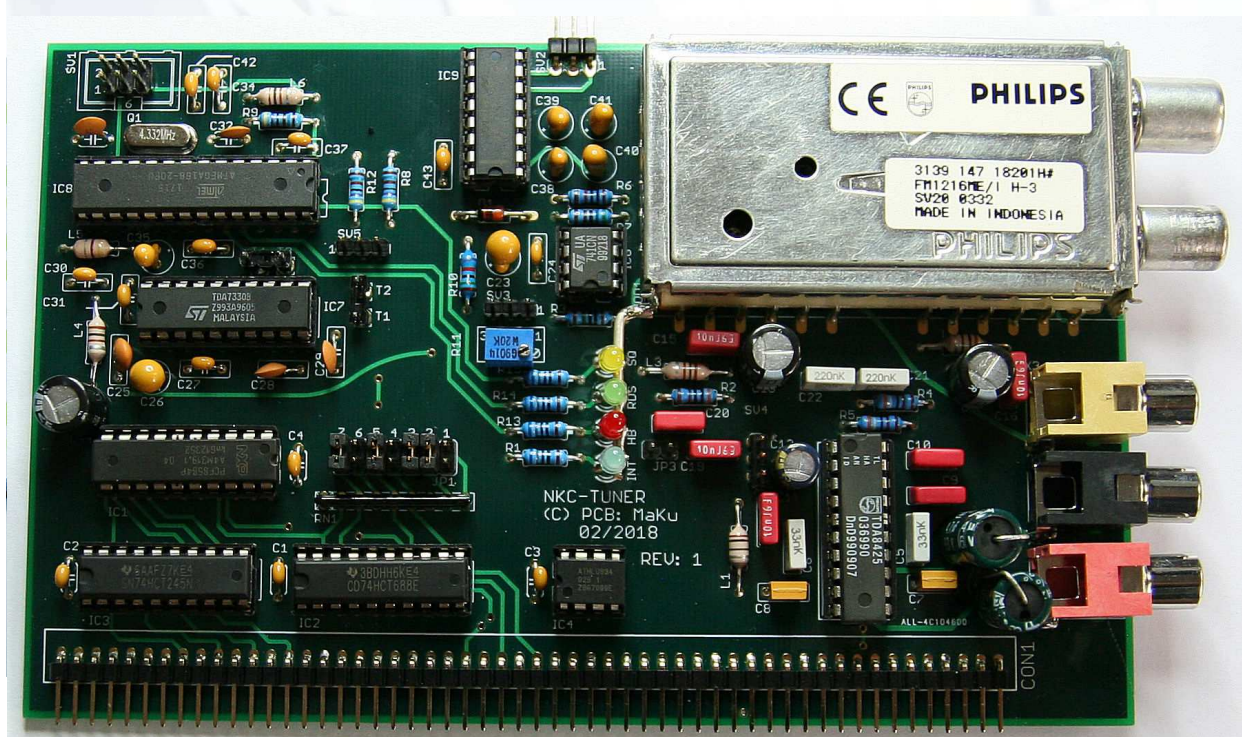


Tuner-Karte für den NDR-Klein-Computer

Spezifikation

Tuner-Karte für den NDR-Klein-Computer



Version 1.5

Idee:

Sascha Neuschl
 Pirolweg 21
 48167 Münster
 Email: scn69@gmx.de

Industrielle Ausführung, Schaltungsverbesserung und Konzepterweiterung

Thomas Heller, Matthias Kübeck
 Email: info@helkueb.de

Dokumentenhistorie

Version	Autor(en)	Änderung	Datum
1.0	Neuschl, Sascha	Erste Version – ohne RDS-Dekoder	01.07.2013
1.1	Neuschl, Sascha	Formale Korrekturen	20.01.2014
1.2	Neuschl, Sascha	Version – mit RDS-Dekoder	01.05.2014
1.3	Neuschl, Sascha	Bedienprogramm erstellt	05.08.2014
1.4	Neuschl, Sascha	Bereitstellung der Tuner-Regelspannung (AGC) für UKW-Signalstärkemessung, Integration der SER2-Karte in das Bedienprogramm sowie Anzeige der Signalstärke mit Hilfe der AD10X1-Karte	12.02.2018
1.5	Neuschl, Sascha	Anpassung der Dokumentation an die neue industrielle Platine	29.04.2018

Inhaltsverzeichnis

1	Vorwort.....	5
1.1	Idee.....	5
1.2	Ansatz.....	5
1.3	Erweiterungen.....	5
1.4	Aktueller Stand	5
2	Beschreibung des Konzepts	5
3	Schaltungsprinzip.....	7
3.1	Adresslogik	7
3.2	Datenpuffer	7
3.3	I2C-Prozessor.....	7
3.4	I2C-Bus.....	8
3.4.1	I2C-Bus-Adressierung	8
4	Schaltplan	9
5	Anmerkungen.....	12
5.1	Spannungsversorgung:	12
5.2	Eingänge und Ausgänge:	12
6	Test und Programmierung der Karte	14
6.1	Programmierung mit Schichtenkonzept:	14
6.2	Schicht 1 - Beispielcode:	15
6.3	Schicht 2 - Beispielcode:	17
6.4	Schicht 3 - Beispielcode:	21
7	RDS-Dekoder.....	25
7.1	Ansatz.....	25
7.2	Aktueller Stand	25
7.3	Schaltungsprinzip:	26
7.4	Funktionen des RDS-Dekoder-Programms:.....	28
7.5	Kommunikation zwischen RDS-Dekoder und NKC.....	30
7.6	Test des RDS-Dekoders.....	30
7.7	Testpunkte auf der Platine.....	30
7.8	Schaltplan	31
7.9	Programmierung des Microcontrollers ATMEGA 168-P für den RDS-Dekoder.....	32
7.9.1	Programmkopf und Verweis auf ATMEL Studio Projekt	32
8	S-Meter - Messung der Signalstärke	33
8.1	Ansatz.....	33

Tuner-Karte für den NDR-Klein-Computer

8.2	Schaltungsprinzip:	33
8.3	Schaltplan - Ausschnitt:	34
8.4	Kalibrierung:.....	35
8.4.1	Ausgangsspannung des Operationsverstärkers:	35
8.4.2	Anpassung des Bedienprogramms:	35
9	Bedienprogramm	37
9.1	Parameter und Systemkonfiguration	37
9.2	Funktionen	37
9.3	Masken	38
10	Stückliste.....	39
10.1	Tunerkarte	39
10.1.1	Platine	39
10.1.2	Bauteile:	40
11	Anhang.....	42
11.1	Datenblätter TTL-Bausteine:	42
11.1.1	74LS688	42
11.1.2	74LS245	42
11.2	Datenblatt RS232-Pegelwandler:	43
11.2.1	MAX232	43
11.3	Datenblatt Operationsverstärker:.....	44
11.3.1	µA 741-P.....	44
11.4	Verweis auf Datenblätter komplexer Bausteine und Spezifikationen.....	45

1 Vorwort

1.1 Idee

Für den NDR-Klein-Computer (NKC) wurden schon viele Platinen von Enthusiasten - lange nach der eigentlichen Ära des NKCs - entwickelt. Dabei ging es aber oft um Speicher- oder allgemeine I/O-Projekte.

Dies nahm ich zum Anlass, mal einen Exoten zu entwickeln - nämlich den Einstieg des NKCs in die mediale Welt mit einer Tunerkarte ...

1.2 Ansatz

Es sollte eine Tunerkarte entwickelt werden, die sich mit vertretbarem Hardwareaufwand aufbauen und auch recht einfach in Assembler programmieren ließ, ohne dabei der Oberexperte zu sein.

Deshalb fiel die Wahl auf ein Konzept, das den I2C-Bus von Philips nutzt, der für verschiedenste Bausteine z.B. innerhalb eines Fernsehgerätes für die Programmwahl und diverse Einstellungen wie Helligkeit, Kontrast, Lautstärke und Klang genutzt wird.

1.3 Erweiterungen

In Zukunft sollte die Tunerkarte noch einen RDS-Dekoder für das Radio bekommen. Dies war zunächst eine Standalone-Karte mit einem ATMEGA168, die Huckepack auf die Tunerkarte gesetzt wurde. Die dekodierten RDS-Zeichen sollen von dem ATMEGA168 per serielle Schnittstelle an den NKC übergeben werden. Des Weiteren sollte die Regelspannung des Tuners (AGC) zur Anzeige der Signalstärke eines Senders mit Hilfe einer AD-Wandler-Karte verwendet werden.

1.4 Aktueller Stand

Thomas Heller und Matthias Kübeck haben nun eine industrielle Platine mit den entsprechenden Dateien zum PCB-Layout erstellt. Es gibt Prototypplatinen, die fehlerfrei funktionieren.

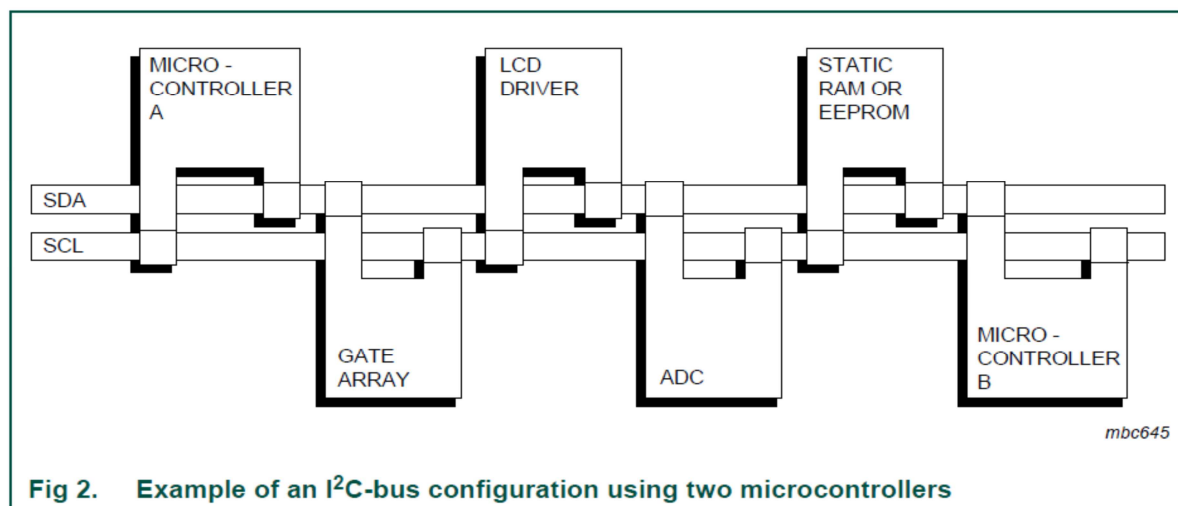
Die Platine enthält die Logik für den I2C-Bus, den RDS-Dekoder und die Aufbereitung des Tuner-AGC-Signals zur Auswertung der Signalstärke eines Senders.

2 Beschreibung des Konzepts

Das Konzept der Tuner-Karte besteht im Wesentlichen darin, ein Interface des NKCs zum Philips I2C-Bus bereitzustellen. Der I2C-Bus ist ein serieller Bus mit einer Taktleitung (SCL) und einer Datenleitung (SDA). Genauer ist dem Dokument „UM10204.pdf - I2C-bus specification and user manual“ zu entnehmen.

Es gibt Master- und Slave-Geräte auf diesem Bus. Ein Master-Gerät steuert den Datenfluss beim Lesen und Schreiben gegenüber einem Slave-Gerät. Ein Master ist i.d.R. ein Microcontroller. Grundsätzlich lässt der I2C-Bus mehrere Master-Geräte und natürlich mehrere Slave-Geräte zu.

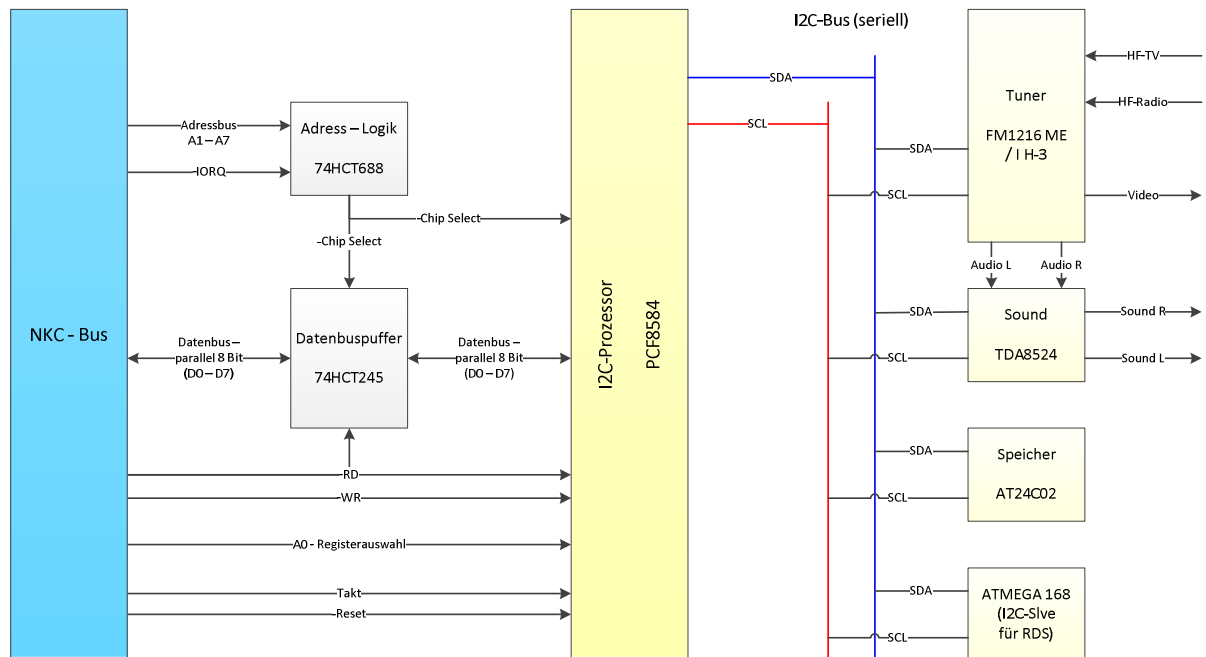
Tuner-Karte für den NDR-Klein-Computer



Im vorliegenden Fall der NKC-Tuner-Karte haben wir folgende Konstellation für den I2C-Bus:

Gerät	Funktion	Baustein	Beschreibung	Datenblatt
Master	I2C-Prozessor	PCF 8584-P	Der Baustein stellt das Interface zwischen dem parallelen NKC-Bus und dem seriellen I2C-Bus dar	PCF8584.pdf
Slave	Tuner	Philips FM1216 ME / I H-3	I2C-programmierbarer Tunerbaustein für Radio und analoges Fernsehen	FM1216ME_MK3.pdf
Slave	Sound	TDA 8425	Lautstärke- und Klangeinstellung mit zwei Eingangskanälen (Tunerkarte liegt auf erstem Kanal!)	TDA8425.pdf
Slave	Speicher – 2 KByte	AT24C02	EEPROM zum residenten Speichern der aktuellen Settings des Tuners und des Soundbausteins für Neustart sowie der Programme (Frequenzen) für Radio (und Fernsehen)	AT24C02.pdf
Slave	RDS-Erfang / Verarbeitung	ATMEGA 168-P	Microcontroller zur Aufbereitung des RDS-Datenstroms sowie der Signalstärke und Übermittlung an den NKC	ATMEGA168.pdf

3 Schaltungsprinzip



3.1 Adresslogik

Die Adresslogik wird mit dem Baustein **74HCT688** abgebildet. Er benötigt zur Freigabe das **IO-Request-Signal (IORQ)** des NKC-Busses. Des Weiteren werden die **Datenleitungen A1 bis A7** zugeführt und mit den Werten, die am **JP1** (siehe Schaltplan) eingestellt werden, verglichen. **Am JP1 wird somit die IO-Adresse des I2C-Prozessors PCF 8584-P eingestellt.** Bemerkung: Da der 74LS688 8 Zustände vergleicht, wir aber nur 7 benötigen, wurden die Eingänge P0 und Q0 (siehe Schaltbild 74HCT688) auf 0V gelegt, damit diese immer gleich sind. Die **Datenleitung A0** ist direkt an **den I2C-Prozessor PCF 8584-P** geführt, um dort interne Register anzusprechen. Sind die angelegten Adressdaten des **JP1** und der **Datenleitungen A1 bis A7** identisch, dann wird das Signal **-Chip Select** ausgelöst, was einerseits den **Datenpuffer 74HCT245** und den **I2C-Prozessor PCF 8584-P** frei gibt.

3.2 Datenpuffer

Als **Datenpuffer** wird ein **74HCT245** eingesetzt. Seine Freigabe erfolgt durch das Signal **-Chip Select**. Die Richtung, also ob Daten zum **I2C-Prozessor PCF8584** gesendet oder von ihm geholt werden, bestimmt das Signal **-RD** vom NKC-Bus. Die zu sendenden oder zu holenden Daten werden über die Datenleitungen **D0 bis D7** vom **NKC-Bus** übertragen.

3.3 I2C-Prozessor

Der **I2C-Prozessor PCF 8584-P** wird durch das **Signal -Chip Select** freigegeben. Mit der Adressleitung **A0 vom NKC-Bus** wird entschieden, ob das **Datenregister** (Zustand = logisch 1) oder das **Status-register** (Zustand = logisch 0) angesprochen wird. Daten, um aus dem **I2C-Prozessor PCF 8584-P** zu lesen oder auf ihn zu schreiben, gelangen über die durch den **74HCT245** gepufferten Datenleitungen **D0 bis D7** zu ihm. Der Zugriff des Lesens wird über das Signal **-RD** und der des Schreibens über das Signal **-WR** des NKC-Busses ausgelöst. Für eine korrekte Funktion benötigt der **I2C-Prozessor PCF 8584-P** noch den **CPU-Takt** vom NKC-Bus. Zudem wurde das **-Reset-Signal** des NKC-Busses an den Prozessor gelegt, um ihn bei einem System-Reset zurückzusetzen.

3.4 I2C-Bus

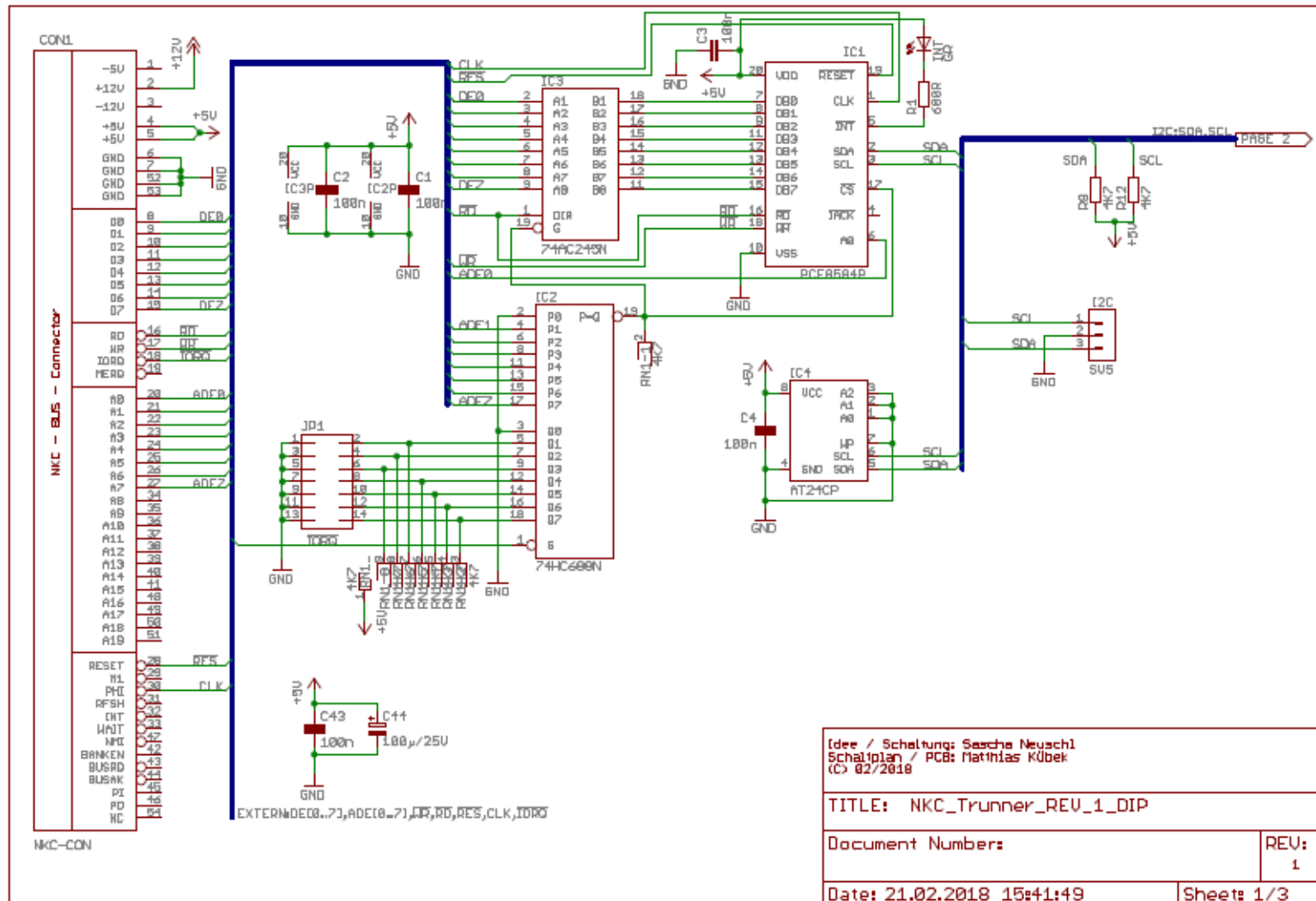
Der **I2C-Prozessor PCF 8584-P** übersetzt die **parallelen 8-Bit-IO-Daten** in die **seriellen Daten des I2C-Busses** um. Dieser Bus besitzt eine **Taktleitung (SCL)** und eine **Datenleitung (SDA)**. An diesen beiden Leitungen hängen alle Geräte des **I2C-Busses**. In unserem Fall (neben dem I2C-Prozessor) der Tuner, der Soundbaustein, der Speicher - das EEPROM und der ATMEGA 168-P (siehe Tabelle oben).

3.4.1 I2C-Bus-Adressierung

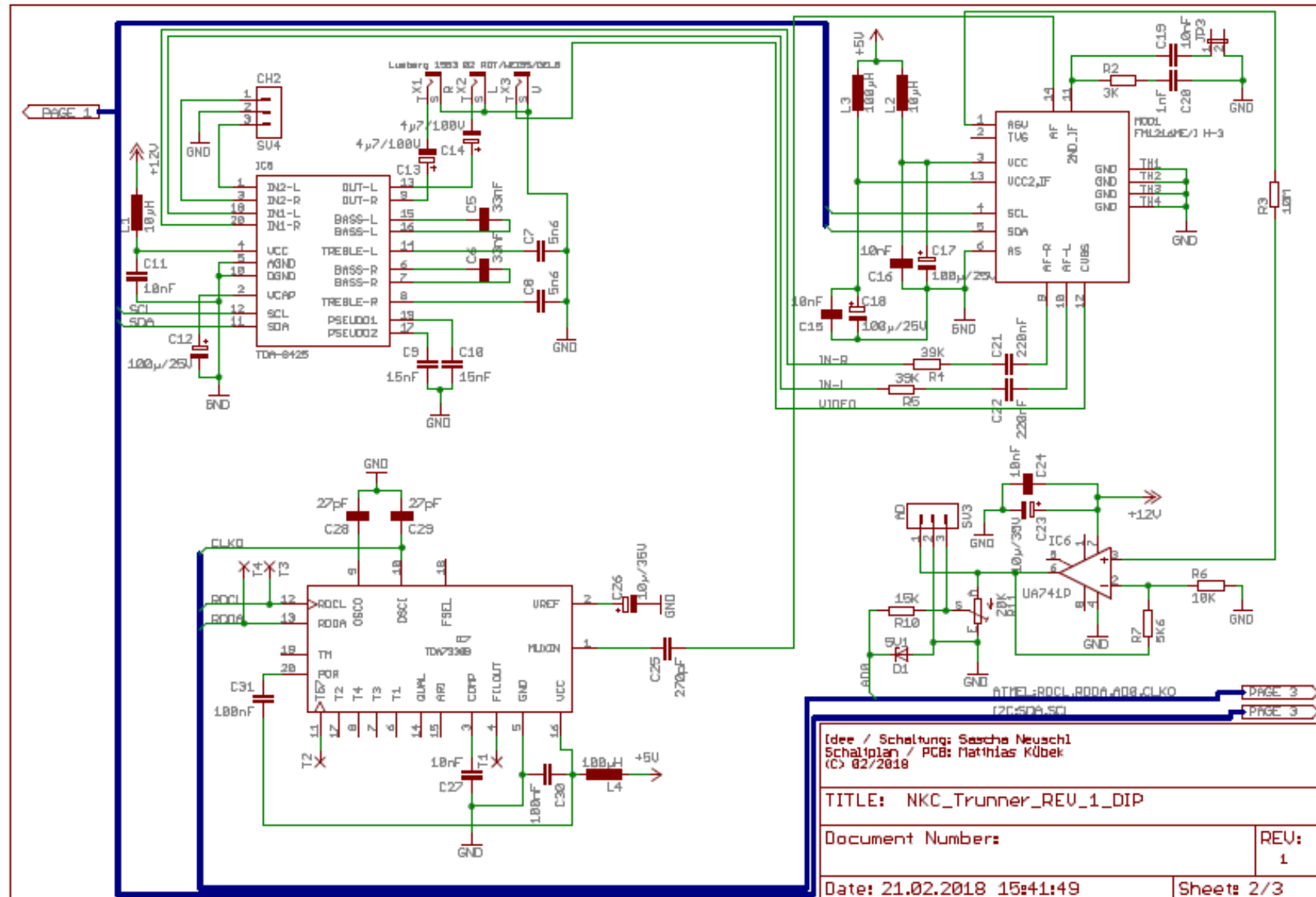
Ähnlich wie bei den **IO-Adressen auf dem NKC-Bus** müssen alle Geräte auf dem **I2C-Bus** eine **Adresse (I2C module address MAD)** besitzen, über die sie angesprochen werden. Wenn Geräte einen Schreib- und einen Lesezugriff zulassen, dann besitzen sie für jeden dieser Zugriffe eine eigene **I2C-Adresse**. Die folgende Tabelle zeigt die verwendeten **I2C-Adressen**:

Gerät	Funktion	Baustein	I2C-Adresse (MAD)	Bemerkung
Master	I2C-Prozessor	PCF 8584-P	\$AA	Adresse wird im Bedienprogramm bei der Initialisierung mitgegeben. Baustein wird nur im Master-Betrieb eingesetzt. Adresse ist nicht weiter relevant, muss aber gesetzt werden!
Slave	Tuner_HF	Philips FM1216 ME / I H-3	\$C0 - Schreibzugriff \$C1 - Lesezugriff	Adresse ist fest, kann aber in bestimmten Grenzen durch Beschaltung von PIN 6 geändert werden (siehe Datenblatt FM1216ME_MK3.pdf). In der Schaltung liegt PIN 6 auf 0V.
Slave	Tuner_IF	Philips FM1216 ME / I H-3	\$86 - Schreibzugriff \$87 - Lesezugriff	Adresse ist fest, nicht änderbar.
Slave	Sound	TDA 8425	\$82 - Schreibzugriff	Adresse ist fest, nicht änderbar.
Slave	Speicher – 2 KByte	AT24C02	\$A0 - Schreibzugriff \$A1 - Lesezugriff	Adresse ist fest, kann aber in bestimmten Grenzen durch Beschaltung der PINs A0 bis geändert werden (siehe Datenblatt AT24C02.pdf). In der Schaltung liegen alle PINs auf 0V.
Slave	RDS-Erfang / Signalstärke	ATMEGA 168-P	\$\$ - Schreibzugriff \$\$ - Lesezugriff	Erst in der Nächsten Version der Spezifikation!

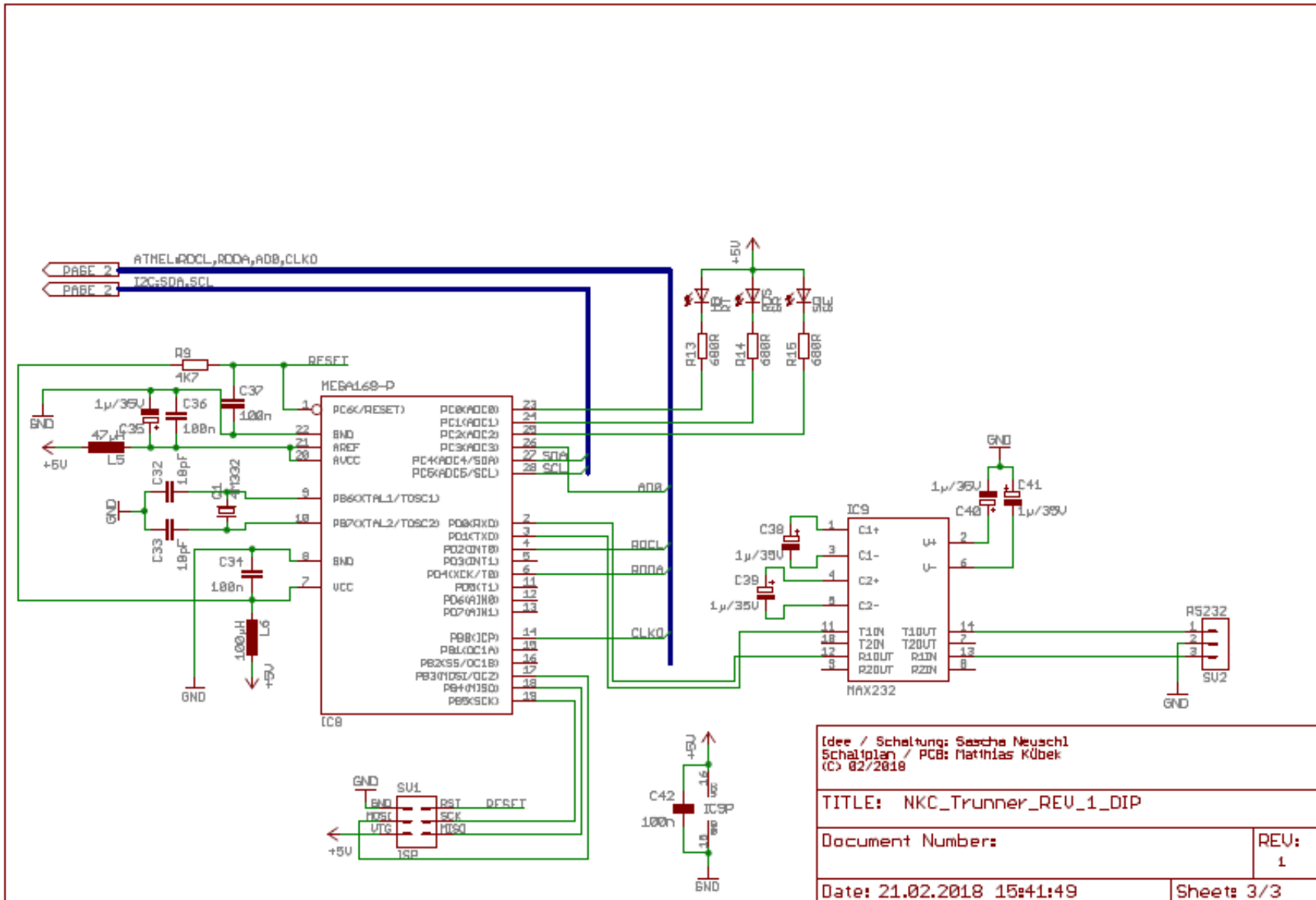
4 Schaltplan



Tuner-Karte für den NDR-Klein-Computer



Tuner-Karte für den NDR-Klein-Computer



5 Anmerkungen

5.1 Spannungsversorgung:

Um HF-Störungen von den Spannungsversorgungen fernzuhalten, wurden folgende Kombinationen aus Spule und Kondensator für die verschiedenen Bausteine eingesetzt:

Baustein	Spule	Kondensator
Philips FM1216 ME / I H-3 - <i>HF</i>	10 μ H	10 nF
Philips FM1216 ME / I H-3 - <i>IF</i>	100 μ H	10 nF
TDA 8524	10 μ H	10 nF
TDA 7330	100 μ H	100 nF
UA 741-P	-- -- --	10 nF
ATMEGA 168-P	100 μ H	100 nF

5.2 Eingänge und Ausgänge:

Hier werden alle Ein- und Ausgänge sowie Steckverbinder und Jumper der Tunerkarte beschrieben:

Steckerleiste/ Buchse	Beschreibung	Standard	Bemerkung
JP1	Jumper zur Einstellung der NKC-IO-Adresse des I2C-Prozessors PCF 8584-P bzw. der Tuner-Karte	ja	Im Bedienprogramm ist die IO-Adresse \$FFFFFF52 verwendet.
JP3	Zweite Sound-IF bei TV-Empfang	optional	Mit einem weiteren Baustein wird 2-Kanal- oder Stereo-TV-Empfang ermöglicht.

Tuner-Karte für den NDR-Klein-Computer

SV1	ISP-Programmierschnittstelle (SPI) ATMEGA 168-P	ja	1 = GND, 2 = MOSI, 3 = VTG, 4 = RESET, 5 = SCK, 6 = MISO
SV2	Serielle Schnittstelle für den RDS-Betrieb	ja	Übertragung von RDS-Zeichen an den NKC und Steuerung des RDS-Dekoders 1 = TX, 2 = GND, 3 = RX
SV3	AGC-Regelspannung des Tuners zur Anzeige der Signalstärke eines Senders	ja	1 = out OP-Verstärker 741, 2 = GND, 3 = out - einstellbar
SV4	Eingang 2. Kanal des Soundbausteins TDA 8425	optional	Man kann hier z.B. den Ausgang der NKC-Soundplatine anschließen. Auf Kanal 1 liegt der Tuner!
SV5	I2C-Bus	optional	Hier können weitere Bausteine an den I2C-Bus angeschlossen werden.
X1	Audio-Signal - rechts	ja	Anschluss eines NF-Verstärkers
X2	Audio-Signal - links	ja	Anschluss eines NF-Verstärkers
X3	Video-Signal (TV)	ja	CVBS- oder BAS-Signal an 75 Ohm Impedanz
Antenneneingänge Radio und TV	HF-Eingang mit 75 Ohm Impedanz	ja	Wichtig: Der NKC „sendet“ mit seinem CPU-Takt und dem Takt der Grafikkarte in der Nähe des UKW-Bandes oder zumindest dessen IF von 10,7 MHz. Deshalb sollte man an die Antennenbuchse auch eine etwas abseits stehende Antenne mit Koaxialkabel und – stecker anschließen. Sonst rauscht es eventuell nur, obwohl die Karte funktioniert und richtig programmiert ist! Ein UKW-Bandpass (87,5 -108 MHz) vor dem Radio-Antenneneingang hilft ebenfalls, Störungen zu vermeiden und erhöht die Selektivität des Tuners.

6 Test und Programmierung der Karte

Die Tunerkarte kann mit den im Folgenden beschriebenen Programmteilen - am Beispiel von 680XX-Assembler getestet werden. Es existiert dazu die Datei `,tunertst.asm'`.

6.1 Programmierung mit Schichtenkonzept:

Die Programmierung der Tuner-Karte erfolgt in einem **Schichtenkonzept**:

In der **ersten Schicht** gibt es **Basisroutinen**, um den **I2C-Prozessor** auf seinem **Statusregister** oder seinem **Datenregister** anzusprechen – **mit schreibendem** oder **lesendem Zugriff**.

In der **zweiten Schicht** gibt es **Basisroutinen**, um den **I2C-Prozessor zu initialisieren**, **n Datenbytes** an den **I2C-Bus zu senden** oder von ihm **zu lesen**.

In der **dritten Schicht** gibt es **Routinen**, die es erlauben, die **einzelnen Bausteine** (Tuner, Soundbaustein, Speicher – EEPROM und ATMEGA168) **anzusprechen**.

Schicht	Objekt	Funktion
1	Status- und Datenregister des I2C-Prozessors PCF 8584-P	I2C-Prozessor über NKC-IO-Adresse ansprechen und Register schreiben oder lesen
2	I2C-Bus	I2C-Prozessor initialisieren, n Datenbytes auf den I2C-Bus schreiben oder von ihm lesen
3	I2C-Bausteine (Tuner, Soundbaustein, Speicher – EEPROM, ATMEGA168 als Slave für RDS - noch nicht implementiert)	Programmierung der I2C-Bausteine

6.2 Schicht 1 - Beispielcode:

Bemerkungen:

- 1) Es wurden die **NKC IO-Adressen \$FFFFFF52** (Datenregister) und **\$FFFFFF53** (Statusregister) für den **I2C-Prozessor** gewählt.

```

;*****
;*
;*      Tunertestprogramm by sEn (Sascha Neuschl)      *
;*      Version 1.0 - 02.07.2013                      *
;*
;*****
;* Hardwarekonfiguration:                             *
;*
;* Die Tunerkarte stellt in erster Linie ein          *
;* Interface zum I2C-Bus von Philips dar.             *
;* Der Interfacebaustein ist der PCF 8584, der         *
;* über den parallelen 8-Bit-Datenbus angesteuert wird.*
;*****
;* Der I2C-Bus:                                       *
;*
;* besteht aus dem I2C-Prozessor PCF 8584 (Master), dem *
;* Tuner FM1216 ME / I H-3 dem Vorverstärker- und Klang- *
;* stell-IC TDA 8425 und einem seriellen EEPROM AT24C02B *
;* für die Speicherung von Programmplätzen für Radio/TV *
;* sowie den Initialeinstellungen des Tuners und Vorver- *
;* stärker-/ Klangregel-ICs                           *
;*****
;* Die Adressierung der Karte:                       *
;*
;* liegt zurzeit auf $FFFFFF52.                      *
;* Zur Ansteuerung des I2C-Prozessors werden 2 Adressen *
;* benötigt.                                          *
;*****

```


Tuner-Karte für den NDR-Klein-Computer

```

;***Schicht 1:

;* I2C-Basisroutinen zur Ansteuerung des I2C-Prozessors PCF 8584

;* Vereinbarungsbaustein

;* Register des I2C-Prozessors PCF8584

CTRLSTAT equ $fff53 ;Control-/Statusregister
REG       equ $fff52 ;Standardregister

;* I2C-Adressen der Peripheriebausteine

TUNERW     equ $C0 ;tuner - write
TUNERR     equ $C1 ;tuner - read
TUNERIFW   equ $86 ;tuner-if - write
TUNERIFR   equ $87 ;tuner-if - read

SOUNDW     equ $82 ;sound - write
SOUNDR     equ $83 ;sound - read

MEMORW     equ $A0 ;memory - write
MEMORR     equ $A1 ;memory - read

;* Datenpufer für zu sendende oder lesende Bytes vom I2C-Bus und nostop-Flag für
;* senden ohne Stopcondition

sendbuff:   ds.b 10
readbuff:   ds.b 10
nostop:     ds.b 2

;* Routine sendctrl schreibt ein Byte in das Control-/Stusregister
;* Übergaberegister ist D1

sendctrl:
move.b d1,(CTRLSTAT)
rts

```

Tuner-Karte für den NDR-Klein-Computer

```

; * Routine sendreg schreibt ein Byte in ein zuvor über das Control-/
; * Statusregister ausgewähltes Register. Übergaberegister ist D2.

```

```

sendreg:
move.b d2, (REG)
rts

```

```

; * Routine readctrl liest ein Byte aus dem Control-/Statusregister
; * Übergaberegister ist D1.

```

```

readctrl:
move.b (CTRLSTAT), d1
rts

```

```

; * Routine readreg liest ein Byte aus einem zuvor über das Control-/
; * Statusregister ausgewählten Register. Übergaberegister ist D2.

```

```

readreg:
move.b (REG), d2
rts

```

6.3 Schicht 2 - Beispielcode:

Bemerkungen:

- 1) In der Routine zur Initialisierung wurde von einer **CPU-Taktrate des NKC** von **8 MHz** und **2 Wait-States** ausgegangen.
- 2) Für Diagramme zum Ablauf der Initialisierung des I2C-Prozessors und des Sendens und Empfangens von Datenbytes an bzw. vom I2C-Bus siehe Dokument „PCF8584.pdf“.

;* Schicht 2:**

```

; * Routine INIT initialisiert den I2C-Prozessor PCF 8584

```

```

init:
move.b #$80, d1 ;Register S0' wird ausgewählt.
jsr sendctrl
move.b #$55, d2 ;$55 wird in Register S0' (eigene I2C-Adresse) geschrieben.
jsr sendreg      ;Effektive I2C-Busadresse ist damit $AA (siehe Datenblatt)

```

Tuner-Karte für den NDR-Klein-Computer

```

;Die Adresse ist unerheblich, weil der I2C-Prozessor nur als
;Master eingestzt wird und von keinem anderen I2C-Device ange-
;sprochen wird!
;Aber der Schritt ist notwendig, damit der I2C-Prozessor
;erkennt, ob es sich um einen INTEL- oder Motorola-Bus handelt!
move.b #$A0, d1 ;Register S2 (Clock-Register) wird ausgewählt
jsr sendctrl
move.b #$18, d2 ;$18 wird in Register S2 (Clockregister) geschrieben.
jsr sendreg      ;Es wird 8 MHz Systemtakt und 90 KHz I2C-SDL-Takt gesetzt.
move.b #$C1, d1 ;Im Statusregister S1 wird das serielle Interface des I2C-
jsr sendctrl      ; Busses aktiviert.
rts               ; Ende der Initialisierung

```

```

; * Routine Master Transmitter Mode sendet n Bytes über das serielle Interface
; * des I2C-Busses. In D3 wird die I2C-Adresse des anzusprechenden Peripherie-
; * bausteins übergeben. In D4 wird die Anzahl zu sendender Bytes übergeben, in
; * D5 die Anzahl der gesendeten Bytes gezählt. Die zu übertragenden Bytes
; * stehen im Sendbuffer sendbuff.
; * Wenn keine Stopcondition ausgelöst werden soll, sondern eine Repeatedstart-
; * condition, dann steht dies in nostop im ersten Byte als Wert $EE.
; * Soll der Busbusy-Check übersprungen werden, so steht dies in nostop im
; * zweiten Byte als Wert $BB.

```

```

mastertr:
; * Initialisierung
clr.b d5          ;Zähler für gesendete Bytes initialisieren
lea sendbuff, A0;Sendbuffer holen
lea nostop, A1    ;Sonderbedingungen laden für Repeatedstart oder no Busbusy-Check
move.b 1(A1), d0;Test, ob Busbusy-Check und Startcondition ausgeschaltet sind

cmp.b #$BB, d0
beq.s adronly
busbusyt:
jsr readctrl      ;Statusregister S1 auslesen
and.b #1, d1      ;Test auf Bit BB - bus busy
beq.s busbusyt    ;wenn 0, dann warten und nochmal lesen
move.b d3, d2     ;I2C-Adresse des Peripheriebausteins übergeben (Slave-Adresse)
jsr sendreg       ;Ausgabe auf das S0-Register - Datenregister zum Senden/
                  ;Empfangen von Bytes des I2C-Busses
move.b #$C5, d1   ;Startcondition für I2C-Bus wird ausgelöst
jsr sendctrl

```

Tuner-Karte für den NDR-Klein-Computer

```

bra.s finisht      ;Mit Startcondition weiter bei finisht
adronly:
move.b d3, d2      ;I2C-Adresse des Peripheriebausteins übergeben (Slave-Adresse)
jsr sendreg        ;Ausgabe auf das S0-Register - Datenregister zum Senden/
                    ;Empfangen von Bytes des I2C-Busses

finisht:
jsr readctrl       ;Statusregister S1 auslesen
move.b d1, d0      ;Wert aus d1 für zweiten Bittest retten
and.b #128, d1     ;Test auf PIN-Bit - Übertragung beendet?
bne.s finisht      ;Wenn 1, dann warten und nochmal Statusregister lesen
and.b #8, d0       ;Test auf LRB-Bit - slave acknowledged?
bne.s stopt        ;Wenn 1, dann Übertragung beenden
cmp.b d4,d5        ;Alle Bytes übertragen?
beq.s stopt        ;Wenn ja, dann Übertragung beenden
addq #1, d5        ;Zähler für übertragene Bytes um 1 inkrementieren
move.b (A0)+, d2; nächstes Byte aus Sendbuffer holen
jsr sendreg        ;umd übertragen
bra.s finisht      ;Schleife, solange noch nicht alle Bytes gesendet sind
stopt:
move.b 0(A1), d0; Test, ob Repeatedstartcondition vorliegt
cmp.b #$EE, d0
beq.s repstart
move.b #$C3, d1 ;Stopcondition über Controlregister auslösen
jsr sendctrl
bra.s readyt
repstart:
move.b #$45, d1 ;Repeatedstartconditon wird ausgelöst
jsr sendctrl
readyt:
move.b #$00, 0(A1)      ;Stopcondition wieder einschalten
move.b #$00, 1(A1)      ;Busbusy-Check und Startcondition wieder einschalten
move.l #1, d0          ;Warten, bevor nächster Aufruf von mastertr oder masterec,
jsr @delay             ;sonst zu schnell!
rts

```

```

; * Routine Master Receiver Mode empfängt n Bytes über das serielle Interface
; * des I2C-Busses. In D3 wird die I2C-Adresse des anzusprechenden Peripherie-
; * bausteins übergeben. In D4 wird die Anzahl zu empfangender Bytes übergeben,
; * in D5 die Anzahl der empfangenen Bytes gezählt. Die zu empfangenden Bytes
; * stehen im Readbuffer readbuff.
; * Wenn keine Stopcondition ausgelöst werden soll, sondern eine Repeatedstart-

```

Tuner-Karte für den NDR-Klein-Computer

```

; * condition, dann steht dies in nostop im ersten Byte als Wert $EE.
; * Soll der Busbusy-Check übersprungen werden, so steht dies in nostop im
; * zweiten Byte als Wert $BB.

masterec:
; * Initialisierung
clr.b d5          ; Zähler für empfangene Bytes initialisieren
subq #1, d4        ; Zähler für zu empfangende Bytes wird um 1 dekrementiert, weil
                    ; das letzte Byte besonders behandelt wird
lea readbuff, A0   ; Readbuffer holen
lea nostop, A1     ; Sonderbedingungen laden für Repeatedstart oder no Busbusy-Check

move.b d3, d2      ; I2C-Adresse des Peripheriebausteins übergeben (Slave-Adresse)
jsr sendreg        ; Ausgabe auf das S0-Register - Datenregister zum Senden/
                    ; Empfangen von Bytes des I2C-Busses

move.b 1(A1), d0   ; Test, ob Busbusy-Check ausgeschaltet ist
cmp.b #$BB, d0
beq.s finishr

busbusyr:
jsr readctrl       ; Statusregister S1 auslesen
and.b #1, d1       ; Test auf Bit BB - bus busy
beq.s busbusyr     ; wenn 0, dann warten und nochmal lesen
move.b #$C5, d1    ; Startcondition für I2C-Bus wird ausgelöst
jsr sendctrl

finishr:
jsr readctrl       ; Statusregister S1 auslesen
move.b d1, d0      ; Wert aus d1 für zweiten Bittest retten
and.b #128, d1     ; Test auf PIN-Bit - Übertragung beendet?
bne.s finishr      ; Wenn 1, dann warten und nochmal Statusregister lesen
and.b #8, d0       ; Test auf LRB-Bit - slave acknowledged?
bne.s stopr        ; Wenn 1, dann Übertragung beenden
cmp.b d4, d5       ; Alle Bytes übertragen, bis auf letztes?
beq.s lastbyte     ; Wenn ja, dann letztes Byte behandeln
addq #1, d5        ; Zähler für übertragene Bytes um 1 inkrementieren
jsr readreg        ; nächstes Byte empfangen
move.b d2, (A0)+   ; nächstes Byte in Readbuffer schreiben
bra.s finishr      ; Schleife, bis vorletztes Byte empfangen wurde

lastbyte:
move.b #$40, d1    ; negatives Acknowledgement wird in Controlregister gesetzt
jsr sendctrl
jsr readreg        ; Dies ist ein Dummyread der Slave-Adresse und wird nicht im
                    ; Readbuffer gespeichert

```

Tuner-Karte für den NDR-Klein-Computer

```

finishlb:
jsr readctrl      ;Statusregister S1 auslesen
and.b #128, d1    ;Test auf PIN-Bit - Übertragung beendet?
bne.s finishlb   ;Wenn 1, dann warten und Stausregister erneut auslesen
stopr:
move.b #$C3, d1  ;Stopcondition über Controlregister auslösen
jsr sendctrl
jsr readreg      ;letztes zu empfangendes Byte holen
move.b d2, (A0)  ;und im Readbuffer speichern
move.b #$00, 0(A1) ;Stopcondition wieder einschalten
move.b #$00, 1(A1) ;Busbusy-Check und Startcondition wieder einschalten
move.l #1, d0    ;Warten, bevor n chster Aufruf von mastertr oder masterec,
jsr @delay      ;sonst zu schnell!
rts

```

6.4 Schicht 3 - Beispielcode:

Bemerkungen:

- 1) Am Anfang befindet sich jeweils ein Testprogramm für das **EEPROM**, den **Soundbaustein** und den **Tuner**.
- 2) Das Programm „Start“ ruft nacheinander „Initialisierung des I2C-Prozessors“, „Einstellen des Soundbausteins“ und „Einstellen eines festen UKW-Senders im Tuner“ auf, sodass die Tunerkarte nun einen Output liefern sollte. Das EEPROM wird hier nicht verwendet.

;*** Schicht 3:

EEPROM: ;EEPROM-Test

```

jsr init          ;PCF8584 initialisieren

;schreiben
clr.l sendbuff    ;alle Buffer - mindestens am Anfang - initialisieren
clr.l readbuff
clr.w nostop
move.b #MEMORW, d3 ;I2C-Adresse zum Schreiben des EEPROMS
lea sendbuff, A0   ;Sendbuffer holen
lea nostop, A1     ;nostop-Flag holen
move.b #$00, (A1)+ ;mit Stopcondition
move.b #$00, (A1)  ;mit Busbusy-Check
move.b #2, d4      ;Anzahl zu sendender Bytes
move.b #$00, (A0)+ ;Es wird auf Adresse $00 im EEPROM geschrieben
move.b #$CC, (A0)  ;Es wird der Wert $CC geschrieben
jsr mastertr      ;und senden

;lesen

move.b #MEMORW, d3 ;I2C-Adresse zum Schreiben des EEPROMS
;Dummywrite mit Speicheradresse des EEPROMS

```

Tuner-Karte für den NDR-Klein-Computer

```

lea sendbuff, A0          ;Sendbuffer holen
lea nostop, A1            ;nostop-Flag holen
move.b #$EE, (A1)+        ;mit Repeatedstartcondition
move.b #$00, (A1)         ;mit Busbusy-Check
move.b #1, d4             ;Anzahl zu sender Bytes
move.b #$00, (A0)         ;Es wird im EEPROM die Adresse $00 ausgewählt
jsr mastertr

read:
move.b #MEMORR, d3        ;I2C-Adresse zum lesen des EEPROMS
                           ;Jetzt Auslesen des Werts mit Leseroutine auf I2C-Bus
lea nostop, A1            ;nostop-Flag holen
move.b #$00, (A1)+        ;mit Stopcondition
move.b #$BB, (A1)         ;ohne Busbusy-Check
move.b #1, d4             ;Anzahl zu empfangender Bytes
jsr masterec              ;und Lesen - der Readbuffer wird durch die Routine
                           ;masterec gefüllt!

rts

```

SOUND: ;Sound-Test

```

clr.l sendbuff ;alle Buffer - mindestens am Anfang - initialisieren
clr.l readbuff
clr.w nostop

move.b #SOUNDW, d3        ;I2C-Adresse, um auf den Soundbaustein zu schreiben
                           ;Diesen Baustein kann man nur schreiben!
lea sendbuff, A0          ;Sendbuffer holen
lea nostop, A1            ;nostop-Flag holen
move.b #$EE, (A1)+        ;mit Repeatedstartcondition
move.b #$00, (A1)         ;mit Busbusy-Check, weil erstes Byte
move.b #2, d4             ;Anzahl zu sender Bytes
move.b #$00, (A0)+        ;Register VL
move.b #$F8, (A0)         ;Wert
jsr mastertr

move.b #SOUNDW, d3
lea sendbuff, A0
lea nostop, A1
move.b #$EE, (A1)+        ;mit Repeatedstartcondition
move.b #$BB, (A1)         ;ohne Busbusy-Check
move.b #2, d4
move.b #$01, (A0)+        ;Register VR
move.b #$F8, (A0)         ;Wert
jsr mastertr

move.b #SOUNDW, d3
lea sendbuff, A0
lea nostop, A1
move.b #$EE, (A1)+        ;mit Repeatedstartcondition
move.b #$BB, (A1)         ;ohne Busbusy-Check
move.b #2, d4
move.b #$02, (A0)+        ;Register BA
move.b #$FF, (A0)         ;Wert
jsr mastertr

move.b #SOUNDW, d3
lea sendbuff, A0
lea nostop, A1
move.b #$EE, (A1)+        ;mit Repeatedstartcondition
move.b #$BB, (A1)         ;ohne Busbusy-Check
move.b #2, d4
move.b #$03, (A0)+        ;Register TR
move.b #$FF, (A0)         ;Wert
jsr mastertr

```


Tuner-Karte für den NDR-Klein-Computer

```

move.b #SOUNDW, d3
lea sendbuff, A0
lea nostop, A1
move.b #$00, (A1)+      ;mit Stopcondition, weil letztes Byte
move.b #$BB, (A1)       ;ohne Busbusy-Check
move.b #2, d4
move.b #$08, (A0)+      ;Register SW (Eingang 1 oder 2)
move.b #$CE, (A0)       ;Wert = Eingang 1 - Tuner, lineares Stereo
jsr mastertr
rts

```

TUNER: ;Tuner-Test

```

clr.l sendbuff ;alle Buffer - mindestens am Anfang - initialisieren
clr.l readbuff
clr.w nostop

;* Erst TV-Mode setzen, damit Tuningspannung nicht auf 0 stehen bleibt
;* low band - 150 MHz

move.b #TUNERW, d3      ;I2C-Adresse für Schreiben der Tunersektion
lea sendbuff, A0        ;Sendbuffer holen
lea nostop, A1          ;nostop-Flag holen
move.b #$00, (A1)+      ;mit Stopcondition
move.b #$00, (A1)       ;mit Busbusy-Check, weil erstes Byte
move.b #5, d4           ;Anzahl zu übertragender Bytes
move.b #$0B, (A0)+      ;Dividerbyte 1
move.b #$F5, (A0)+      ;Dividerbyte 2
move.b #$86, (A0)+      ;Controlbyte
move.b #$44, (A0)+      ;Bandswitchbyte
move.b #$20, (A0)       ;Auxiliarybyte
jsr mastertr            ;und senden

;* Erst danach FM-Mode setzen und Frequenz einstellen

;* FM-Mode - if part

move.b #TUNERIFW, d3    ;I2C-Adresse zum Schreiben der Tuner-IF-Sektion
lea sendbuff, A0
lea nostop, A1
move.b #$00, (A1)+      ;mit Stopcondition
move.b #$00, (A1)       ;mit Busbusy-Check, weil erstes Byte
move.b #4, d4
move.b #$00, (A0)+      ;Erste Registeradresse, weitere 2 durch Autoinkrement
move.b #$4E, (A0)+      ;Switchingbyte (high sensitivity)
move.b #$D0, (A0)+      ;Adjustbyte
move.b #$77, (A0)+      ;Databyte
jsr mastertr

;* Frequenz einstellen - tuner part

move.b #TUNERW, d3      ;I2C-Adresse zum Schreiben der Tuner-Sektion
lea sendbuff, A0
lea nostop, A1
move.b #$00, (A1)+      ;mit Stopcondition
move.b #$00, (A1)       ;mit Busbusy-Check, weil erstes Byte
move.b #5, d4

;eingestellte Frequenz ist 91,2 MHz
;zur Berechnung der Dividerbytes siehe Datenblatt
;des Tuners

move.b #$07, (A0)+      ;Dividerbyte 1
move.b #$F6, (A0)+      ;Dividerbyte 2
move.b #$80, (A0)+      ;Controlbyte
move.b #$19, (A0)+      ;Bandswitchbyte

```

Tuner-Karte für den NDR-Klein-Computer

```
move.b #$B0, (A0)      ;Auxiliarybyte
jsr mastertr
rts
```

start: ;Gesamttest der Karte

```
jsr init      ;I2C-Prozessor initialisieren
jsr sound     ;Soundbaustein einstellen
jsr tuner     ;Tuner einstellen
rts           ; ...und nun hört man endlich Radio am NKC :-)

end
```

7 RDS-Dekoder

7.1 Ansatz

Da es sehr viele Projekte für einen RDS-Dekoder mit einem ATMEGA-Prozessor im Internet gibt, wollte ich an dieser Stelle das Rad nicht neu erfinden. Allerdings handelte es sich fast durchweg um Lösungen, die mit einem LCD-Display arbeiten, und es ergab sich die Frage, wie ich die Daten in den NKC kriegen sollte.

Die einfachste Methode schien ein **Transfer über die RS 232-Schnittstelle** zu sein, so denn der ATMEGA und der NKC schnell genug wären und in der Kommunikation zusammenpassen würden.

Marc verwendet seine Lösung in Zusammenhang mit einer Hauppauge-Karte in seinem PC und lässt sich die RDS-Texte via Hyperterminal auf seinem PC anzeigen. Genau das hatte ich gesucht. Vielen Dank an Marc an dieser Stelle!

Ich fand das folgende Projekt „**Atmel AVR Atmega168 RDS decoder with serial output**“ von **Marc Ketel**: <http://atoomnet.net/atmel-avr-atmega168-rds-decoder-with-serial-output/>
Dort gibt es eine Beschreibung und auch den Code für den ATMEGA 168-P.

Ich änderte die **Übertragungsgeschwindigkeit für die RS 232-Schnittstelle** im Code des ATMEGA 168-P von „**19.200**“ auf „**9.600 BAUD**“, weil der NKC mit einer höheren Geschwindigkeit in meinem System (68008 mit 8 MHz) nicht klarkommt.

Durch diese geringere Übertragungsgeschwindigkeit ergibt sich ein **Laufzeitproblem** bei der **Darstellung des RDS-Texts (RTA/RTB)**. Die ersten 3 Zeichen werden fehlerhaft an den NKC übertragen.

Deshalb habe ich den Original-Code so geändert, dass der **RDS-Text nur jedes zweite Mal ausgegeben** wird. **Jedes erste Mal** wird eine **Stringvariable mit dem RDS-Text gefüllt**, um ihn mit dem **neu zu empfangenden RDS-Text zu vergleichen**, um die **Mehrfachausgabe zu unterdrücken**. Der Stringvergleich u n d die Ausgabe zusammen dauern zu lange. Da die Aussendung des RDS-Texts mehrfach mit demselben Inhalt geschieht, findet kein Informationsverlust statt.

Die Änderungen habe ich im Code im Abschnitt 7.7 in Rot hervorgehoben.

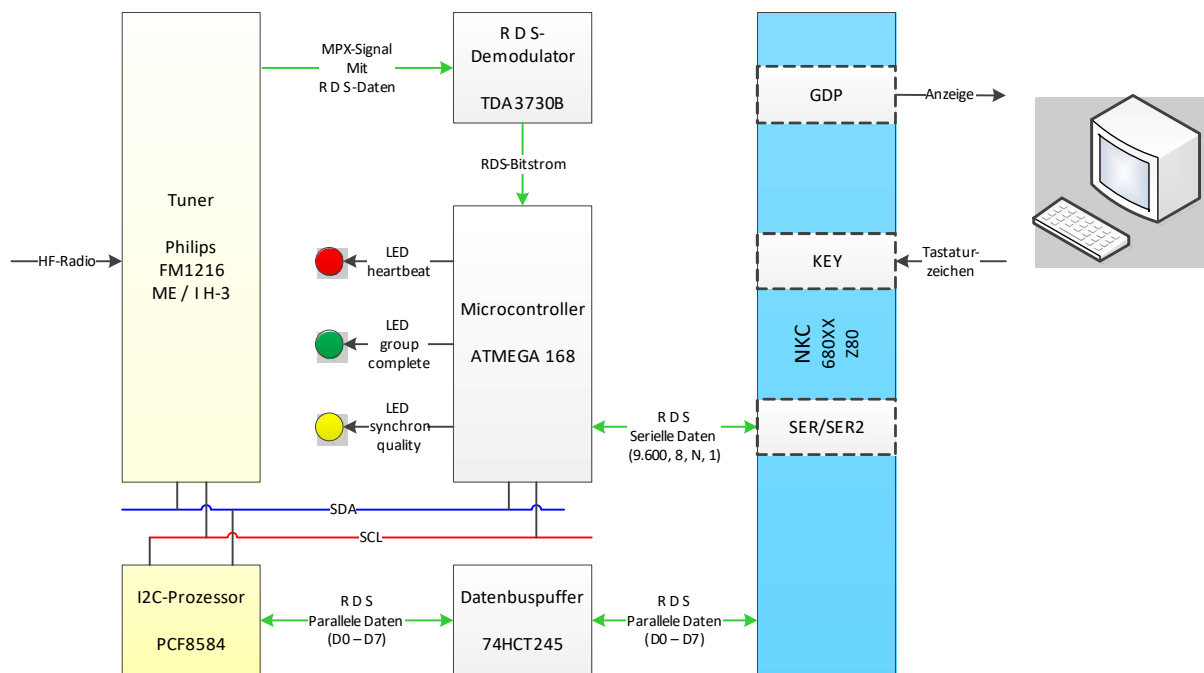
7.2 Aktueller Stand

Thomas Heller und Matthias Kübeck haben nun eine weitere Möglichkeit geschaffen, RDS-Zeichen an den NKC zu übertragen. Dazu wird der **ATMEGA 168-P** (der die Signalaufbereitung des RDS-Datenstroms vornimmt) **als Slave an den I2C-Bus** angebunden. Über den **I2C-Prozessor PCF 8584-P** werden die **RDS-Zeichen aus dem ATMEGA 168-P ausgelesen** bzw. der RDS-Dekoder gesteuert.

Es gibt also aktuell zwei Methoden, den RDS-Dekoder an den NKC anzubinden:

Methode	Standard	Schnittstelle
1	optional (Debug)	RS232: Adresse = Standardadresse, wie sie das Grund-programm für die SER- / SER2-Karte vorsieht Konfiguration = 9.600 BAUD, 8 Bit, keine Parität, 1 Stoppbit (Dies kann im Code des ATMEGA 168-P und im Programmcode der Tunerkarte geändert werden. Allerdings ist dies die höchste Baudrate, die ich mit einer 68008 CPU bei 8 MHz Prozessortakt verarbeiten konnte.)
2	ja	I2C-Bus (noch nicht implementiert)

7.3 Schaltungsprinzip:



Der **Philips-Tuner FM1216 ME / IH-3** liefert an **PIN 14** das **MPX-Signal**, aus dem die **RDS-Daten** gewonnen werden.

Der **Demodulatorbaustein TDA 3730B** filtert aus dem **MPX-Signal** einen **Bit-Strom** aus und liefert ihn an den **Microcontroller ATMEGA 168-P**.

Dieser **dekodiert** aus dem **Bitstrom** zunächst „**Zeichen**“ und **interpretiert** aus den **Zeichen** die relevanten **RDS-Strukturen mit ihren Gruppen und Informationen** (siehe z.B. http://de.wikipedia.org/wiki/Radio_Data_System oder Datenblatt „RDS-GRUNDLAGEN.pdf“).

RDS-Gruppe aus 4 Blöcken:



Beispielbitmuster für die Gruppe „11A“:



Tuner-Karte für den NDR-Klein-Computer

Die **ausgewerteten Informationen** werden vom **ATMEGA 168-P** im **Standard** über den **I2C-Bus** bereitgestellt. **Optional** werden die **Informationen** über die **RS 232-Schnittstelle** des Microcontrollers ausgegeben und können einer **SER- / SER2-Karte** des NKC zugeführt werden.

Es können nicht nur RDS-Informationen von dem ATMEGA 168-P empfangen, sondern auch **Zeichen zur Programmsteuerung übergeben** werden. Im Programm des Microcontrollers sind folgende Zeichen mit Funktionen definiert:

Taste	Funktion
„r“	Reset des Dekoders
„G“ bzw. „g“	Gruppenanzeige „ein“ / „aus“
„B“ bzw. „b“	Anzeige Bitstrom „ein“ / „aus“

Auf der Platine befinden sich **drei LEDs**, die eine **Aussage über den RDS-Datenstrom** machen und eine **weitere LED**, die den **Interrupt-Betrieb des I2C-Prozessors PCF 8584-P** anzeigt:

LED-Farbe	Funktion
Rot	Heartbeat bzw. Bitstrom vorhanden
Grün	RDS-Gruppe vollständig empfangen
Gelb	Synchronqualität
Blau	Interrupt-Betrieb PCF 8584-P

7.4 Funktionen des RDS-Dekoder-Programms:

Die Software des Microcontrollers kann folgende RDS-Informationen ermitteln:

Nr.	Funktion
1	Program Identification code (PI: 0x83C7, Detected new station)
2	Program service name (PS: RADIO538)
3	Program Type code (PTY: 0x0A, Pop Music.)
4	Traffic Program Identification code & Traffic announcement code (TP&TA: 0x01 0x00, Traffic announcements available on this station and maybe via EON on another station.)
5	Music Speech switch code (MS: 0x01, Music is being broadcasted or station does not use MS flag.)
6	Decoder-identification control code (DI: 0x01, Stereo, Dynamic PTY.)
7	Alternative frequency codes (AF: 0x98, 102.7MHz.)
8	Linkage Actuator (LA: 0x00)
9	Extended Country Code (ECC: 0xE3)
10	Radio Text (RTA: Radio 538 = Randstad (Zuid) 102.7 FM)
11	Clock-time and date (CT: 0x1A53CD844, UTC 2006-07-02 (MJD 53918) 13:33:00 +02:00, TIME 15:33:00)

Tuner-Karte für den NDR-Klein-Computer

RDS-Zeichenstrom:

```

RTA: Kulthits und das Beste von Heute. ANTEENNE M NSTER-Der beste Mix!
RTA: 1thits und das Beste von Heute. ANTEENNE M NSTER-Der beste Mix!
CT: 0x188213102, UTC 2014-03-04 (MJD 56720) 19:04:00 +01:00, TIME 20:04:00
RTB: ANTEENNE M NSTER Redaktion und Studios 0251/289540 * * *
RTB: TEIWE M NSTER Redaktion und Studios 0251/289540 * * *
INFO: 0x01, 20060629-1 RDS Decoder by Marc Ketel alias DiNo, marc@atoomnet.net.
INFO: 0x01, 20060629-1 RDS Decoder by Marc Ketel alias DiNo, marc@atoomnet.net.
PI: 0x0399, Detected new station.
PTY: 0x00, None.
TP&TA: 0x01 0x01, Traffic announcement in progress.
MS: 0x01, Music is being broadcasted or station does not use MS flag.
AF: 0x46, 94.5MHz.
TP&TA: 0x01 0x00, Traffic announcements available on this station and maybe via
EDN on another station.
MS: 0x00, Speech is being broadcasted.
AF: 0x27, 91.4MHz.
MS: 0x01, Music is being broadcasted or station does not use MS flag.
PS: FEN
DI: 0x00, Mono, Dynamic PTY.
AF: 0x4F, 95.4MHz.
PS: MUEENSTER
DI: 0x01, Stereo, Dynamic PTY.
RTA: Kulthits und das Beste von Heute. ANTEENNE M NSTER-Der beste Mix!
  
```

Unkodierte Gruppenanzeige:

```

GR00x84050x420F x4823
GR00x310 0x4CD 0xC04
GR0 0x400 0x15F 0x000
GR0 0x00A 0xE1F 0x535
GR09 08400 0xE5F 0x000
GR0990x240E x3020 x2020
GR0x039 0x840 0x1E5F0x0400
GR00x03990x3410 x0007 xC046
GR0039 0x84060x92E70x4412
GR0990x040A xE14F 05354
GR09 x8406 092E7 0A412
GR0 x2401 0454E 0xE45
GR0x8406 x92E7 0A412
GR0x310 0x4CD 0xC04
GR0 08406 0x210 0x000
GR09 0040A 0E14F 0x354
GR00x840 0x42100x0000
GR0x24040x2020 x2044
GR00x806 0x410 0x000
GR09 03410 0x007 0xC46
GR0 0x84080x4197 x27B1
GR039 0x040A0xE14F x5354
GR0x039 0x84080x4197 x27B1
  
```


7.5 Kommunikation zwischen RDS-Dekoder und NKC

Die Kommunikation des **ATMEGA 168-P** über den **I2C-Bus** oder die **RS232-Schnittstelle** gestaltet sich recht problemlos

Bei **Betrieb mit der RS232-Schnittstelle** sind die **korrekten Einstellungen** im Programm des Microcontrollers und für die SER- / SER2-Kartende vorzunehmen.

Wie man auf dem Bild der **RDS-Zeichnestroms** sieht, werden **Umlaute nicht standardmäßig vom NKC dargestellt**. Es sollte ja „Antenne Münster“ heißen. Hier muss man die ankommenden Zeichencodes für Umlaute im Programm auf dem NKC von z.B. „Ü“ in „UE“patchen.

Für die **Anzeige einer „stehenden Zeile“**, wie man sie aus Autoradios kennt, muss man sich die relevante Info aus dem Text **„ausschneiden“** und in eine Variable speichern. Das Programm in dem ATMEGA 168-P schreibt die empfangenen Daten einfach nacheinander weg.

7.6 Test des RDS-Dekoders

Ein Test des RDS-Dekoders lässt sich in zwei Schritten durchführen. Zuerst wird ein Sender im Tuner eingestellt und das entsprechende Programm dafür verlassen. Der Tuner läuft mit seinen Einstellungen ja weiter. Dann wird ein **„Terminalprogramm“** gestartet, das zusammen mit einer **seriellen Schnittstellenkarte** das Empfangen und Senden von Zeichen ermöglicht.

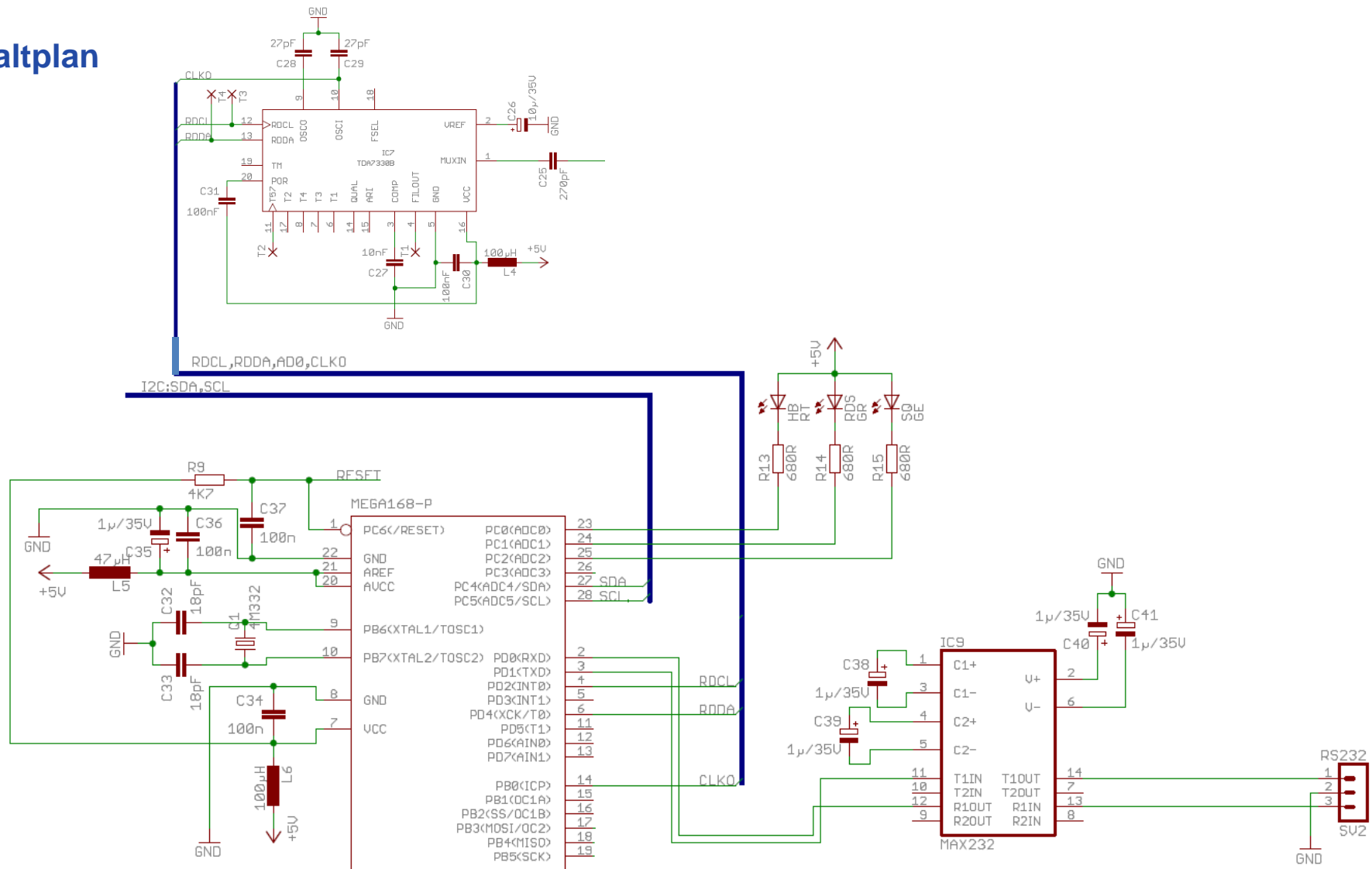
Natürlich kann auch mit dem Bedienprogramm aus Abschnitt 9 getestet werden.

7.7 Testpunkte auf der Platine

Testpunkt	Beschreibung
T1	TDA 7330B – PIN 4: FILOUT (Filterausgang)
T2	TDA 7330B – PIN 11: T57 (Taktausgang mit 57 KHz)
T3	TDA 7330B – PIN 12: RDCL (RDS Takt)
T4	TDA 7330B – PIN 13: RDDA (RDS Daten)

Tuner-Karte für den NDR-Klein-Computer

7.8 Schaltplan



7.9 Programmierung des Microcontrollers ATMEGA 168-P für den RDS-Dekoder

Das ursprüngliche Programm vom Marc Ketel wurde in 2 Punkten geändert:

- Die Baudrate der RS232-Schnittstelle wurde auf 9.600 Baud geändert.
- Der RDS-Text wird nur jedes 2. Mal ausgegeben, um Timing-Probleme mit dem NKC zu vermeiden.

7.9.1 Programmkopf und Verweis auf ATMEL Studio Projekt

/*

Copyright (C) 2006 Marc Ketel

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., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

```
#define INFO "20060629-1 RDS Decoder by Marc Ketel alias DiNo, marc@atoomnet.net."
```

Works with RDS clock signal and RDS data signal. During design a TDA7300B rds demodulator was used.

This source compiles correctly with WinAVR 20060421.

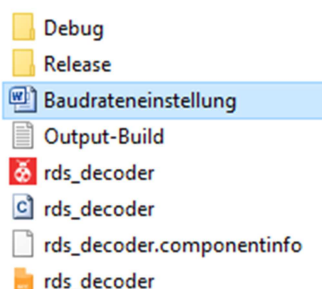
Use a Atmega168 to flash program into. low fuse: 0xF0, high fuse: 0xDD

Versions:

20060629-1 Initial version

20180127-2 Version for use with 9.600 baud serial transfer by Sascha Neuschl sEn
With initial version the serial transfer of RDS text lasts too long and leads to errors within the first 3 to 4 characters.
To fix this RDS text is printed just every second time, not every time! A print flag p is introduced.

Das Programm gibt es als ATMEL Studio Projekt:



8 S-Meter - Messung der Signalstärke

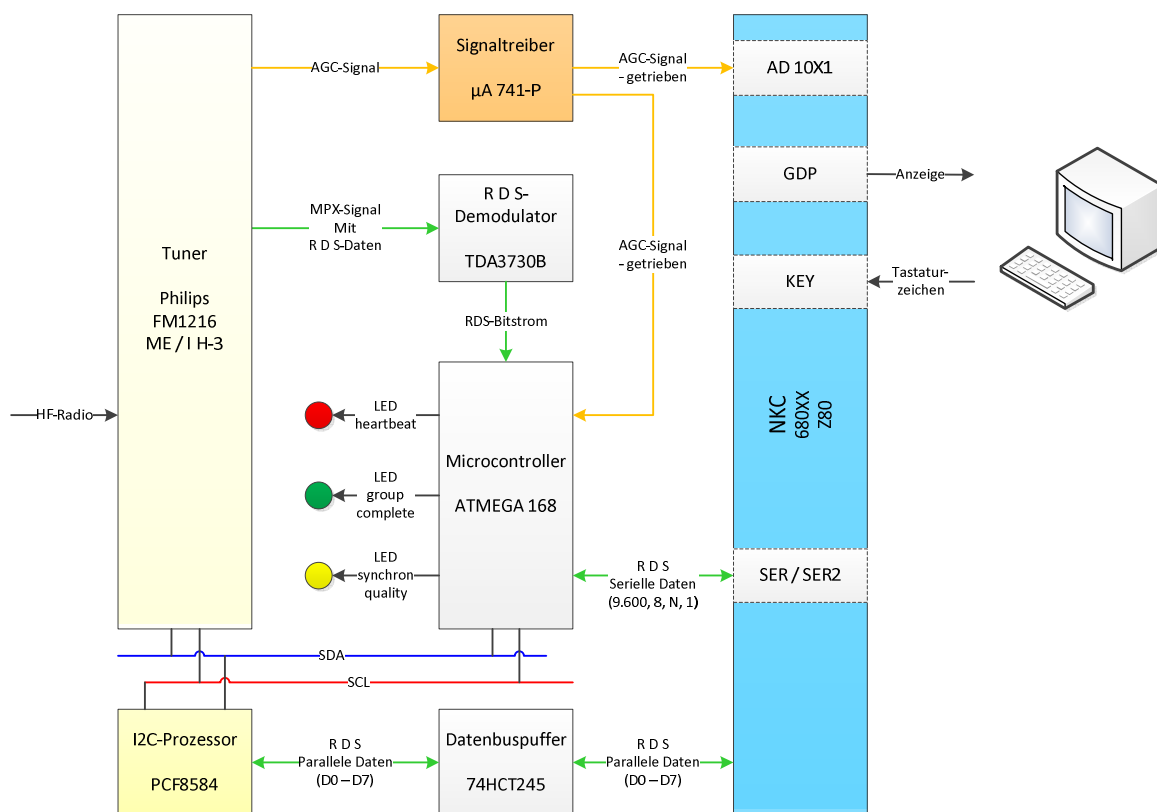
8.1 Ansatz

Der Philips FM1216 ME / I H-3 bietet datentechnisch keinen Wert über die Signalstärke des empfangenen Senders an. Man kann dort nichts über den I2C-Bus abfragen.

Als Ersatz für die Messung der tatsächlichen Signalstärke eines Senders am Antenneneingang, wird in vielen Empfangsgeräten die Tuner-Regelspannung (AGC) herangezogen. Dies ist eine Spannung, die in der IF section des Tuners erzeugt und der Tuner section zugeführt wird, um eine Übersteuerung des Tuners durch einen starken, örtlichen Sender zu verhindern und dafür zu sorgen, dass das IF-Signal immer möglichst konstant hoch ist.

Dieses AGC-Signal verhält sich nun so, dass es bei diesem Tuner bei einem **schwachen Sender hoch** ist und bei einem **starken niedrig**.

8.2 Schaltungsprinzip:



Der Philips FM1216 ME / I H-3 liefert die Tuner-Regelspannung an PIN 1. Diese darf nur sehr hoch-ohmig – **mit minimal 10 Megaohm** – belastet werden, sonst steigt der Tuner aus.

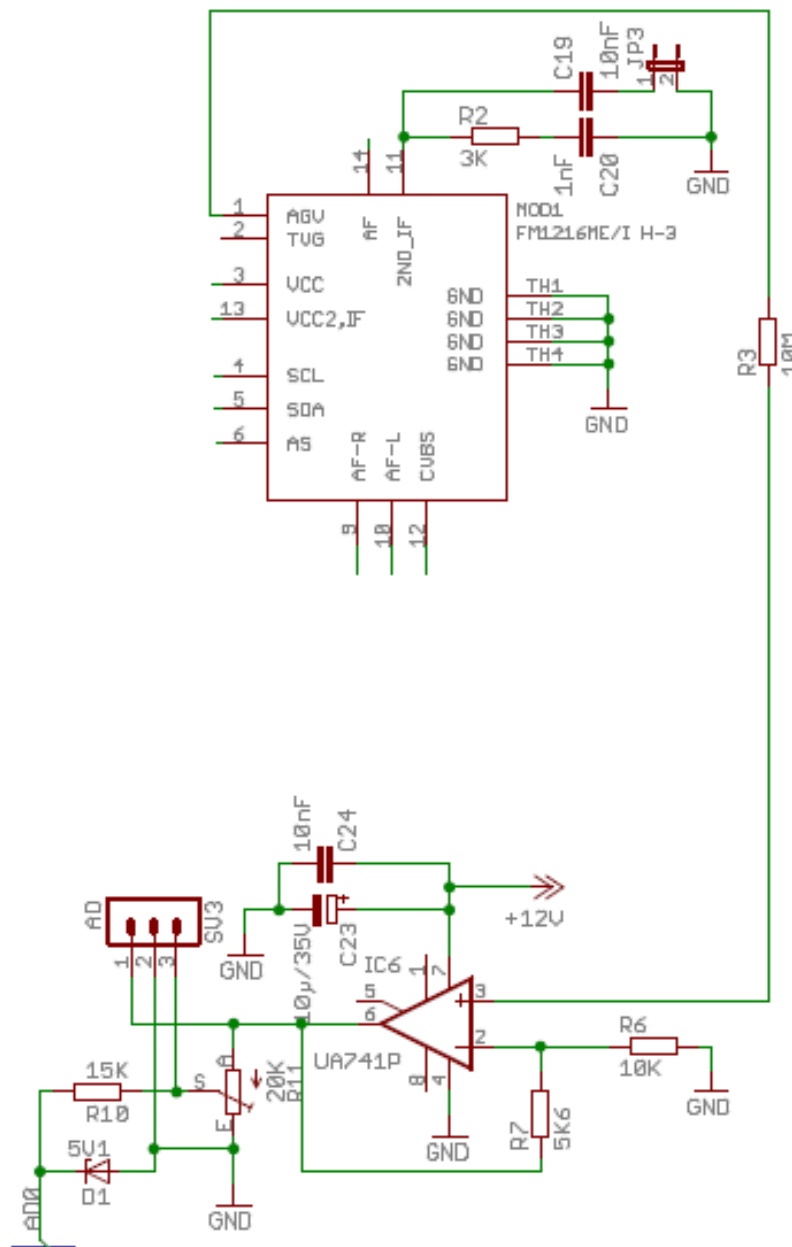
Deshalb wird sie mit dem **Operationsverstärker** µA 741 **als nicht-invertierender Verstärker** getrieben. Die Verstärkung ist auf ca. 1,8-fach eingestellt (10 Kiloohm / 5,6 Kiloohm). Sie kann mit dem Trimmer R11 justiert werden. Eine Z-Diode verhindert, dass die Spannung größer als 5 Volt werden kann. Diese Spannung kann jetzt mit einem Analog-Digital-Wandler ausgewertet werden. Ich nutze

Tuner-Karte für den NDR-Klein-Computer

dazu eine **AD10X1-Karte**. In einem weiteren Ausbau soll der ADC des ATMEGA168 verwendet werden.

Die Treiberschaltung ist in der unteren, rechten Ecke des Tunerschaltplans auf Seite 9 und hier als Ausschnitt angegeben.

8.3 Schaltplan - Ausschnitt:



8.4 Kalibrierung:

Die Signalmessung muss kalibriert werden, damit eine sinnvolle Anzeige der Signalstärke erfolgen kann. Dazu wird einerseits die maximale Spannung am Ausgang des Operationsverstärkers $\mu A 741$ mit dem **Trimmer R11** eingestellt und des Weiteren die Auswertung des Signalwerts des ADCs im Bedienprogramm angepasst.

8.4.1 Ausgangsspannung des Operationsverstärkers:

Zunächst wird die UKW-Antennenbuchse kurzgeschlossen, sodass der Tuner kein Signal empfängt. Jetzt kann die **höchste Spannung mit R11** eingestellt werden, die zu erwarten ist, da die AGC-Spannung bei einem schwachen Sender am höchsten ist.

Ich habe diese Spannung auf **4 Volt** eingestellt.

Jetzt wird die Antenne an die UKW-Antennenbuchse angeschlossen. Z.B. mit der „Auto-Suche“ eines Senders im Bedienprogramm schreite man nun das gesamte Frequenzband ab und notiert die Spannungswerte. Mit meiner Antenne ergibt sich eine **Differenz von höchster zu niedrigster Spannung** von **1,7 Volt**.

8.4.2 Anpassung des Bedienprogramms:

Zu den oben ermittelten Spannungen kann man sich den daraus **ermittelten Wert des ADC** in der oberen rechten Ecke der Hauptmaske des Bedienprogramms anzeigen lassen, wenn man dieses Code-Stück wieder einkommentiert:

```

; * AD10X1 - Test: Für Test wird AD-Wert oben rechts in der Hauptmaske
; angezeigt
;lea ad10x1(pc), a0
;jsr @print4d
;moveq #3-1, d2
;loop:
;move.b #' ', (a0)+
;dbra d2, loop
;clr.b (a0)
;lea ad10x1(pc), a0
;move #$22, d0
;move #440, d1
;move #190, d2
;jsr @write

```

Bei mir ergeben sich die **Werte 384 – 213**. Der Wert des ADCs wird im Bedienprogramm abgefragt. Es sind 5 Signalstufen (Balken) definiert. Ich habe die **Wertdifferenz 171** nun **linear** auf die 5 Balken verteilt. Die Anzeige hat also keinen technischen Hintergrund wie z.B. Dezibel. Die Definition der 5 Signalstufen kann je nach Bedarf geändert werden.

Tuner-Karte für den NDR-Klein-Computer

```

;* Auswertung des gewandelten Wertes für 5 Anzeigebalken
;* Je nach Auswertung wird eine Bitkombination in D3 geladen und
darüber
;* bestimmt, ob ein Balken mit Schreibstift oben oder unten gezeichnet
wird. ADC-Wert in D0.

;* kein Balken
cmp.w #323, d0
ble.s b1
move.b #%00011111, d3
bra.s grafik
;* 1 Balken
b1:
cmp.w #296, d0
ble.s b2
move.b #%00011110, d3
bra.s grafik
;* 2 Balken
b2:
cmp.w #269, d0
ble.s b3
move.b #%00011100, d3
bra.s grafik
;* 3 Balken
b3:
cmp.w #242, d0
ble.s b4
move.b #%00011000, d3
bra.s grafik
;* 4 Balken
b4:
cmp.w #215, d0
ble.s b5
move.b #%00010000, d3
bra.s grafik
;* 5 Balken
b5:
move.b #%00000000, d3

```


9 Bedienprogramm

Es gibt auch ein fertiges Bedienprogramm in 680XX-Assembler für die Tuner-Karte.
Die aktuelle Version ist V3.0 und als Dateien „**nkcradio.asm**“ und „**nkcradio.68k**“ verfügbar.

9.1 Parameter und Systemkonfiguration

- **Bibliothekseintrag**
- **Basisadresse der Tunerkarte (I2C-Prozessor PCF8584)**
- **I2C-Adressen von Tuner, Soundbaustein und EEPROM**
- **Serielle Schnittstellenkarte (RDS):**
 - o Art (SER / SER2)
 - o Konfiguration
- **Analog-Digital-Wandler-Karte (Signalstärke über AGC):**
 - o Fix AD10X1
- **Uhrenkarte**

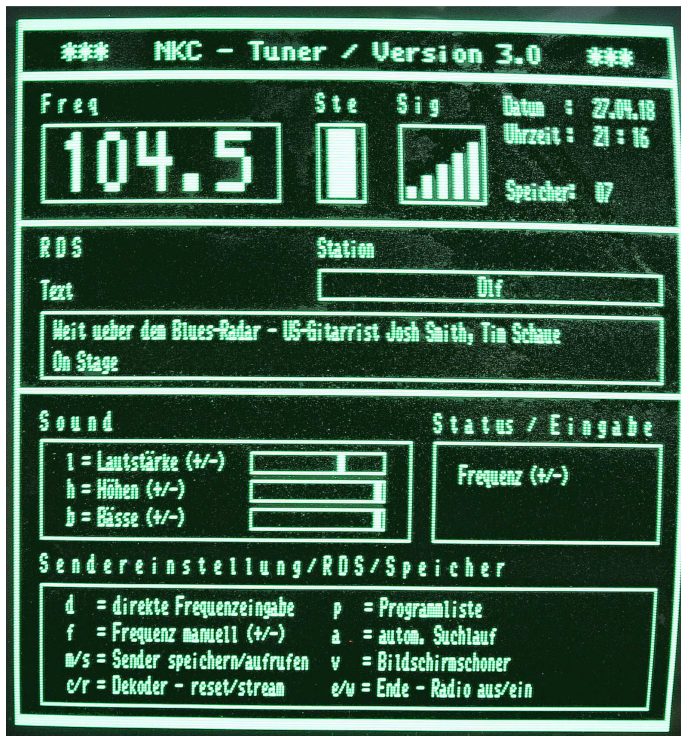
9.2 Funktionen

- **Einstellung der zu empfangenden Frequenz:**
 - o Direkt per Eingabe einer Frequenz
 - o Erhöhen oder Vermindern einer Frequenz um 100 KHz
 - o Automatische Suche des nächsten Senders – nur aufwärts
 - o Speichern eines Senders
 - o Aufruf eines gespeicherten Senders
 - o Aufruf der Programmliste und Auswahl eines Programms/Senders
- **Audioeinstellungen:**
 - o Lautstärke
 - o Höhen
 - o Bässe
- **Anzeigen:**
 - o Datum und Uhrzeit, wenn eine Uhrenkarte im NKC vorhanden ist
 - o Stereo, wenn der Sender in Stereo empfangen wird
 - o Signalstärke des empfangenen Senders in 5 Signalstufen, wenn an ST7 eine AD10X1-Karte angeschlossen ist
 - o RDS:
 - Anzeige des RDS-Stationsnamens der gerade empfangenen Station
 - Anzeige des Radio-Textes der gerade empfangenen Station
 - Anzeige des RDS-Lauftextes in einer separaten Maske
 - o Anzeige der gespeicherten Sender als Programmliste in einer separaten Maske
 - o Einstellung von Lautstärke, Höhen und Bässen
 - o Bedienstatus (z.B. Frequenz, Lautstärke, höhen, Bässe, Sender speichern ...)
- **Speicherung von Einstellungen und Sendern auf dem EEPROM der Karte:**
 - o Die letzten Einstellungen für Frequenz und Audio werden auf dem ersten Speicherplatz im EEPROM gespeichert. Bei Neustart des Programms wird mit diesen Werten gearbeitet
 - o Es können 15 Sender mit ihrer Frequenz und dem RDS-Stationsnamen in einer Programmliste gespeichert werden
- **Bildschirmschoner**

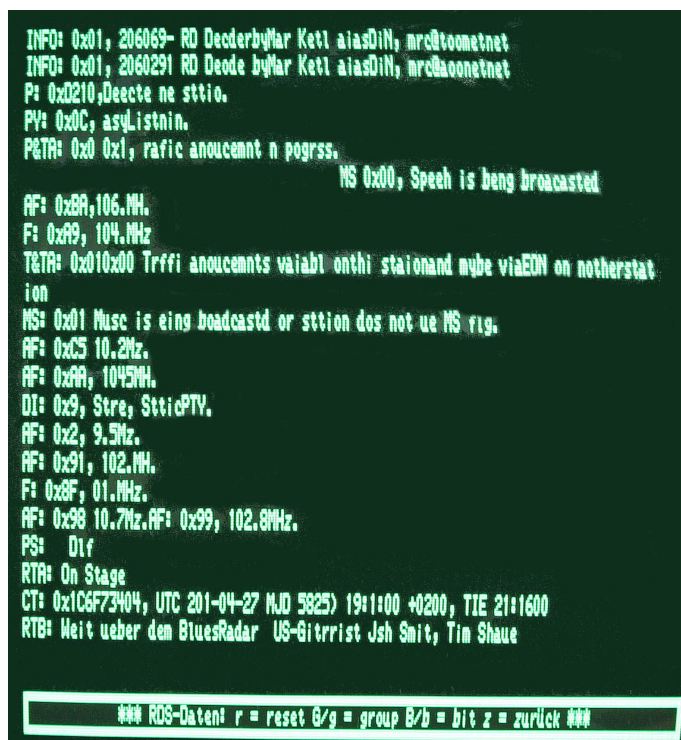
9.3 Masken

Das Programm gliedert sich in die Folgenden Masken:

- Hauptmaske:

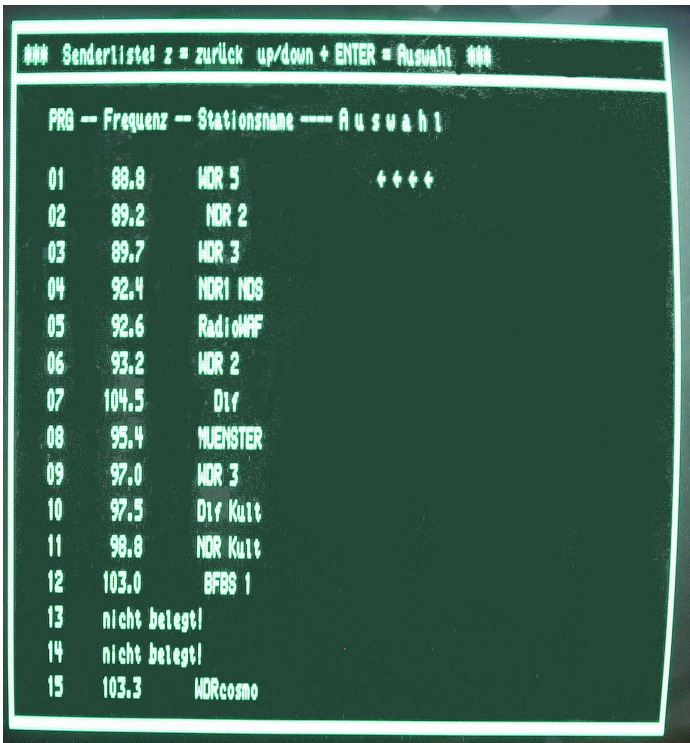


- Maske für RDS-Zeichenstrom:



Tuner-Karte für den NDR-Klein-Computer

- Maske für gespeicherte Sender (Programmliste):



10 Stückliste

10.1 Tunerkarte

10.1.1 Platine

Für die Platine liegen PCB-Dateien vor:

Name	Änderungsdatum	Typ	Größe
Platinenfertigung	29.04.2018 13:47	WinZip-Datei	146 KB
Stückliste	21.02.2018 20:04	Adobe Acrobat D...	341 KB
Tuner_Botton	21.02.2018 19:47	Adobe Acrobat D...	21 KB
Tuner_Layer2	21.02.2018 19:47	Adobe Acrobat D...	125 KB
Tuner_Layer3	21.02.2018 19:47	Adobe Acrobat D...	47 KB
Tuner_Place	21.02.2018 19:47	Adobe Acrobat D...	36 KB
Tuner_PlaceName	21.02.2018 19:47	Adobe Acrobat D...	50 KB
Tuner_PlaceNameValue	21.02.2018 19:47	Adobe Acrobat D...	87 KB
Tuner_PlaceValue	21.02.2018 19:47	Adobe Acrobat D...	74 KB
Tuner_Top	21.02.2018 19:47	Adobe Acrobat D...	18 KB
Tuner_Top_Botton_Best	21.02.2018 19:47	Adobe Acrobat D...	121 KB
Tuner-Schaltplan	21.02.2018 19:47	Adobe Acrobat D...	158 KB

10.1.2 Bauteile:

Name	Wert	Menge	Beschreibung
C1, C2, C3, C4, C30, C31, C34, C36, C37, C42, C43	100n	11	C43 (optional)
C5, C6	33n	2	
C7, C8	5n6	2	
C9, C10	15nF	2	
C11, C15, C16, C19, C24, C27	10n	6	
C20	1n	1	
C21, C22	220n	2	
C25	270p	1	
C28, C29	27p	2	
C32, C33	18p	2	
C12, C17, C18, C44	100μ/25V	4	
C13, C14	4μ7/100V	2	
C23, C26	10μ/35V	2	
C35, C38, C39, C40, C41	1μ/35V	5	C38 - C41 (optional)
D1	5V1	1	

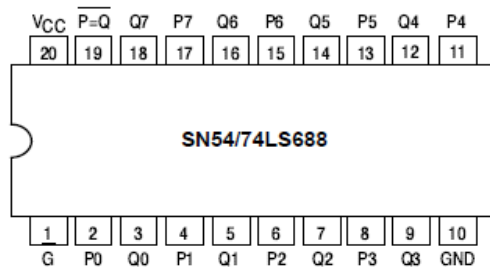
Tuner-Karte für den NDR-Klein-Computer

L1, L2	10µH	2	
L3, L4, L6	100µH	3	
L5	47µH	1	
R1, R13, R14, R15	680R	4	
R2	3K	1	
R3	10M	1	
R4, R5	39K	2	
R6	10K	1	
R7	5K6	1	
R8, R9, R12	4K7	3	
R10	15K	1	
R11	20K	1	Timmer
RN1	4K7	1	
Q1	4M332	1	
SQ	LED 3mm Gelb 2mA	1	
INT, RDS	LED 3mm Grün 2mA	2	
HB	LED 3mm Rot 2mA	1	
IC1	PCF8584P	1	
IC2	74xC688N	1	
IC3	74xC245N	1	
IC4	AT24CP	1	
IC5	TDA-8425	1	
IC6	µA741P	1	
IC7	TDA7330B	1	
IC8	MEGA168-P / MEGA 328-P	1	
IC9	MAX232	1	(optional)
JP1	Stiftleiste 2x7Pol (Adresse)	1	
JP3	Stiftleiste 2Pol	1	
MOD1	Philips FM1216ME	1	
T1 - T4	Testpunkte	4	
X1 - X3	Cinche-Buchsen	3	
CON1	NKC-Busleiste	1	

11 Anhang

11.1 Datenblätter TTL-Bausteine:

11.1.1 74LS688



TYPE	P = Q	P > Q	OUTPUT ENABLE	OUTPUT CONFIGURATION	PULLUP
LS682	yes	yes	no	totem-pole	yes
LS684	yes	yes	no	totem-pole	no
LS688	yes	no	yes	totem-pole	no

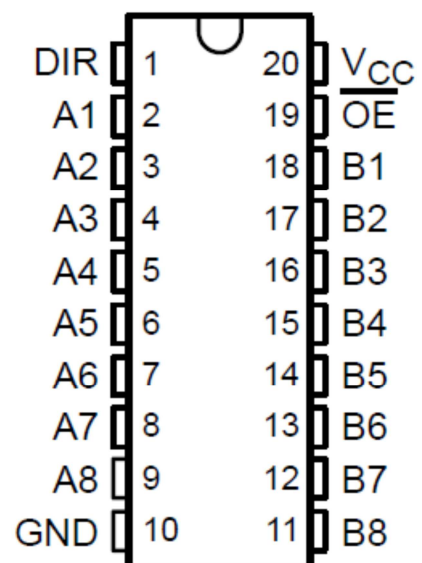
FUNCTION TABLE

INPUTS			OUTPUTS	
DATA	ENABLES			
P, Q	$\overline{G}, \overline{GT}$	$\overline{G2}$	$\overline{P = Q}$	$\overline{P > Q}$
P = Q	L	L	L	H
P > Q	L	L	H	L
P < Q	L	L	H	H
X	H	H	H	H

H = HIGH Level, L = LOW Level, X = Irrelevant

11.1.2 74LS245

FUNCTION TABLE		
INPUTS		OPERATION
\overline{OE}	DIR	
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation



11.2 Datenblatt RS232-Pegelwandler:

11.2.1 MAX232

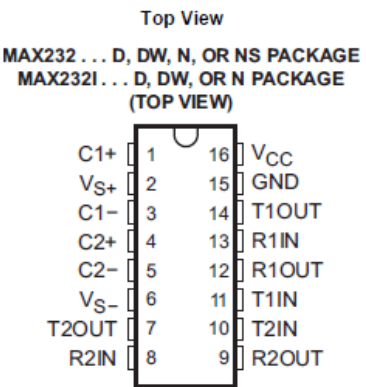


MAX232, MAX232I

www.ti.com

SLLS047M –FEBRUARY 1989–REVISED NOVEMBER 2014

6 Pin Configuration and Functions



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
C1+	1	—	Positive lead of C1 capacitor
VS+	2	O	Positive charge pump output for storage capacitor only
C1-	3	—	Negative lead of C1 capacitor
C2+	4	—	Positive lead of C2 capacitor
C2-	5	—	Negative lead of C2 capacitor
VS-	6	O	Negative charge pump output for storage capacitor only
T2OUT, T1OUT	7, 14	O	RS232 line data output (to remote RS232 system)
R2IN, R1IN	8, 13	I	RS232 line data input (from remote RS232 system)
R2OUT, R1OUT	9, 12	O	Logic data output (to UART)
T2IN, T1IN	10, 11	I	Logic data input (from UART)
GND	15	—	Ground
VCC	16	—	Supply Voltage, Connect to external 5V power supply

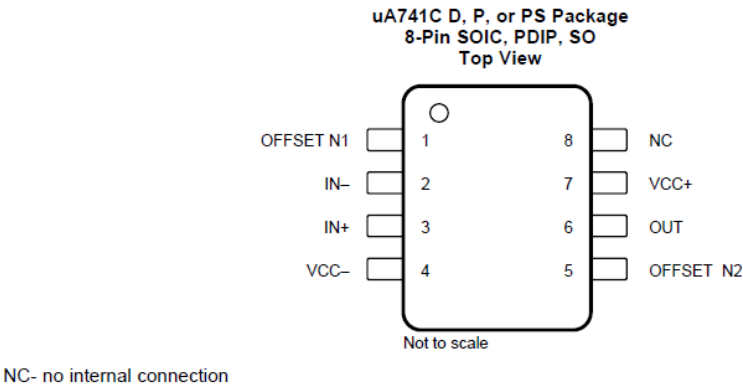
11.3 Datenblatt Operationsverstärker:

11.3.1 µA 741-P



uA741
SLOS094G – NOVEMBER 1970–REVISED JANUARY 2018 www.ti.com

5 Pin Configurations and Functions



PIN		I/O	DESCRIPTION
NAME	NO.		
IN+	3	I	Noninverting input
IN–	2	I	Inverting input
NC	8	—	No internal connection
OFFSET N1	1	I	External input offset voltage adjustment
OFFSET N2	5	I	External input offset voltage adjustment
OUT	6	O	Output
VCC+	7	—	Positive supply
VCC–	4	—	Negative supply

11.4 Verweis auf Datenblätter komplexer Bausteine und Spezifikationen

Baustein/Objekt	Funktion	Datenblatt
I2C-Bus	I2C-Bus-Spezifikation	UM10204.pdf
PCF 8584-P	I2C-Prozessor	PCF8584.pdf
Philips FM1216 ME / I H-3	Tuner	FM1216ME_MK3.pdf
TDA 8425	Sound	TDA8425.pdf
AT24C02	Speicher	AT24C02.pdf
TDA 3730B	RDS-Demodulator	TDA7330B.pdf
ATMEGA 168-P	Microcontroller	ATMEGA_168.pdf
RDS	RDS - Spezifikation	RDS_GRUNDLAGEN.pdf