Alphapoc 602R

letzte Änderung des Dokuments : 06.02.2019

Zur Zeit erlebt das POCSAG Paging eine neue Beliebtheit im Amateurfunk. Zum Einsatz kommen Pager z.B. des Typs Skyper welche aber auch schon grob 20 Jahre alt sind. Aber es gibt modernere Alternativen z.B. den hier näher untersuchten Alphapoc 602R.

Hintergrund der Untersuchung ist die für den Funkamateur unerlässliche Doku „was ist in dem Ding drin“ incl. der nötigen Infos falls repariert oder modifiziert werden muss. Damit nicht jeder seinen Pager aufmachen muss, opfere ich meinen und stelle alles was ich herausfinden kann zur Verfügung.

Aussenansichten

Das ist der Alphapoc 602R - diese Version ist mit PLL und läßt sich einfach im Menü oder durch Programmierung auf die 439,9875 MHz umstellen und damit direkt im DAPNET benutzen ohne weiteren Umbau.

8 verschiedene RIC Adressen (mit je 4 Unteradressen), viele Textspeicher stellen eine zeitgemäße Alternative zum Skyper dar. Einfache Einstellungen lassen sich durch das Menü umstellen, größere Dinge gehen mit dem Programmierkabel incl. der Software (GP2012).

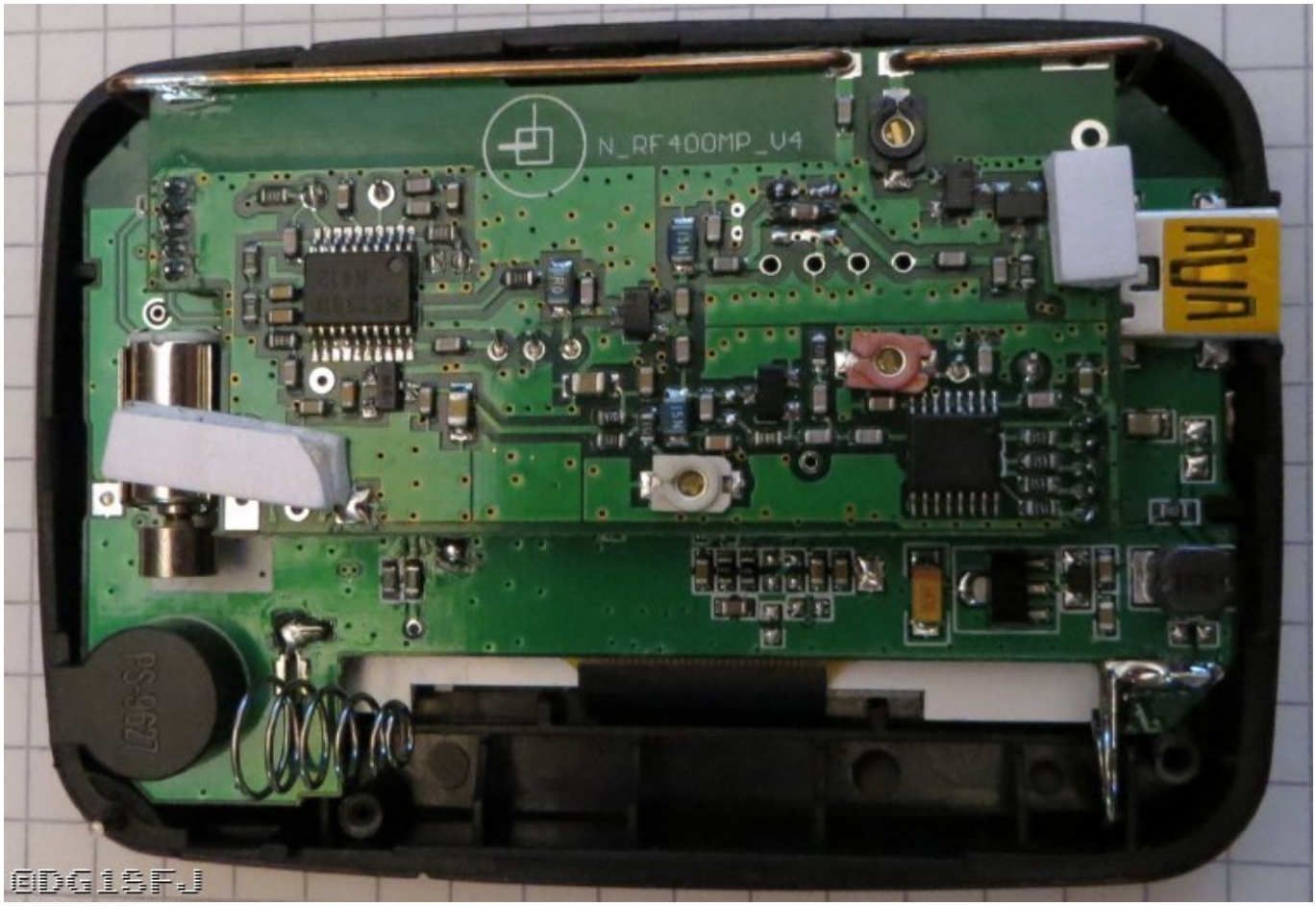
Weitere Besonderheit : Der Pager läuft mit einer Eneloop AAA Zelle bis zu 4 Wochen und läßt sich über die seitlche USB Buchse überall wieder aufladen.

Rückseite : Man erkennt die Herkunft aus China

Hauptplatine N_MB_VA

Im Batteriefach befinden sich 2 Schrauben. Nach dem entfernen läßt sich der Deckel abheben. Der Pager besteht aus 2 Teilen : Der Hauptplatine und der auswechselbaren (je nach QRG) Funkempfängerplatine.

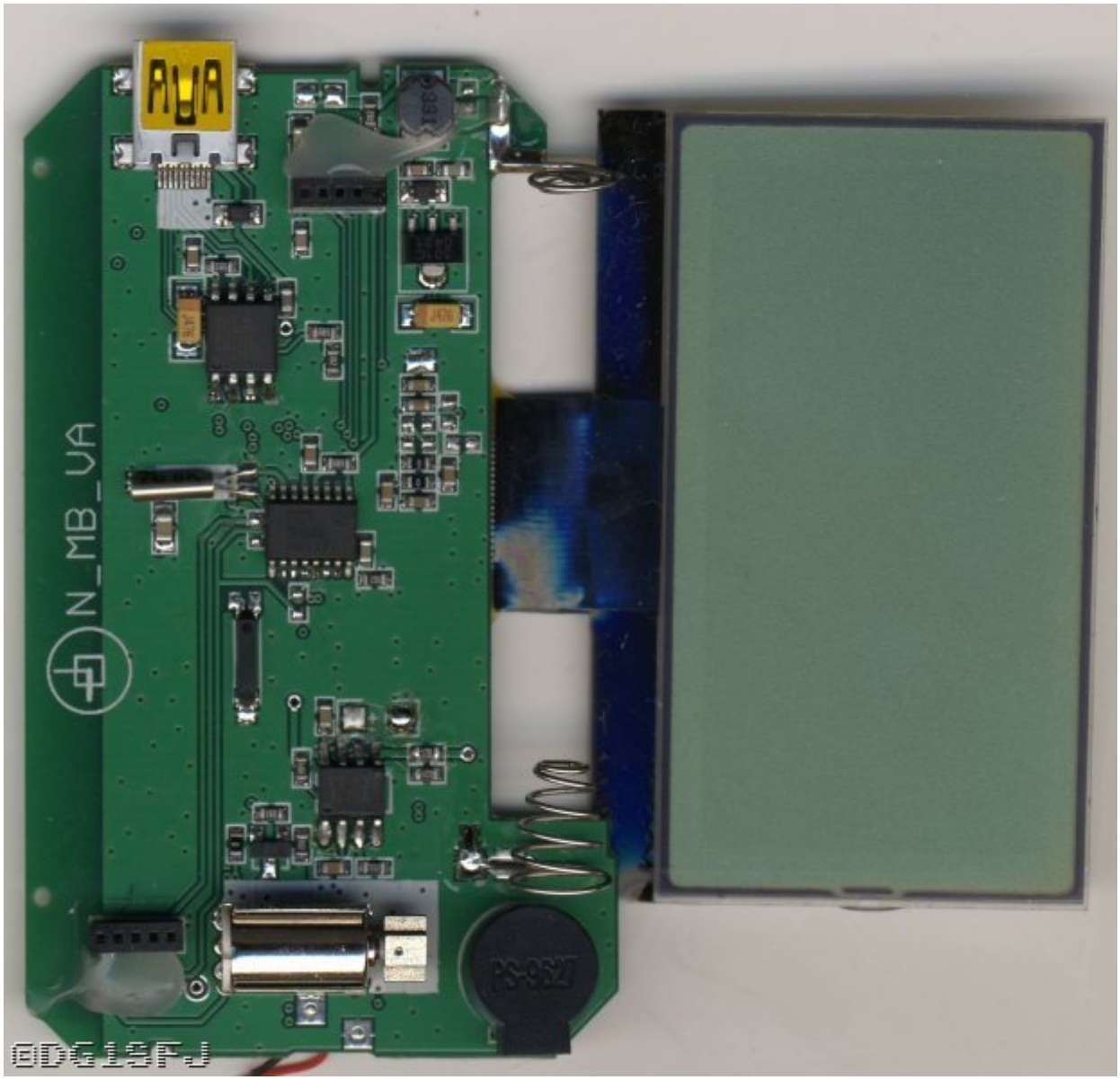
Man erkennt die Antenne im oberen Gehäuseteil, rechts die USB Buchse für Reprog und Laden sowie Piepser, Vibrationsmotor auf der linken Seite.



Vorder/Rückseite

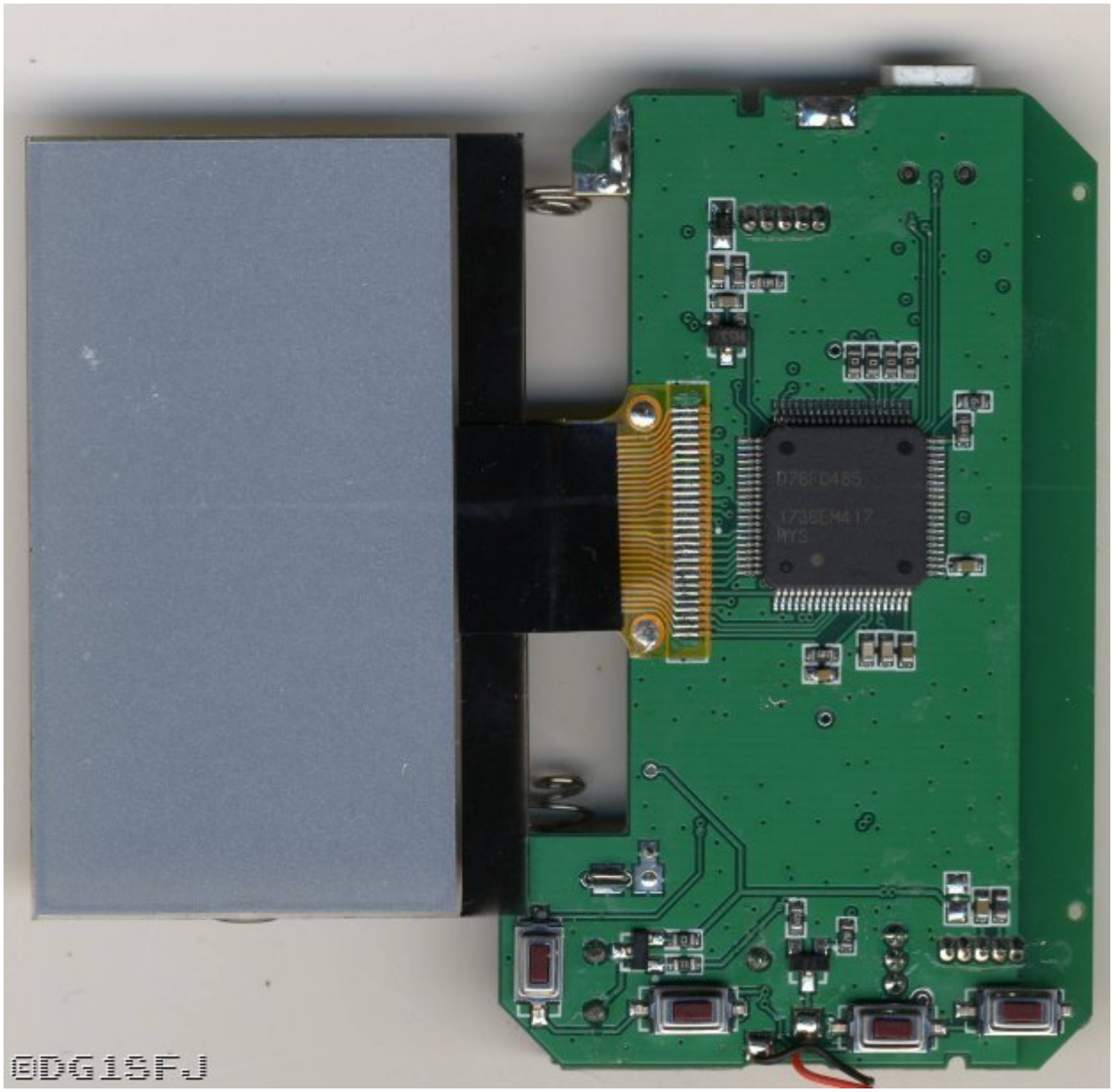
Vorderseite der Hauptplatine :

Hier ist die USB-Buchse, der Flash-Speicher, DCDC Wandler, POCSAG Decoder IC sowie Lade-IC, Vibrationsmotor und Piepser.



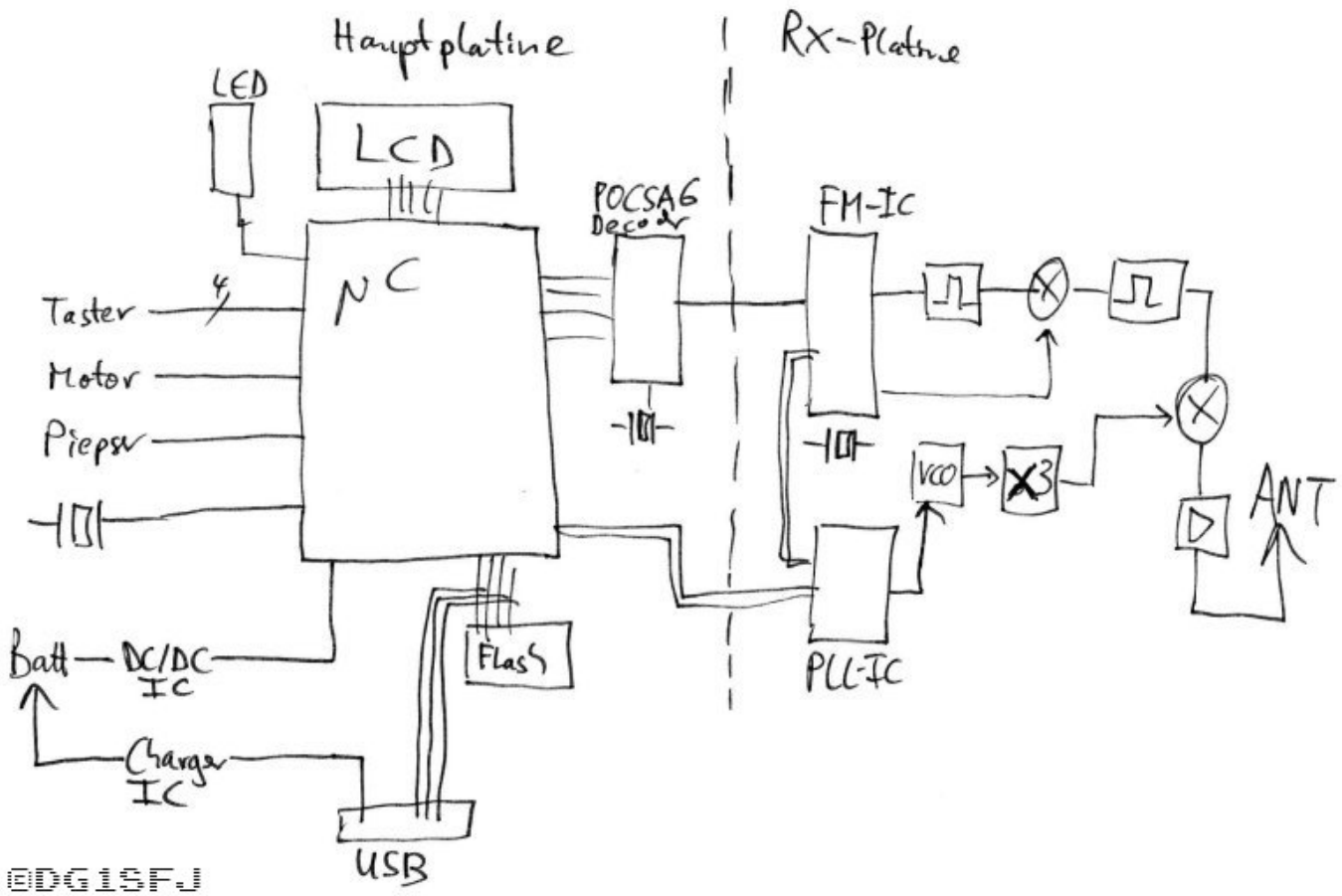
Rückseite der Hauptplatine :

Hier sitzt der Microcontroller von Renesas sowie die 4 Tasten und das Flex-Kabel zum Display.



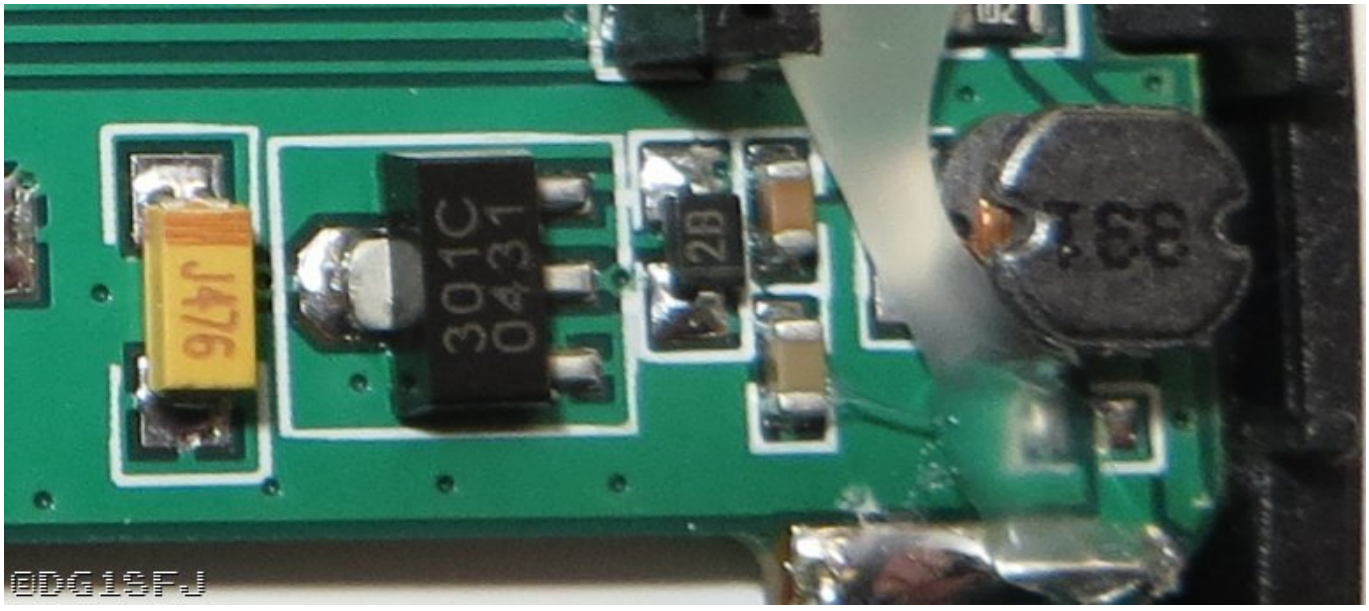
Blockschaltbild

Zur Orientierung hier das Blockschaltbild des Alphapoc 602 mit Haupt und RX Platine :

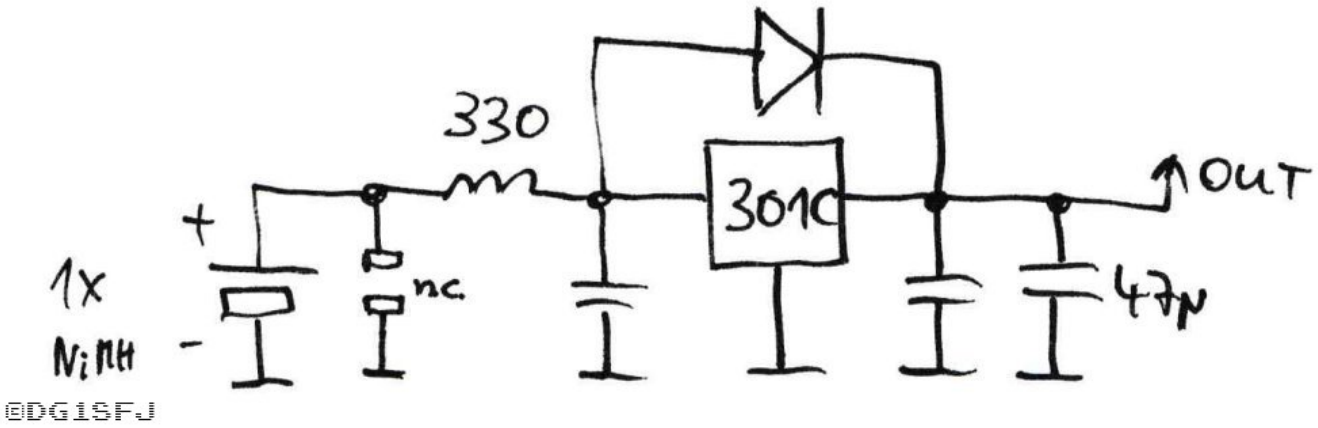


Stromversorgung

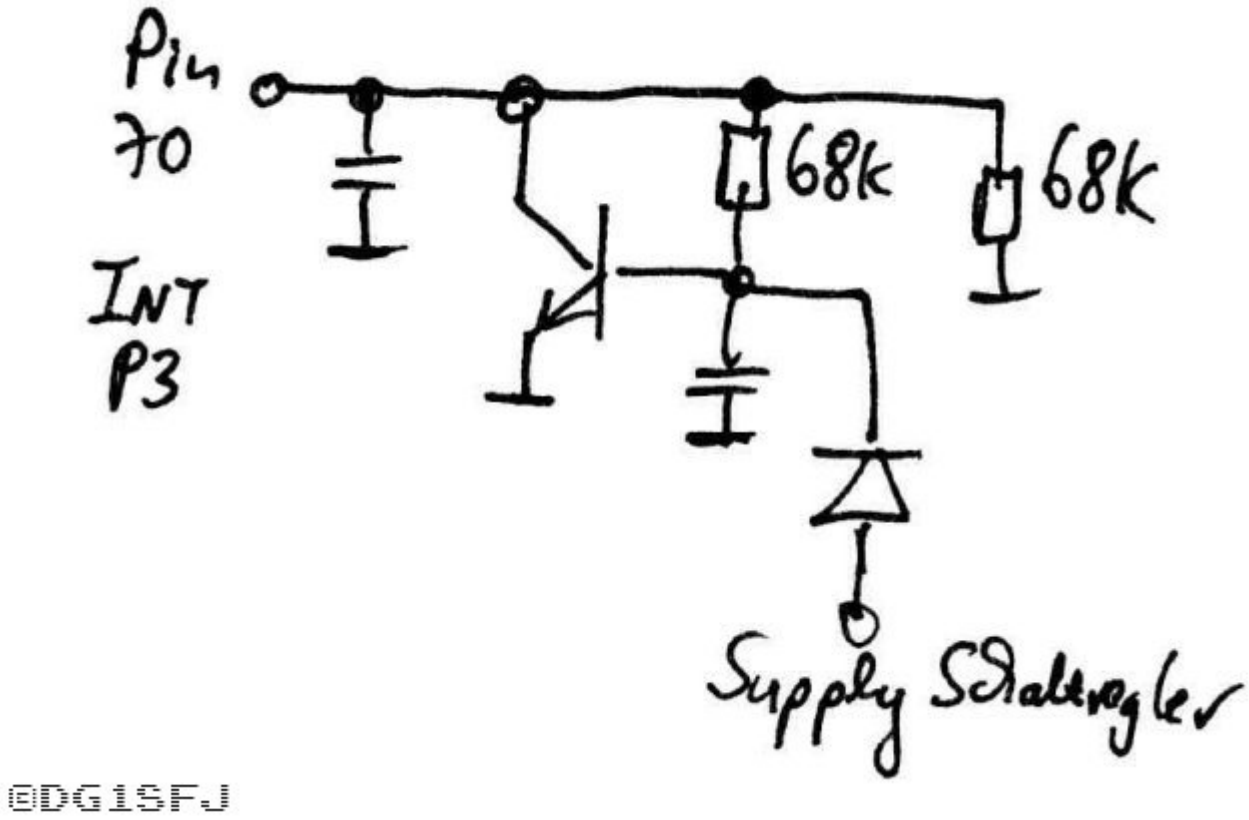
Ein DCDC Wandler im SOT-89 Gehäuse. Step Up Wandler der seine Frequenz an den Bedarf anpasst. Im Ruhezustand wird nur selten getaktet. Ob es sich um den originalen IC von RICOH oder eine Kopie handelt ist unklar.



Hier noch die Aussenbeschaltung :

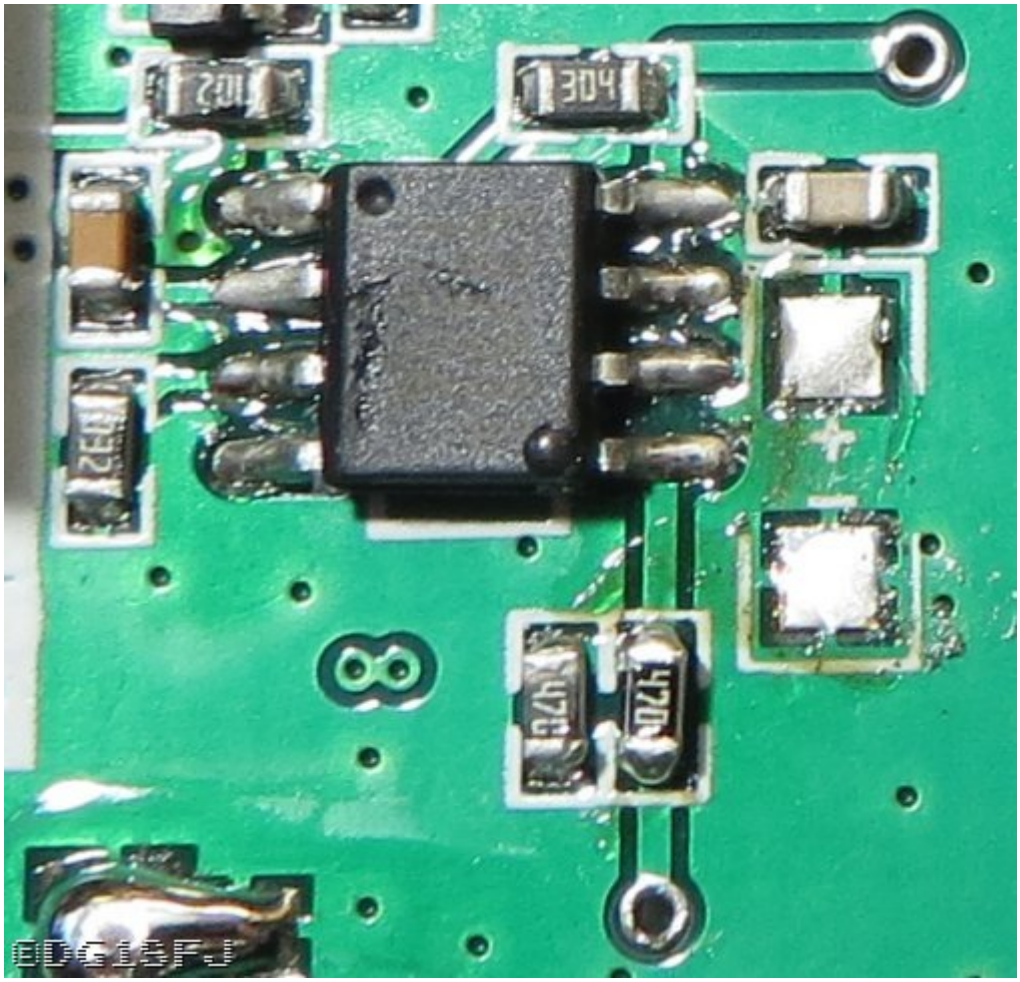


Vermutlich um Änderung oder Neu-Einlegen der Batterie zu erkennen ist eine kleine Schaltung zwischen Supply und einem Interrupt-Pin am Microcontroller angeschaltet - Zweck noch nicht klar :

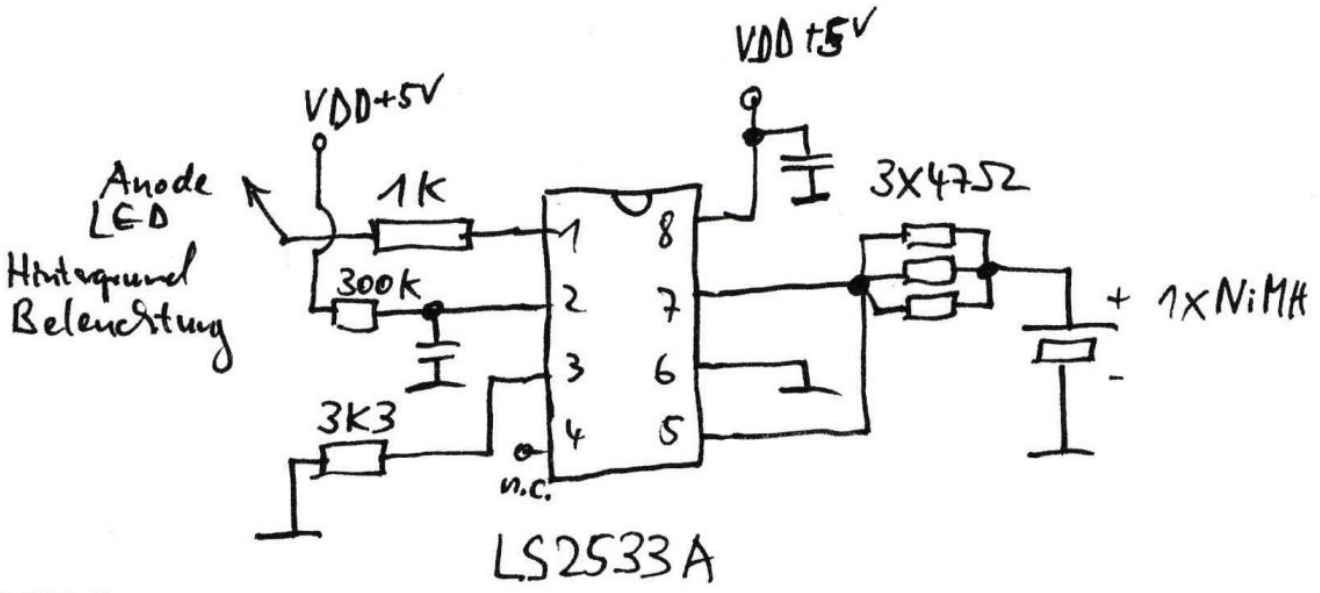


Ladeelektronik

Das Lade-IC ist ein Linkas LS2533A, NiMH Charger IC der sich aus 5V versorgt via USB. Kann auch Trickle Charge, lädt also auch nach wenn die Batterie voll ist um die Selbstentladung auszugleichen.



Hier noch die Aussenbeschaltung. Besonderheit : Das Lade-IC zeigt den Zustand an einem Pin an (1), dieser wird verwendet damit die Hintergrundbeleuchtung leicht leuchtet solange geladen wird. Wenn Laden fertig geht auch das Licht aus.



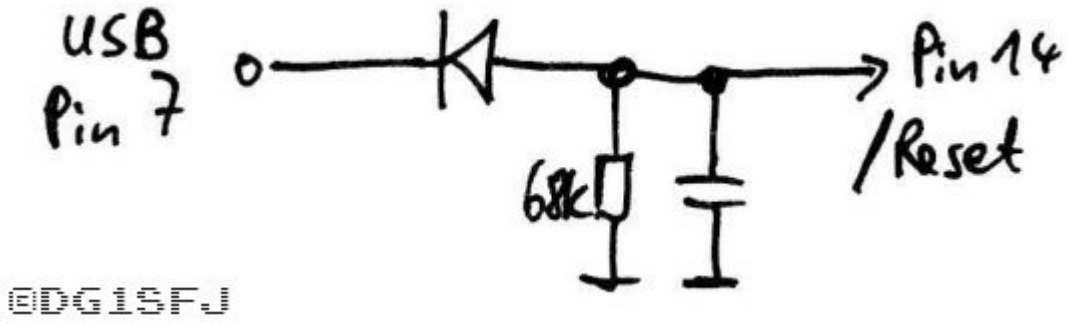
©DG1SFJ

USB Anschluss

Als Besonderheit wartet hier eine Mini USB Buchse auf uns mit 10 statt 5 Pins. Wird das Ladekabel eingesteckt werden nur Masse und +5V vom USB abgenommen. Beim Programmierkabel kommen dann auch die anderen Pins zum Einsatz.

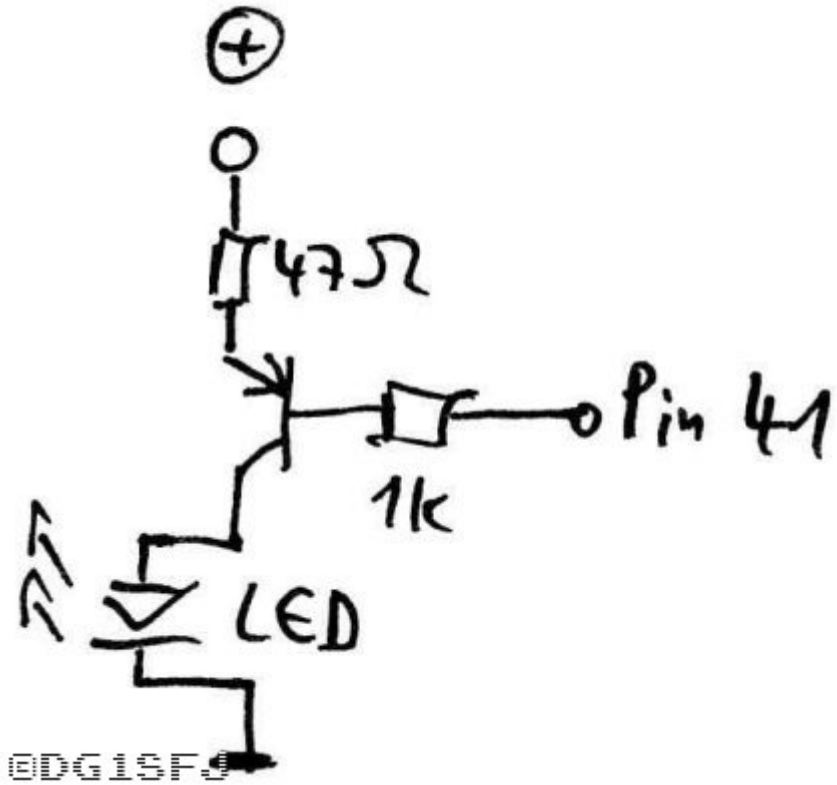
```
01 – GND
02 – zum SPI-Flash Pin 2 – Serial Data In
03 – zum SPI-Flash Pin 50 – Serial Data Out
04 – TXD-Pin des Renesas – Pin 36 : P112/TXD/SEG18
05 – +5V vom USB
06 – Flashmode-Pin des Renesas – Pin 17
07 – /Reset des Renesas Pin 14 (68k+C parallel, dann Diode zum Pin)
08 – RXD-Pin des Renesas – Pin 35 : P113/RXD/SEG19
09 – zum SPI-Flash Pin 6 – Serial CLK
10 – zum SPI-Flash Pin 1 – /ChipSelect
```

Zur Erklärung noch die Beschaltung der Reset Leitung :



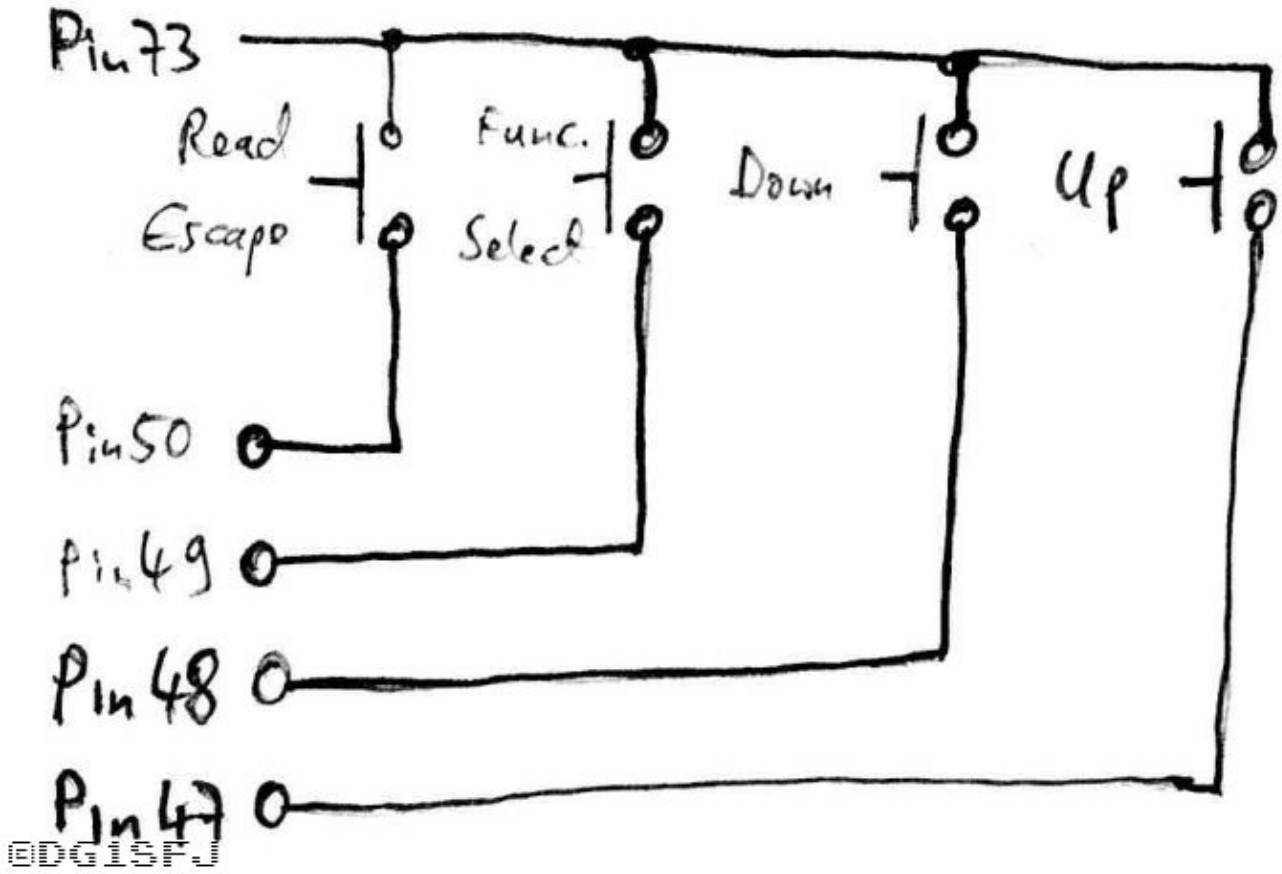
Display-Beleuchtung

Eine Blaue LED wird über eine Treiberstufe aktiviert. Geht an Pin 41 des Renesas-Controllers.



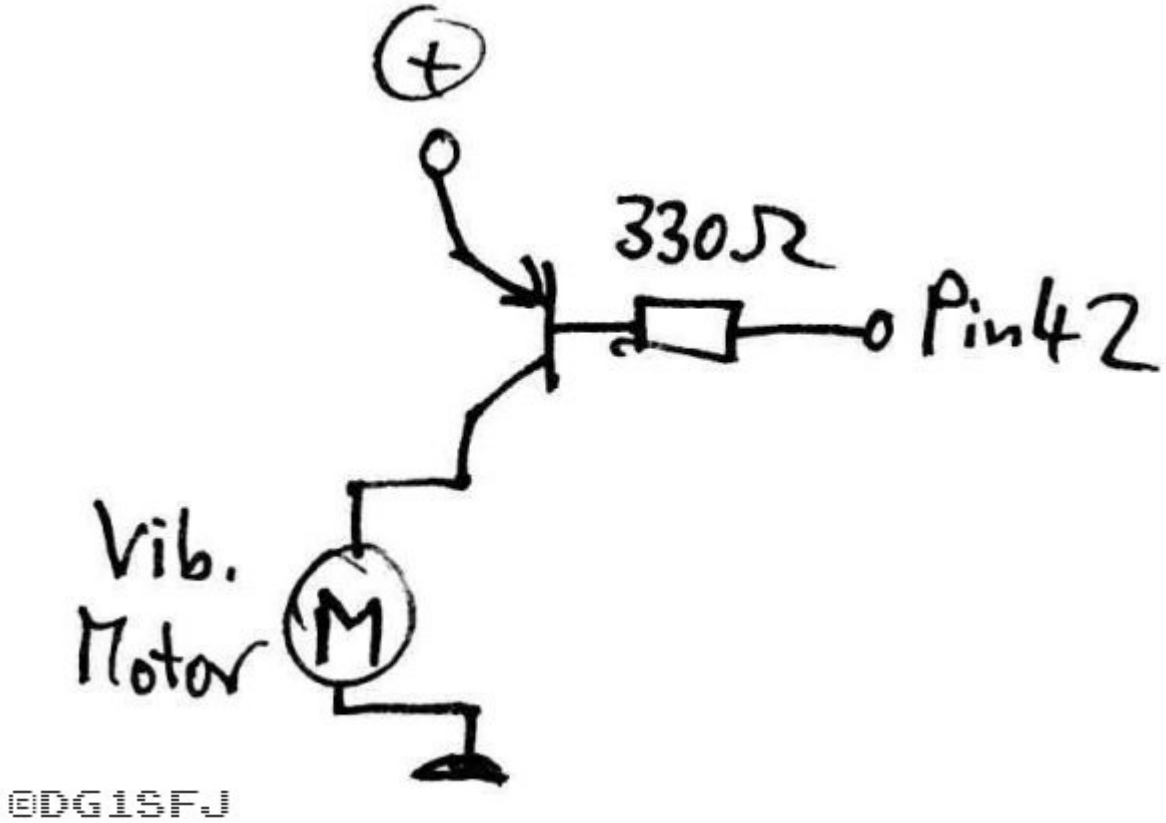
Taster

Pin 73 des Controllers treibt vermutlich die Taster, auslesen dann über die 4 Rückleitungen.



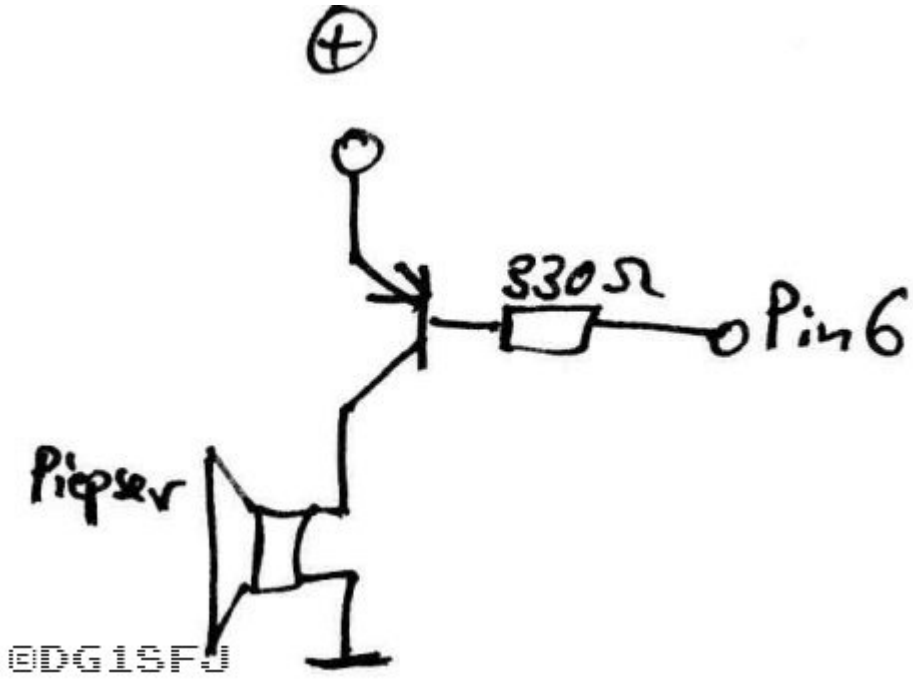
Vibrationsmotor

Ansteuerung des Motors über eine Treiberstufe. Steuerung via Pin 42 des Renesas.



Piepser

Ansteuerung über Treiberstufe von Pin 6 des Renesas.



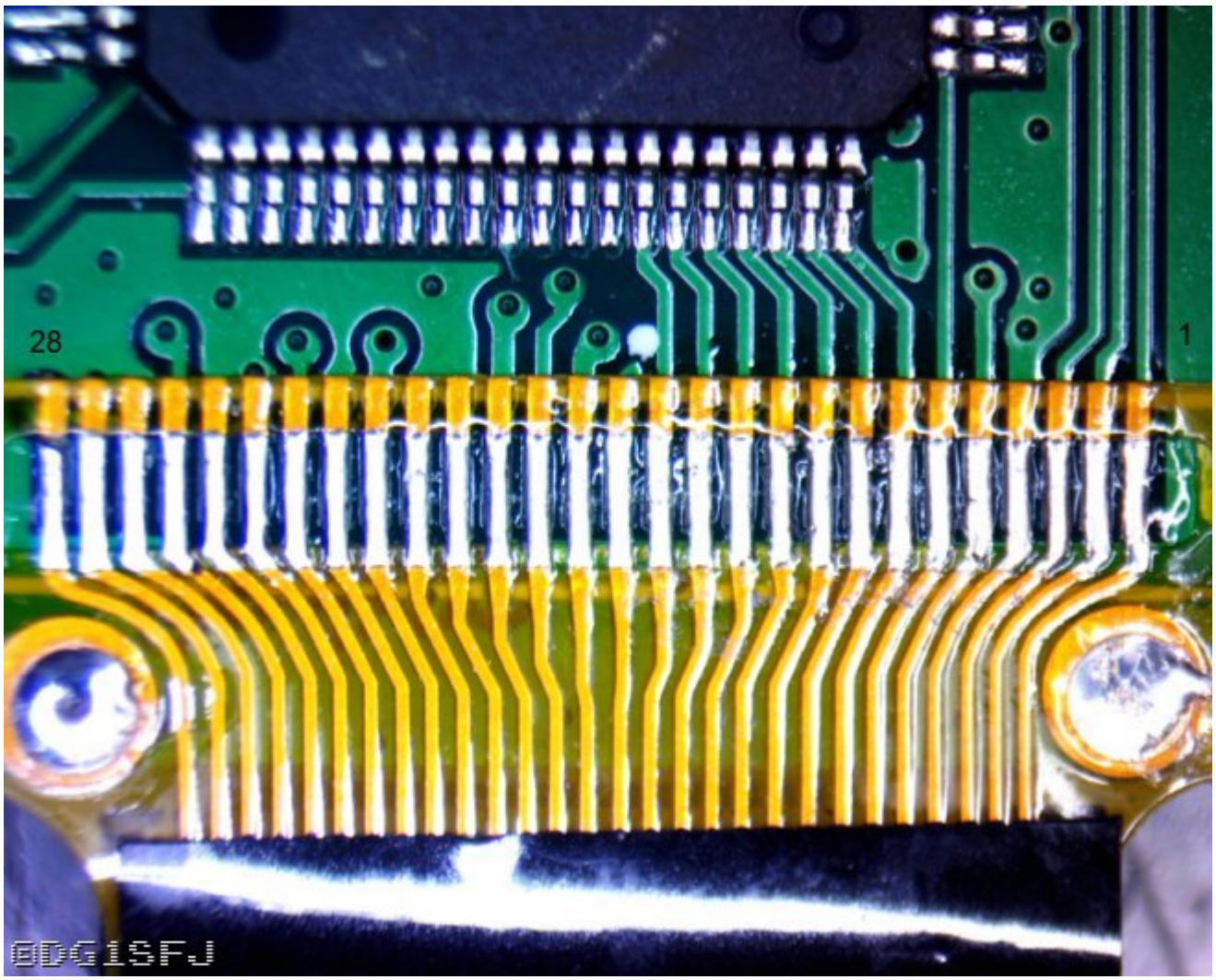
Display

Das Display ist grob 57x36mm groß und ist ein Grafik-Display mit 128×64 Punkten.

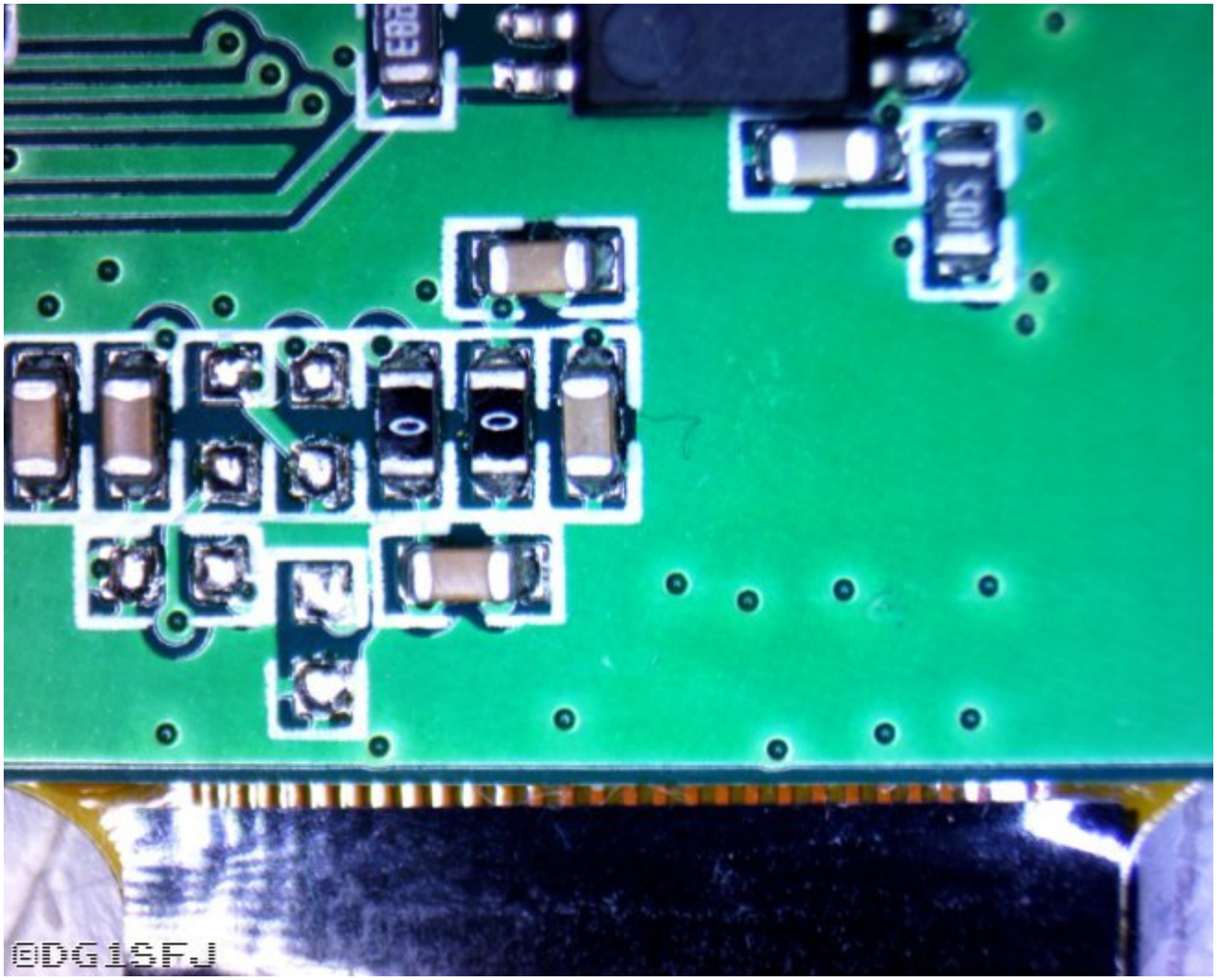
Hersteller unbekannt. Auf dem Display sind nur „DS600010B01“ und „A3/5“ zu finden. Der Controller ist als Chip-on-Glass (COG) vermutlich ein ST7565R.

Hintergrundbeleuchtung per LED an einer seitlichen Streuscheibe.

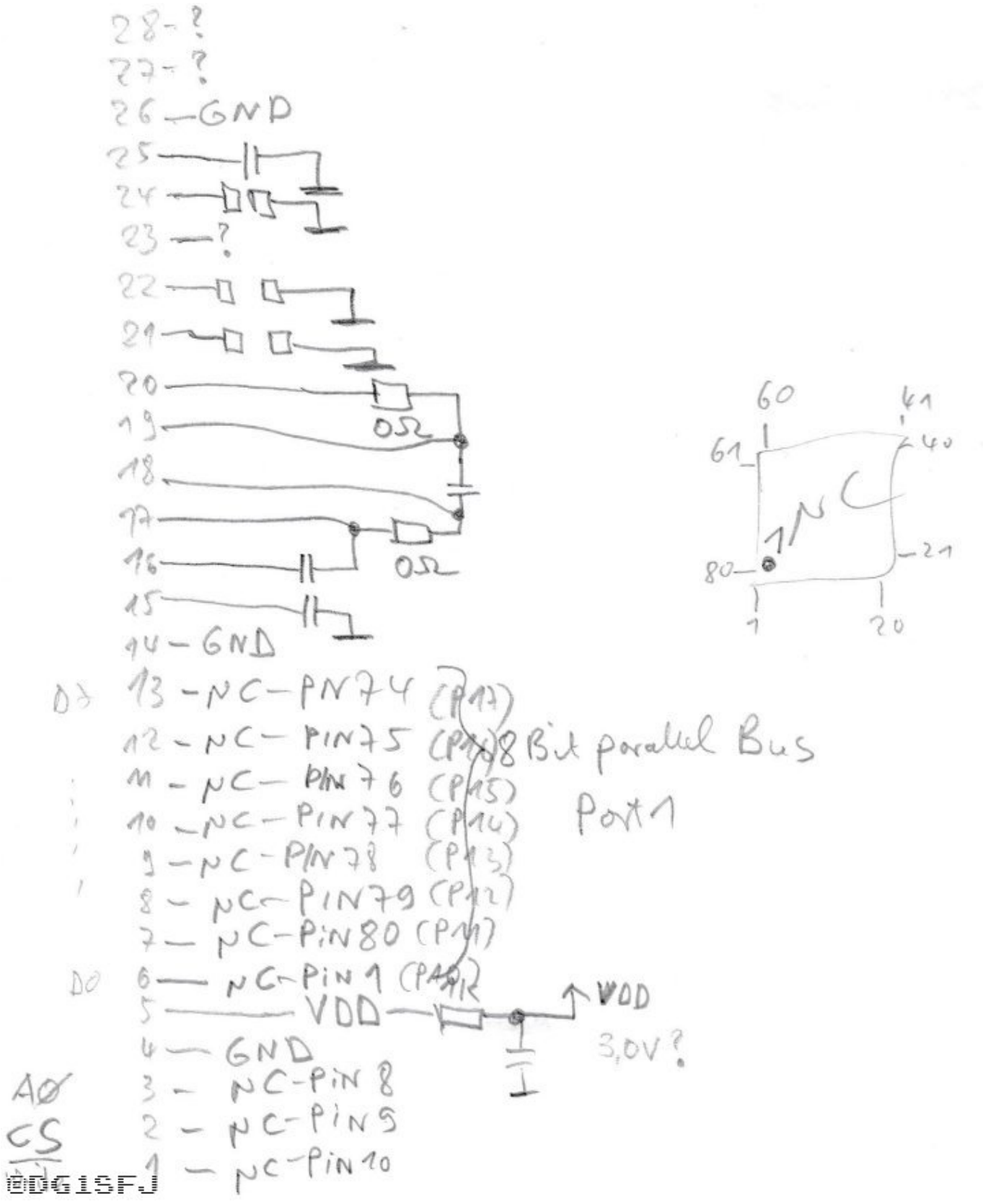
Vorder-Seite der Hauptplatine mit dem Folienkabel :



Rückseite der Hauptplatine - Achtung, Bild ist gespiegelt !



Die Beschaltung des Folienkabels ist ungefähr so :



Der Microcontroller steuert das Display über 8+3 Leitungen an :

```
LCD Pin 6 - uC Pin 1 - D0
LCD Pin 7 - uC Pin 80 - D1
LCD Pin 8 - uC Pin 79 - D2
LCD Pin 9 - uC Pin 78 - D3
LCD Pin 10 - uC Pin 77 - D4
```

```
LCD Pin 11 - uC Pin 76 - D5
LCD Pin 12 - uC Pin 75 - D6
LCD Pin 13 - uC Pin 74 - D7
LCD Pin 1 - uC Pin 10 -  Write Leitung - Datenübernahme bei steigender
Flanke
LCD Pin 2 - uC Pin 9 - Chip-Select Leitung "H" : Display selected
LCD Pin 3 -  uC Pin 8 - A0 - "H" : Display Data, "L" : Control Data
```

Der Renesas schickt 14 Kommandos als initialisierung beim ersten Einschalten los (dabei A0 auf Low) :

```
#01 = 1 0 1 0 0 0 0 0 = ADC select normal "1"
#02 = 1 1 0 0 1 0 0 0 = Common Output mode select "000"
#03 = 1 0 1 0 0 0 1 0 = LCD Bias Set "1"
#04 = 0 0 1 0 0 1 0 0 = Voltage regulator internal resistor ratio set "100"
#05 = 1 0 0 0 0 0 0 1 = Electronic Volumne Mode Set
#06 = 0 0 1 0 0 0 0 1 = Electronic Volumne register set "100001"
#07 = 0 0 0 1 1 1 0 0 = Column Adress Set Upper Bit "1100"
#08 = 0 0 1 0 1 1 0 0 = Power Controller Set
#09 = 0 0 1 0 1 1 1 0 = Power Controller Set
#10 = 0 0 1 0 1 1 1 1 = Power Controller Set
#11 = 1 0 1 0 1 1 1 1 = Display on/off "1" ON
#12 = 1 0 1 1 0 0 0 0 = Page Adress Set "0000"
#13 = 0 0 0 1 0 0 0 0 = Column Adress Set Upper Bit "0000"
#14 = 0 0 0 0 0 0 0 0 = Column Adress Set Lower Bit "0000"
```

Danach beginnt das Beschreiben des Display RAMs durch Übertragung von 131 ? Bytes mit dem Inhalt „00“, A0 leitung ist dabei dann auf High.

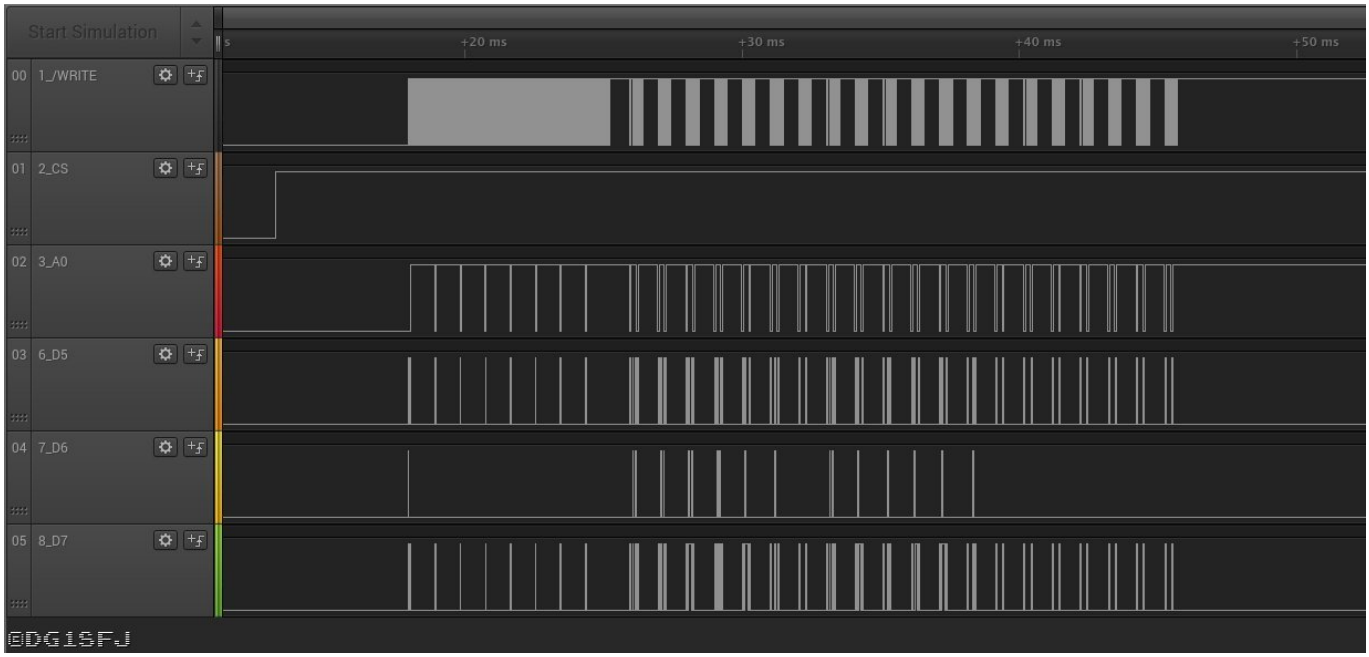
Wenn die Bytes geschrieben wurden wird die Page Adresse auf „0001“ erhöht, und wieder 131 ? Bytes „00“ übertragen.

Die Page Adresse geht dann bis „0111“ rauf. Nach 7,2ms sind alle Speicher mit „00“ überschrieben und das Display RAM leer.

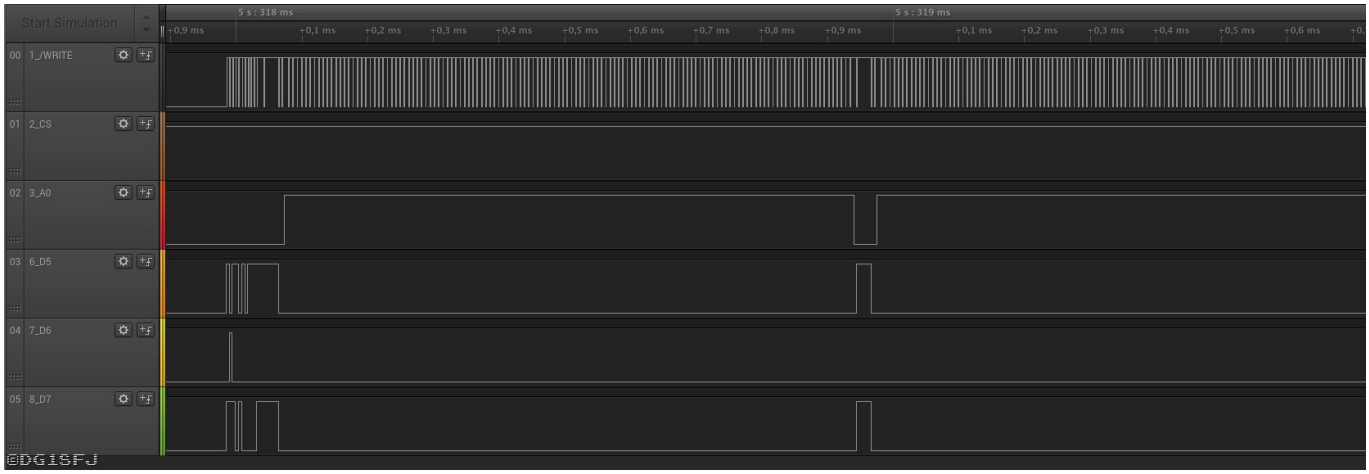
Abschließend wird die Einschaltmeldung des Pagers (aus dem Flash) gelesen und auf dem Display dargestellt.

Der gesamte Einschaltvorgang von Strom an (5s) → Display Init → Display darstellen → Display wieder abschalten (24s) :

Zoom auf die Initialisierungs-Phase (Init + Display löschen + Display Inhalt reinprogrammieren) :



Zoom auf Display Init :



Flash

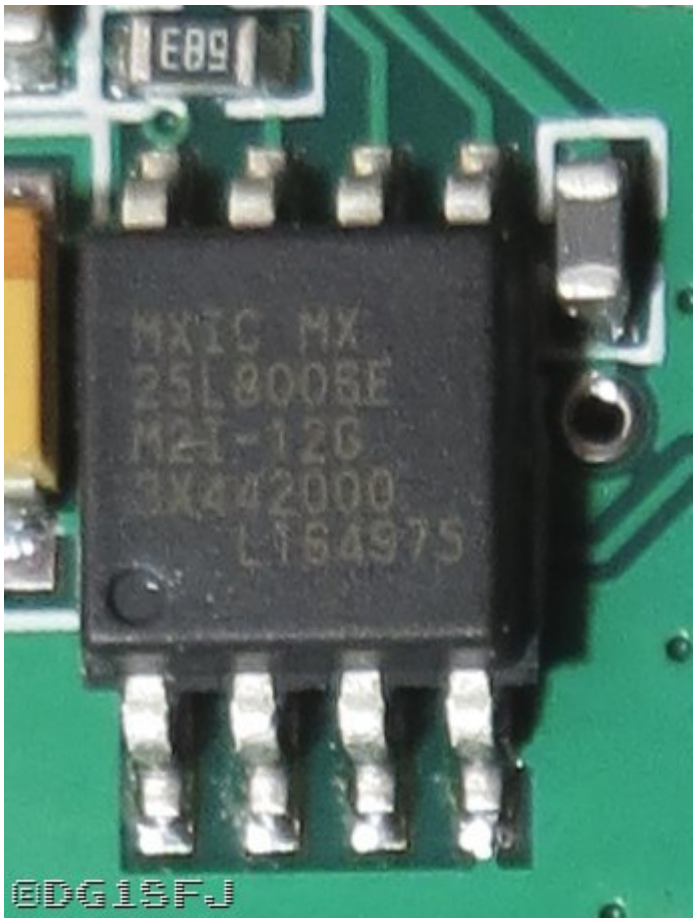
Macronix International, MX25L8006E

8Pin PDIP Gehäuse

8MBit CMOS Serial Flash

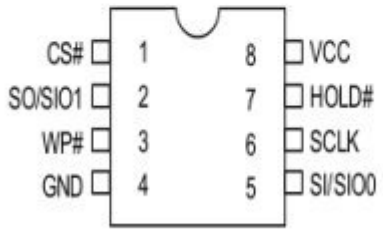
512Bit Secured OTP

SPI Interface Mode 0 und 3



MX25L8006E

8-PIN PDIP (300mil)



PIN DESCRIPTION

SYMBOL	DESCRIPTION
CS#	Chip Select
SI/SIO0	Serial Data Input (for 1 x I/O)/ Serial Data Input & Output (for Dual Output mode)
SO/SIO1	Serial Data Output (for 1 x I/O)/ Serial Data Output (for Dual Output mode)
SCLK	Clock Input
WP#	Write protection
HOLD#	Hold, to pause the device without deselecting the device
VCC	+ 3.3V Power Supply
GND	Ground

Auf der Leiterplatte sind noch 2 Widerstände bestückt :

```
Pin 3 : 68k Pullup nach VCC – für /WriteProtect

Pin 7 : 68k Pullup nach VCC – für /Hold

Pin 4 : Masse

Pin 8 : Supply mit 47uF Tantal
```

Nach dem Start des Pagers liest der Microcontroller einen Speicherblock ein. Dazu verwendet er zwei Befehle :

```


0x04 : WRDI (Write Disable)

0x0B 0x08 0x00 0x00 0xFF : FAST READ (Ab Adresse AD1=08, AD2=00, AD3=00,
```

```
0xFF steht für Dummy)
```

Danach wird solange inkrementell die nächste Speicherstelle ausgelesen bis CS# wieder auf High geht.

Startadresse „0x080000“, es werden 256 Byte eingelesen.

Die 8MBit = 1MByte Speicher sind nur sehr spärlich genutzt. Aufbau sieht wie folgt aus :

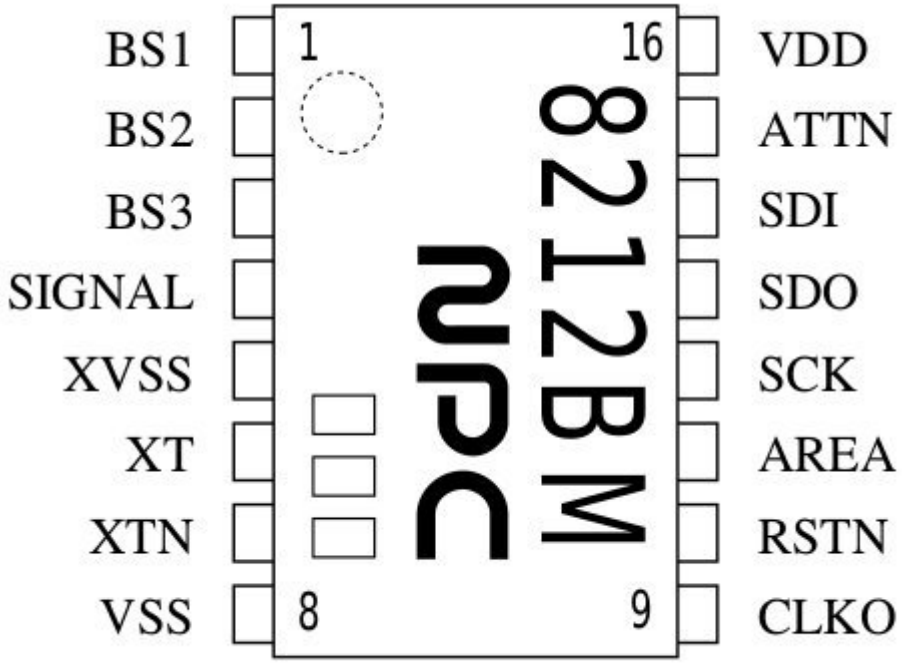
```
$00000 - $7FFFF : leer

$80000 - $800FF : Konfiguration des Pagers, wird beim Start ausgelesen, auch
ASCII Startbotschaft, Frequenz u.s.w.

$80100 - $80FFF : leer

$81000 - $81FFF : RIC Konfiguration (mit ASCII Namen)

$82000 - $82FFF : leer

$83000 - $830FF : Backup der Konfiguration von $80000 ?

$83100 - $87FFF : leer

$88000 - $882FF : unbekannt

$88300 - $88FFF : leer

$89000 - $892FF : unbekannt

$89300 - $89FFF : leer

$8A000 - $8A2FF : unbekannt

$8A300 - $8AFFF : leer

$8B000 - $8B2FF : unbekannt

$8B300 - $8BFFF : leer

$8C000 - $8C2FF : unbekannt

$8C300 - $8CFFF : leer

$8D000 - $8D2FF : unbekannt

$8D300 - $8DFFF : leer

$8E000 - $8E2FF : unbekannt
```

```
$8E300 - $8EFFF : leer

$8F000 - $8F2FF : unbekannt

$8E300 - $8EFFF : leer

$90000 - $900FF : Message + Config

$91000 - $910FF : Message + Config

$92000 - $922FF : Message + Config

$93000 - $932FF : Message + Config

$94000 - $942FF : Message + Config

$95000 - $952FF : Message + Config

$96000 - $962FF : Message + Config

$97000 - $972FF : Message + Config

$98000 - $982FF : Message + Config

$99000 - $992FF : Message + Config

$9A000 - $9A2FF : Message + Config

$9B000 - $9B2FF : Message + Config

$9C000 - $9C2FF : Message + Config

$9D000 - $9D2FF : Message + Config

$9E000 - $9E2FF : Message + Config

$9F000 - $9F2FF : Message + Config

$A0000 - $A02FF : Message + Config

$A1000 - $A12FF : Message + Config

$A2000 - $A22FF : Message + Config


Dann leer bis Speicherende $FFFFF
```

POCSAG Dekoder

Das Herz des Pagers ist der SM8212B ein „POCSAG Decoder for Multiframe Pagers“ der Firma NPC Nippon Precision Circuits.

Auf der Webseite ist der IC als „Mature Product to be Discontinued“ seit 1.20.2011 aufgelistet. Käuflich kann dieser IC noch über den amerikanischen Distributor VCAMERICA erworben werden.

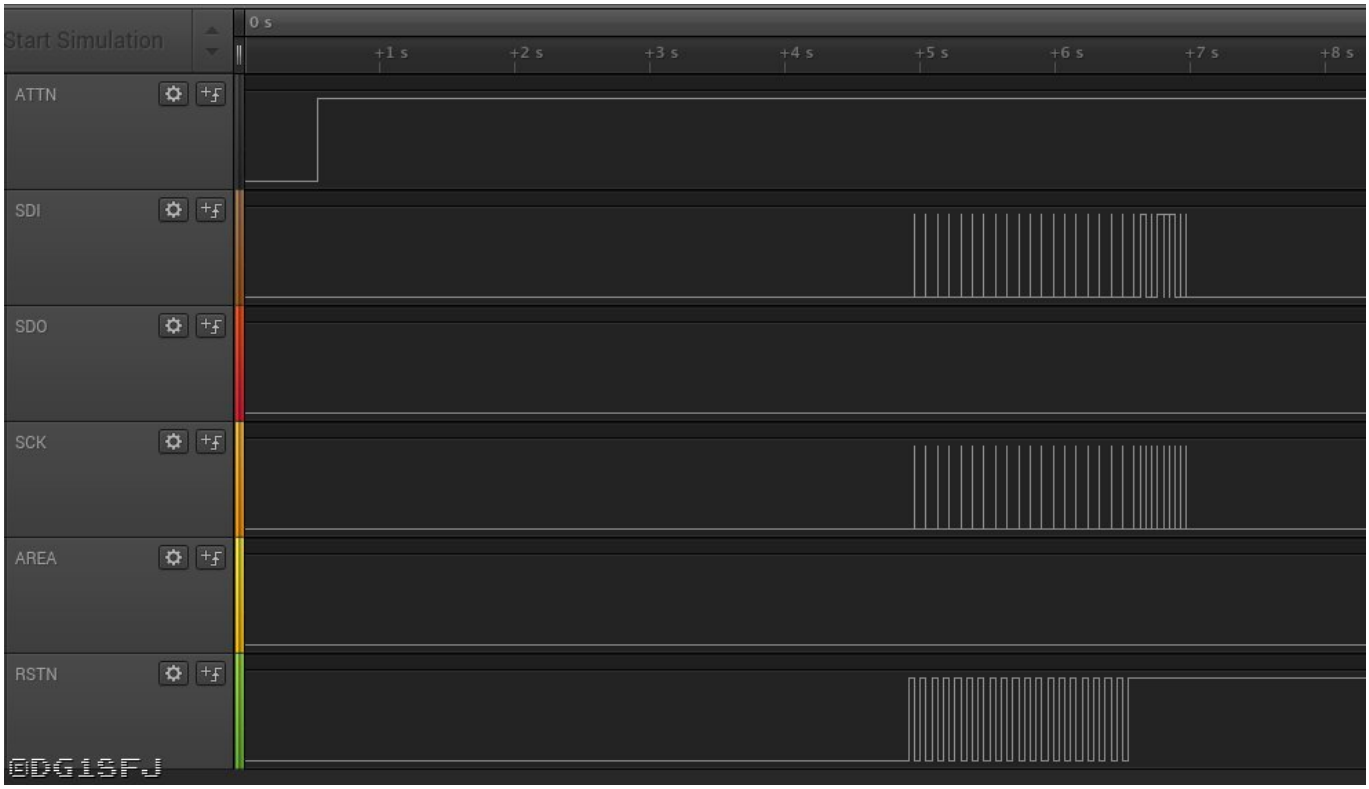


©DG1SFJ

Pin number	Pin name	i/o	Description
1	BS1	o	RF control main output signal
2	BS2	o	RF DC-level adjustment signal
3	BS3	o	PLL setup signal
4	SIGNAL	i	NRZ signal input pin
5	XVSS	-	Crystal oscillator ground. Capacitor connected between XVSS and VDD
6	XT	i	Oscillator input pin
7	XTN	o	Oscillator output pin
8	VSS	-	Ground
9	CLKO	o	76.8 or 38.4 kHz clock output
10	RSTN	i	Hardware clear (reset)
11	AREA	o	Sync code detection output (HIGH for minimum 1 sec. on detection)
12	SCK	i	CPU-to-decoder data transfer sync clock
13	SDO	o	Status and received data output to CPU
14	SDI	i	Data input from CPU (including ID data)
15	ATTN	o	Interrupt detect signal output pin (Ready for data transmission when LOW)
16	VDD	-	Supply voltage

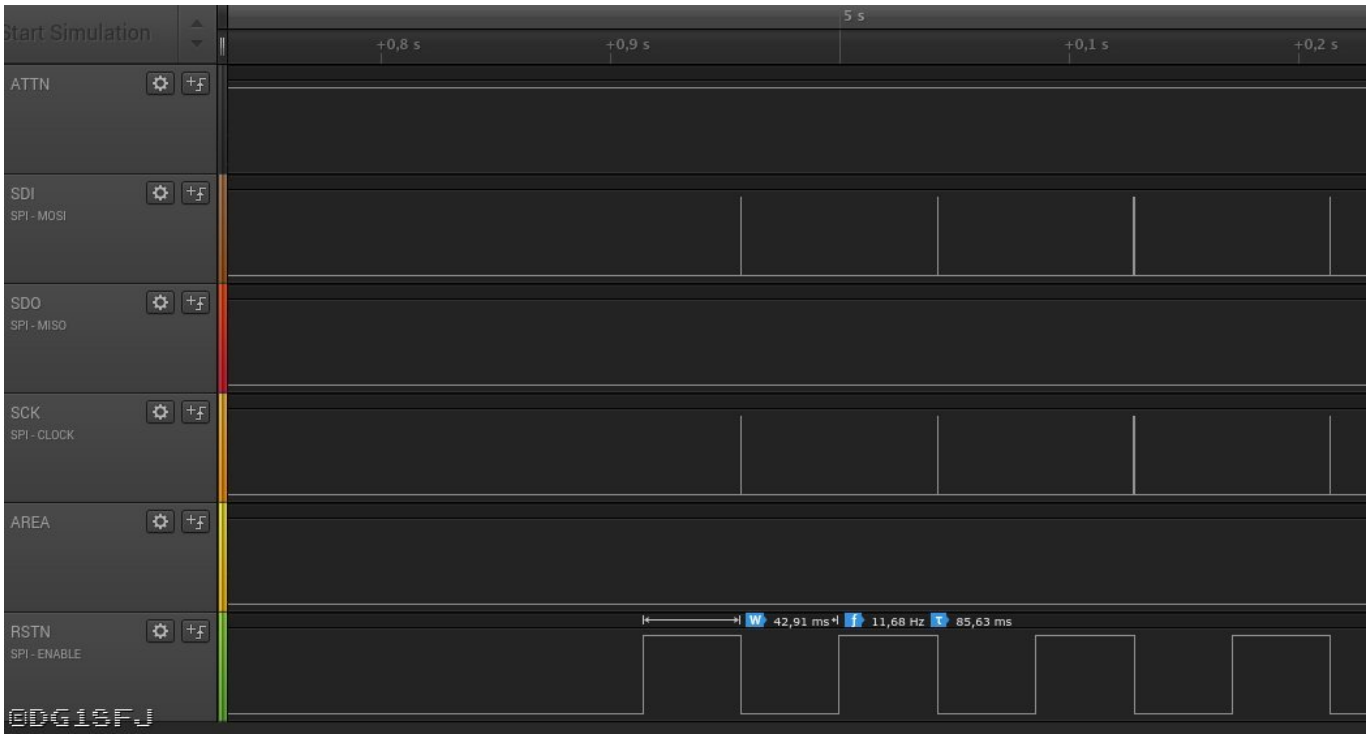
©DG1SFJ i = Input, o = Output

Nach dem einlegen der Batterie wird der Dekoder initial programmiert :

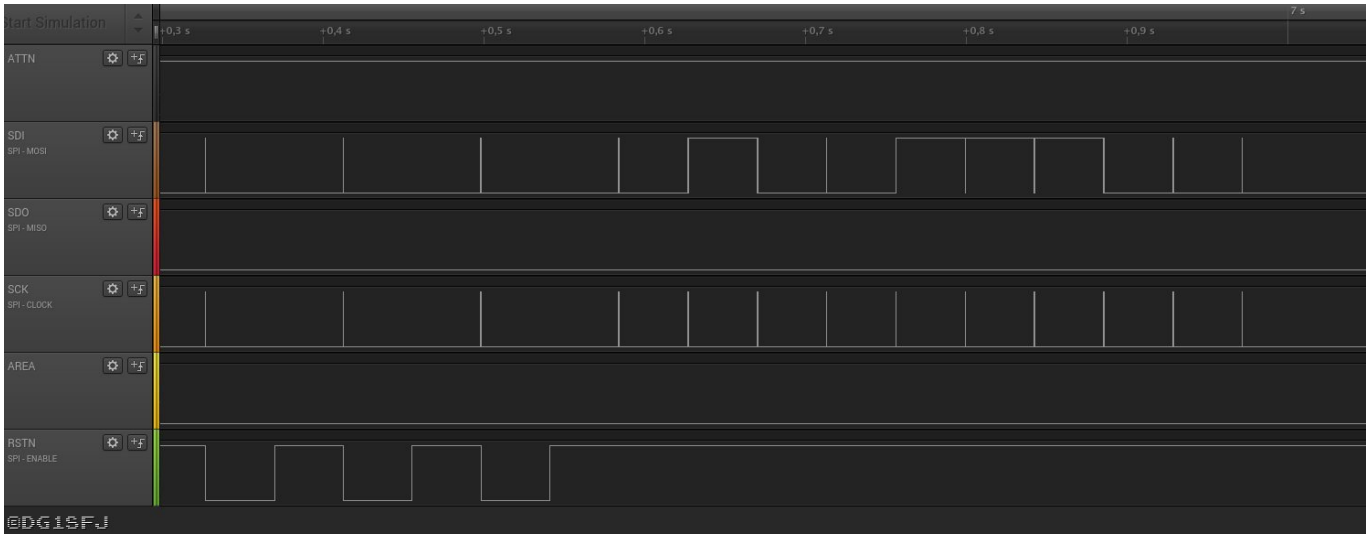


Ablauf :

19x werden 32 Clock Pulse ausgegeben und auf dem SDI das Wort 0x8100. Nach jedem Wort wird dann RSTN wieder auf Low gelegt. Zweck unbekannt.



Danach scheint die Initialisierung zu beginnen mit „echter“ Kommunikation.



Die Blöcke kommen im Abstand von 42ms. Die Clockleitung ist 4,4us High bei einer Wiederholrate von 11,16us.

Übertragen werden :

```
0x81 0x00 0x41 0x08 0x02 0x80

0x61 0x0F 0xDE 0x13 (01100001 00001111 11011110 00010011)

0x61 0x30 0x00 0x7E (01100001 00110000 00000000 01111110)

0x61 0x40 0x00 0x82 (01100001 01000000 00000000 10000010)

0x60 0x6C 0x00 0x83 (01100000 01101100 00000000 10000011)

0x61 0x80 0x00 0x85

0x61 0xA4 0x00 0x85

0x60 0xC0 0x00 0x84

0x60 0xE4 0x00 0x7E

0x82 Einzelbyte
```

Aktiviert waren zu dieser Zeit im Pager :

```
A: [ on] 2027675 (11110 11110000 10011011)

B: [ on] 0001012 (00000011 11110100)

C: [ on] 0001040 (00000100 00010000)

D: [off] 0001051 (00000100 00011011)

E: [ on] 0001064 (00000100 00101000)
```

```
F: [ on] 0001065

G: [off] 0001056

H: [off] 0001009
```

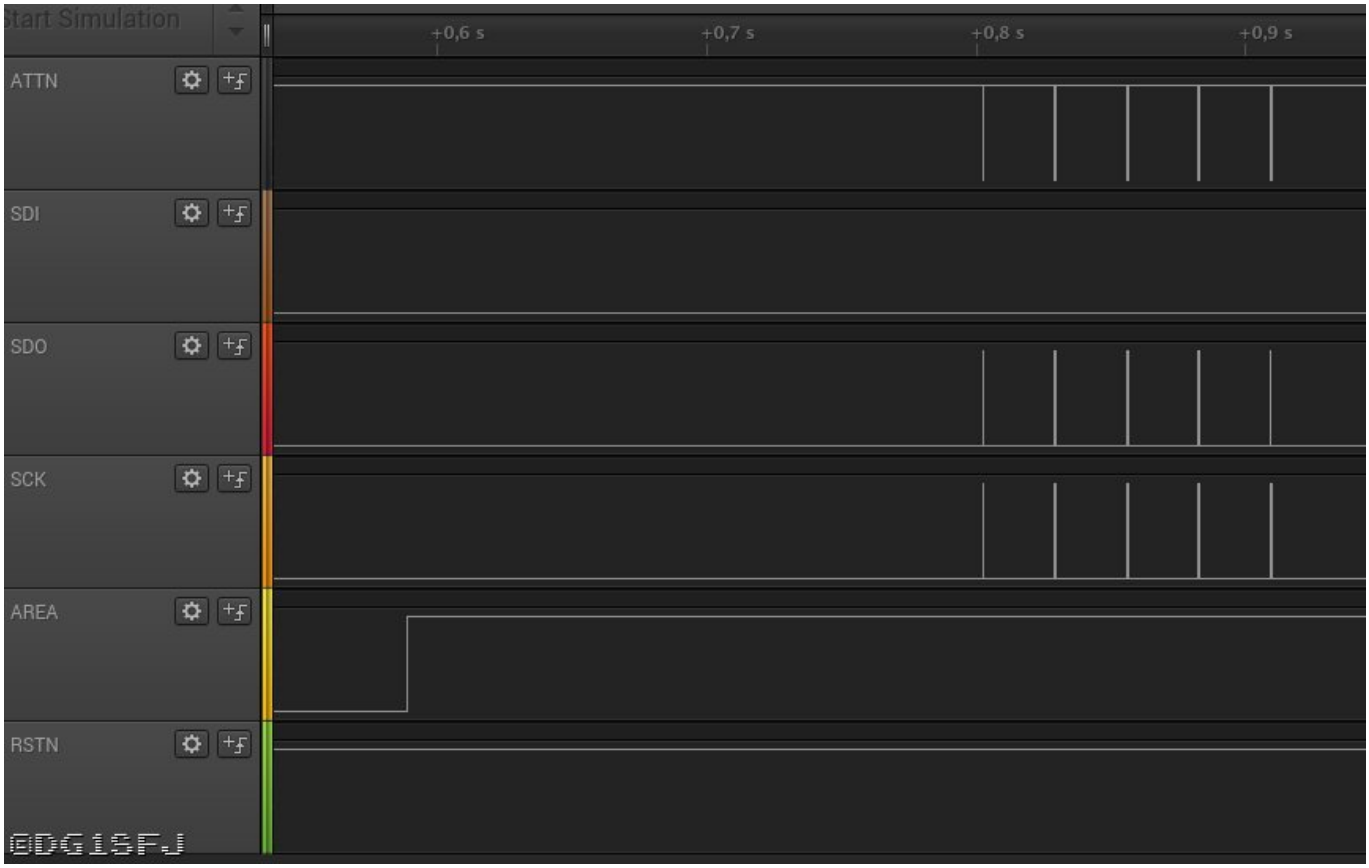
```
0x81 0x00 0x41 0x08 0x02 0x80 :

0x81 0x00 : Read Kommando für Flags/Daten

0x41 0x08 0x02 0x80 : Parameter Set Flag Kommando

01000001 00001000 00000010 10000000

1 BS2 Option=1 : mid-receive adjustment mode

0 End of Message Detection=0

0 CLKO=76,8kHz

0 KILL CLKO=CLKO Output enabled

01 Bit Rate Set=01=1200bps

0 Signal Polarity=normal

00 0000 PL5..0=Receive time Setup BS3

0010 10 RF5..0=Receive time Setup BS1

0 = Filter=Filter OFF

00 = Filter Strength=Filter Strength 1

000 = Rate error detection Standard for Preamble = Count=1
```

```
Das Adress Initialisierungskommando sieht dann wie folgt aus :

0x61 0x0F 0xDE 0x13 (01100001 00001111 11011110 00010011)

01100001 00001111 11011110 00010011

0110000 Adress Set Command

1 Address Enable. Wenn auf 1 dann sind bits 9..32 valid

000 Addresse Call Sign Flag („A..H“)

011 Address Frame Assign Bits (LSBs der Adresse)
```

```
11 11011110 00010011 MSBs der Adresse

Beispiel für die Adress-Dekodierung :

dez2027675 = bin11110111110000010011011

Die unteren 3 Bits werden abgeschnitten und als Frame-Nummer genommen.

Damit : Framenummer 011 und Rest-Adresse ist dann 111101111000010011



Das ganze wird für die 8 Adresse festgelegt, die der Dekoder verarbeiten
kann.



Abschluss :

0x82 : Decoder Set Command Transfer

10000010

10 0 : Decoder Set Command

0 : Break command

0 : Back-up mode command

0 : Write command

0 : BS-test mode

1 : Invoke the Start Mode Operation
```

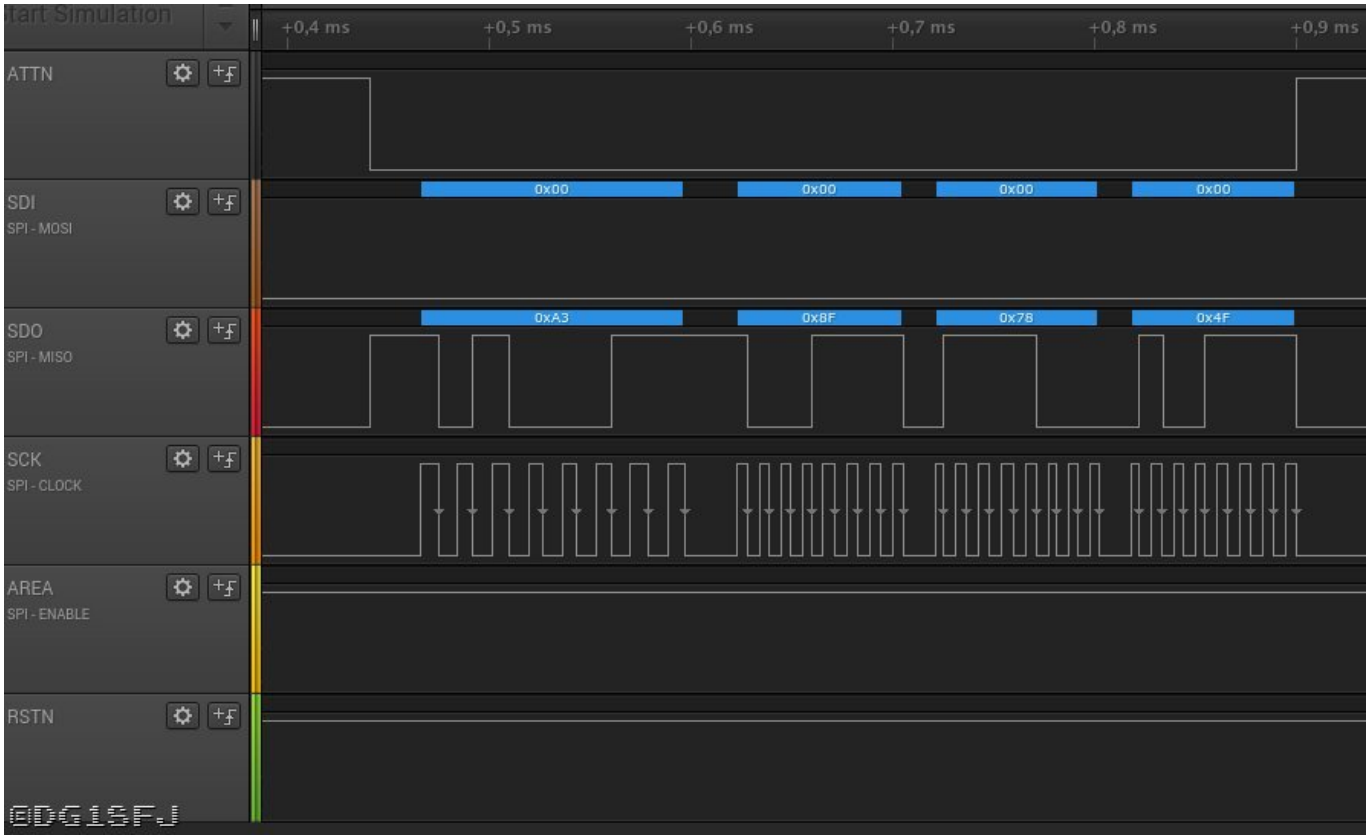
Empfang eines Frames welches zur Adresse passt und zum Alarm führt :

Zuerst geht AREA auf 1 und zeigt an das ein Frame empfangen wird. Nach ca. 200ms beginnt der Auslesevorgang damit das ATTN auf Low geht.



Es werden 5 Blöcke übertragen :

```
0xA3 0x8F 0x78 0x4F (10100011 10001111 01111000 01001111)

0xC0 0x85 0x3E 0x70 (11000000 10000101 00111110 01110000)

0xC0 0x88 0x9A 0x2E (11000000 10001000 10011010 00101110)

0xC0 0x84 0x00 0x00 (11000000 10000100 00000000 00000000)

0xE0 0x80 0x00 0x00 (11100000 10000000 00000000 00000000)
```



Man erkennt das das erste Byte langsamer Übertragen wird (8,7us High, 9,6us Low, Dauer grob 17..18us). Die nächstes 3 Bytes schneller (grob 10..11us, 4.4...6us High).

```
Zuerst kommt die Receive Data – Address Data Format :

0xA3 0x8F 0x78 0x4F (10100011 10001111 01111000 01001111)

101 = Recepetion Data / Adress data format Kennung

000 = Addresse („A..H“) = „A“ = „000“

11 = Function = 11 = „D CALL“

1 = SYN-VAL = High=2 oder weniger Fehler

00 = Error = Low = no errors

0 = Parity error = Low=no parity error

1111 01111000 010011 = Adress 1..18



Beispiel für die Adress-Dekodierung :

dez2027675 = bin11110111100010011011

Die unteren 3 Bits werden abgeschnitten und als Frame-Nummer genommen.
```

Damit : Framenummer 011 und Rest-Adresse ist dann 111101111000010011

Dann folgt die Botschaft :

0xC0 0x85 0x3E 0x70 (11000000 10000101 00111110 01110000)

11000000 = Reception Data / Message Data

1000

1 = SYN-VAL = High=2 oder weniger Fehler

00 = Error = Low = no errors

0 = Parity error = Low=no parity error

0101 00111110 01110000 = Message 1-20

Damit würde insgesamt 396 Bits zur Verfügung stehen „JOCHEN“ = 6mal ASCII:

0xC0 0x85 0x3E 0x70 (11000000 10000101 00111110 01110000)

0xC0 0x88 0x9A 0x2E (11000000 10001000 10011010 00101110)

0xC0 0x84 0x00 0x00 (11000000 10000100 00000000 00000000)

Die Botschaft lautet also (7Bit Blöcke!) :

010100111110011100001000100110100010111001000000000000000000

Da das ASCII LSB zuerst gesendet wird müssen alle gedreht werden :

1001010 = J

1001111 = O

1000011 = C

1001000 = H

1000101 = E

1001110 = N

Der Rest sind 0er und füllen den letzte 20 Bit Block aus.

```
Am Ende wird signalisiert das die Botschaft vorbei ist :

0xE0 0x80 0x00 0x00 (11100000 10000000 00000000 00000000)

11100000 = Reception Data – End of Message

1 = SYN-VAL = High=2 oder weniger Fehler

000 = Unknown

0000 00000000 00000000 = must be zero
```

Microcontroller

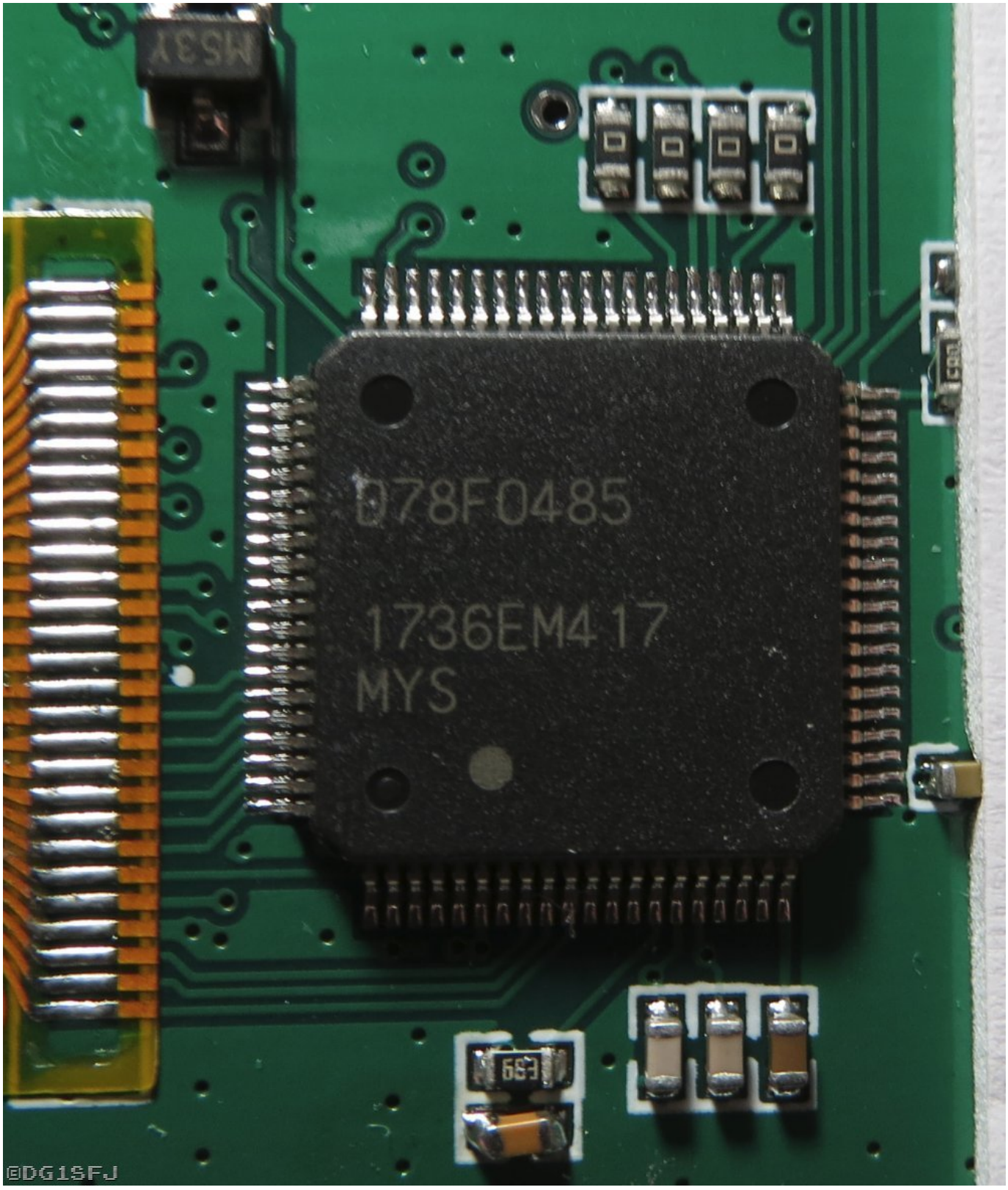
Renesas (ex. NEC) uPD78F0485

80-Pin QFP

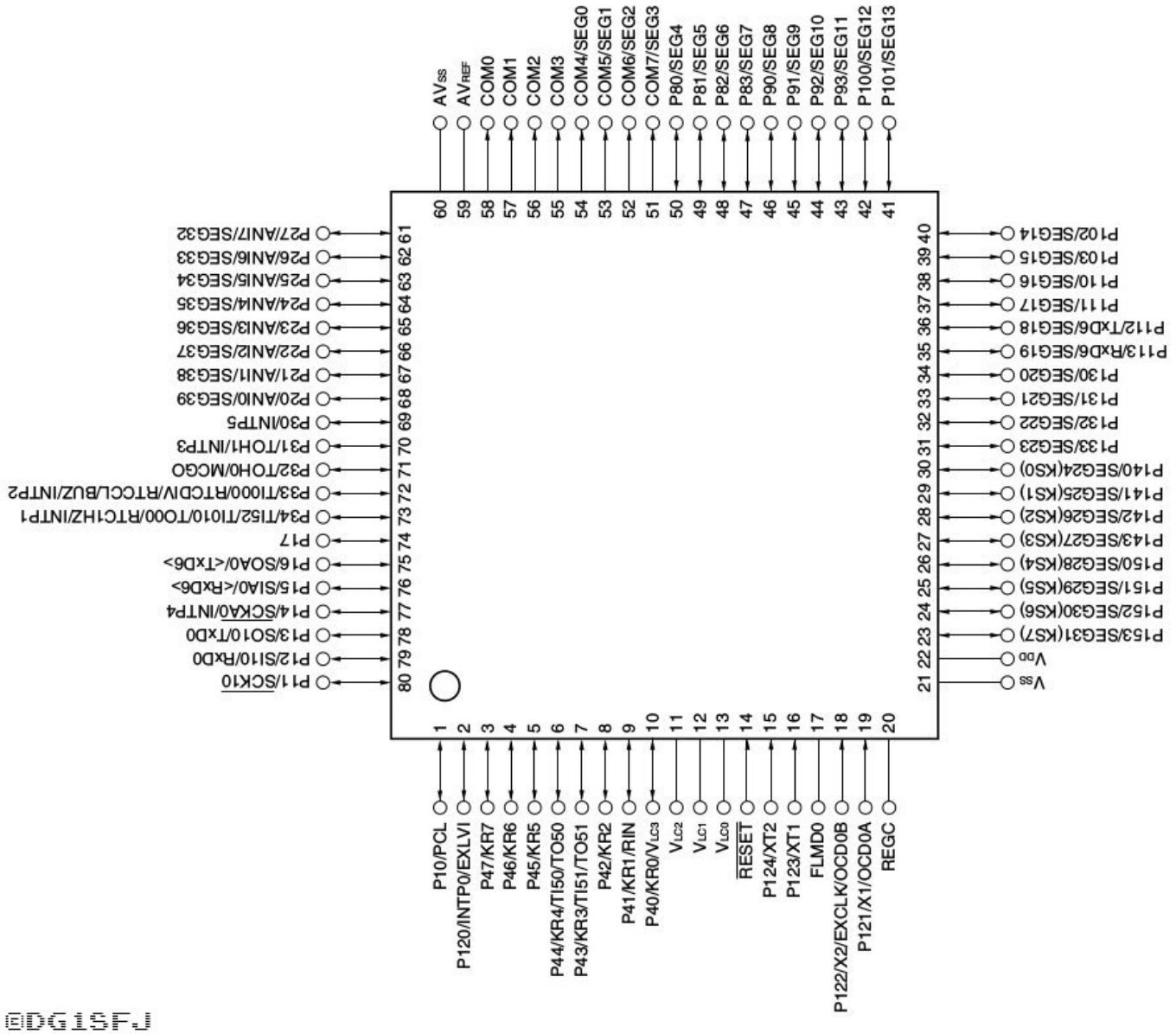
8Bit Controller mit LCD Treiber

78K0/LF3 Serie

60KB ROM, 1KB RAM, 1KB Expansion RAM, Display RAM(40x4bits oder 36x8bits)



Pinout :



©DG1SFJ

```
Belegung der Pins im Alphapoc :



01 - Ausgang zum LCD, D0

02

03

04

05

06 - Ausgang zum aktivieren des Piepsers

07 - Reset Eingang

08 - Ausgang zum LCD, A0
```

```
09 - Ausgang zum LCD, CS

10 - Ausgang zum LCD, WRITE

11

12

13

14

15 - Quarz

16 - Quarz

17 - Flashmode

18

19

20 - REGC - Versorgungsspannung Plus

21 - VSS - GND

22 - VDD - Versorgungsspannung Plus

23

24

25

26

27

28

29

30

31

32

33

34
```

```
35 - RXD Flash Reprgo

36 - TXD Flash Reprog

37

38

39

40 - ADC Eingang ? 68k zu 100k Teiler

41 - Ausgang zum aktivieren der LED Hintergrundbeleuchtung

42 - Ausgang zum aktivieren des Vibrationsmotors

43 - Flash - Chip Select

44 - Flash - Serial Clock

45 - Flash - Serial Data In

46 - Flash - Serial Data Out

47 - Eingang von Taste "UP"

48 - Eingang von Taste "DOWN"

49 - Eingang von Taste "Function/Select"

50 - Eingang von Taste "Read/Escape"

51

52

53

54

55

56

57

58

59

60 - VSS - GND
```

```
61

62 - CLK vom PLL IC

63 - Data vom PLL IC

64 - LE vom PLL IC

65

66

67

68

69

70 - Interrupt-Eingang erzeugt vom Schaltregler

71

72

73 - Sammelleitung für die 4 Tasten

74 - Ausgang zum LCD, D7

75 - Ausgang zum LCD, D6

76 - Ausgang zum LCD, D5

77 - Ausgang zum LCD, D4

78 - Ausgang zum LCD, D3

79 - Ausgang zum LCD, D2

80 - Ausgang zum LCD, D1
```

Empfängerplatine N_RF400MP_V4

Die Empfängerplatine ist mir 2 Stifleisten auf der Hauptplatine aufgesteckt. Je nach Frequenzbereich können hier im Pager also unterschiedliche Empängerplatinen nachgerüstet werden. Auf der Platine ist auch die Antenne befestigt. Auf den 2 Stifleisten liegen die Signale für/vom FM IC sowie die Steuerinformationen für das PLL IC.

Vorder/Rückseite

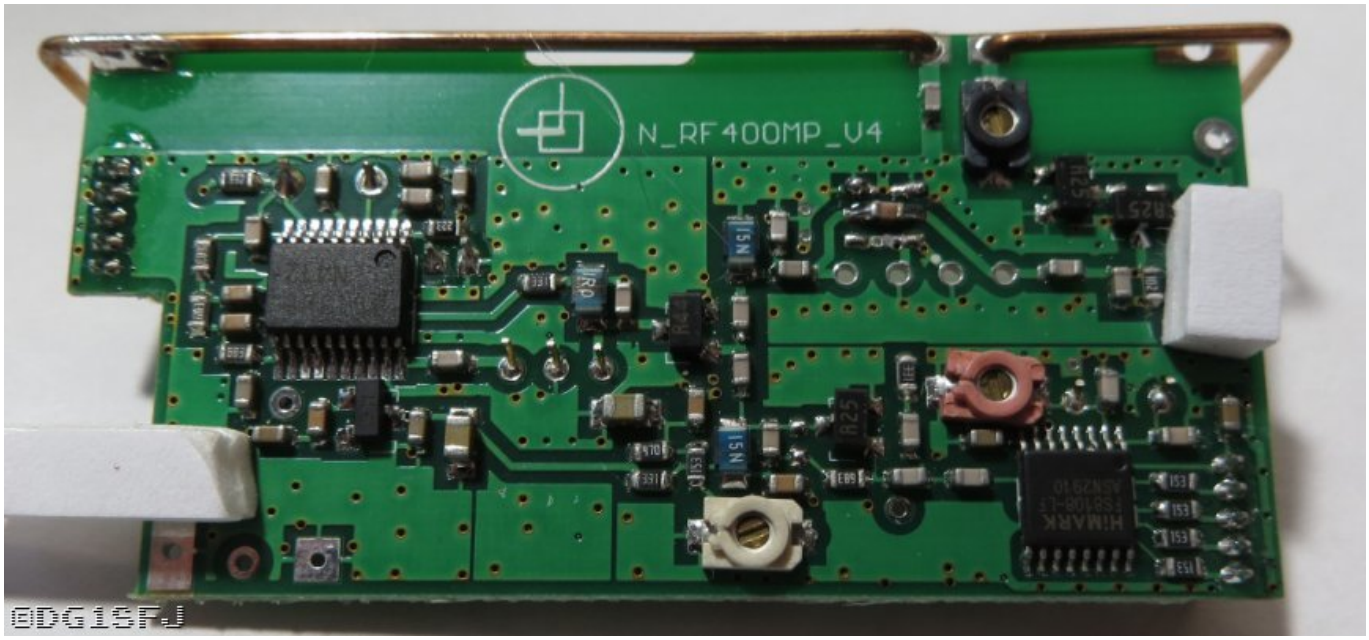
Vorderseite :

„R25“ = NEC/Renesas 2x2SC3356 NPN Transistor

„R44“ = NEC/Renesas 2SC4228 oder 2SC3585 NPN Transistor

„Tf“ = Varicap Diode Toshiba 1SV270

„M6“ = PNP Transistor (Steuerung um REGO Ausgang für VCC am BSC durchzuschalten. Siehe Application Note des ICs)



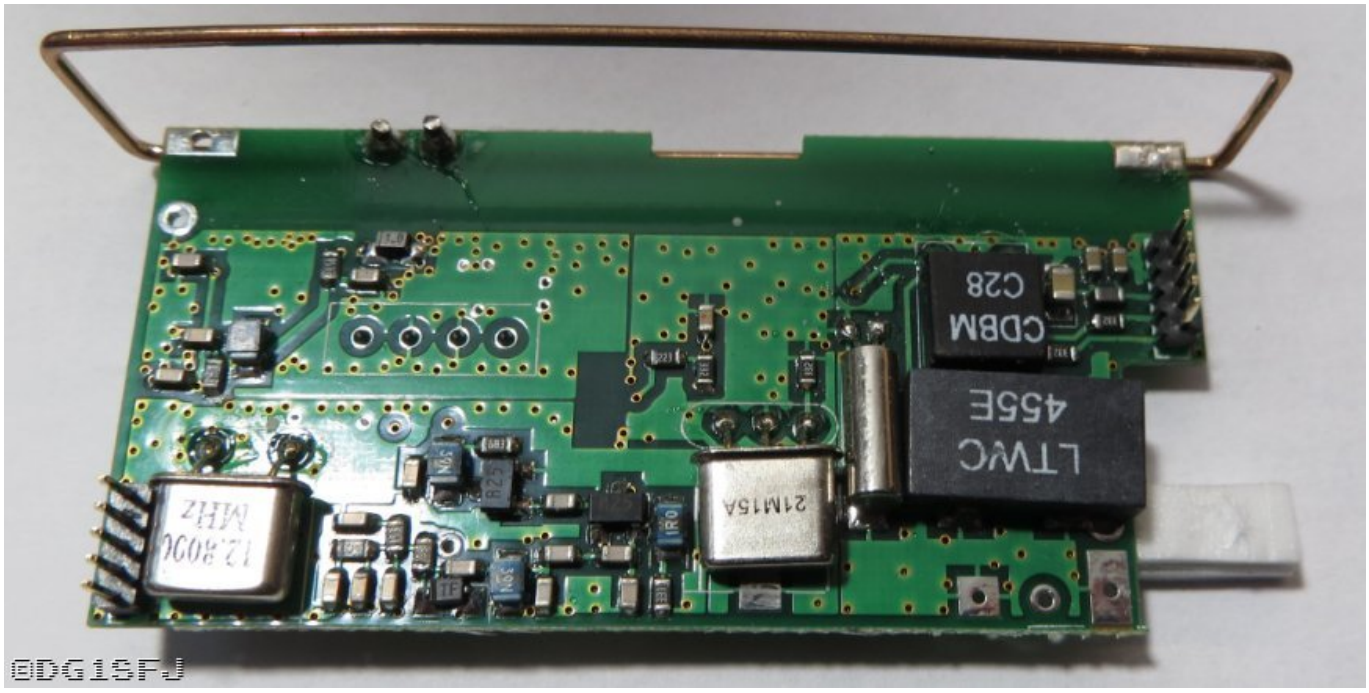
Rückseite :

„1_B“ = ESD Schutz Diode

„CDBM C28“ = Murata Keramik Diskiminator für 455kHz +-4kHz Bandbreite (3dB)

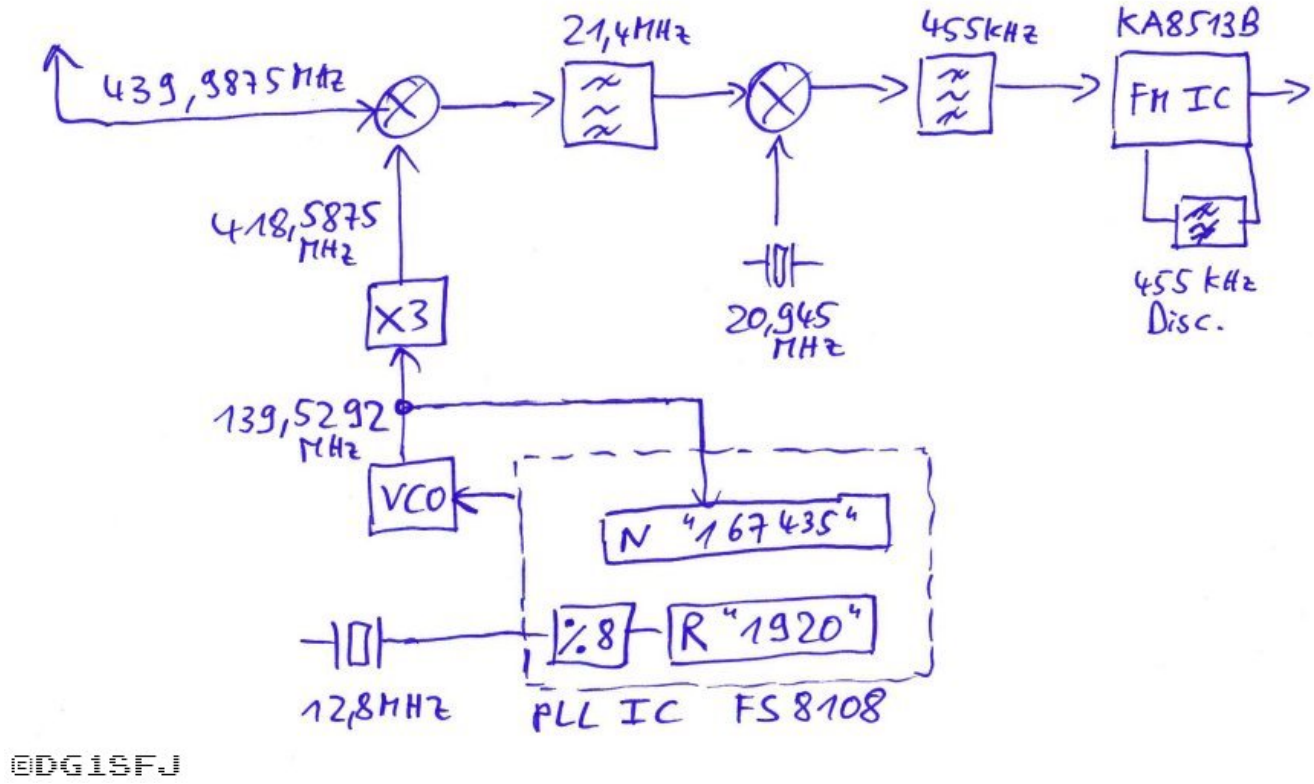
„LTWC 455E“ = Shoulder Keramik Filter 455kHz +-7,5kHz Bandbreite (6dB)

„21M15A“ = Quartz-COM Quarz-Filter 21,4MHz +-7,5kHz Bandbreite (3dB)



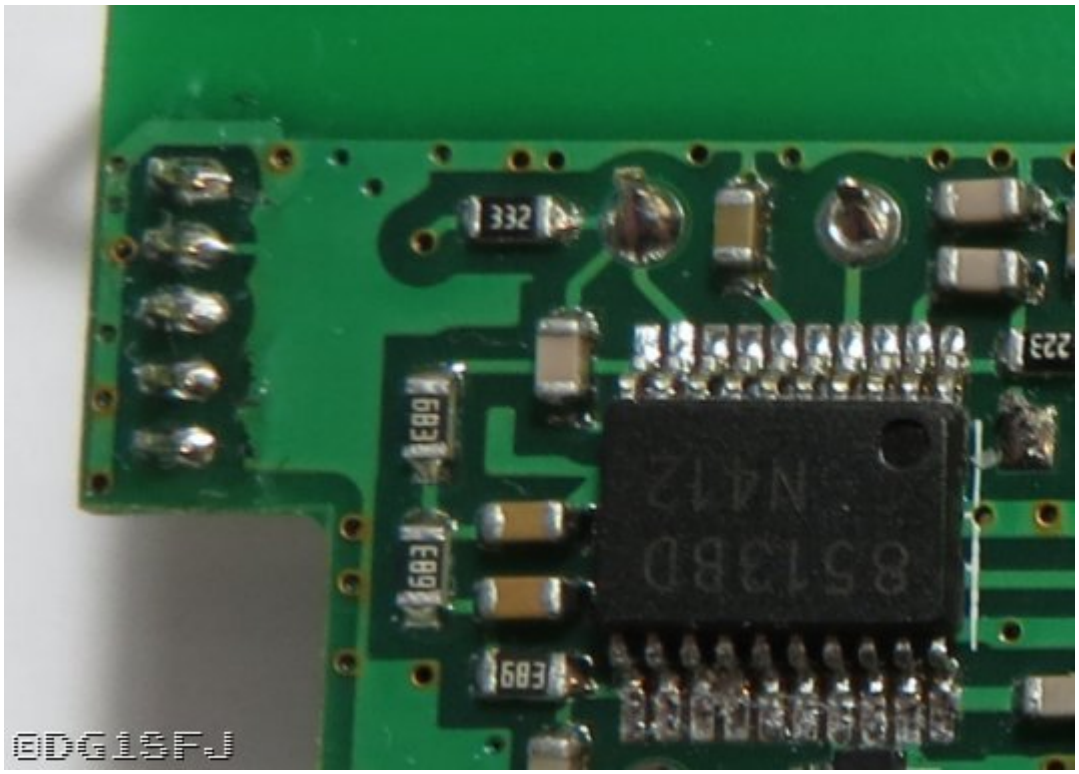
Blockschaltbild

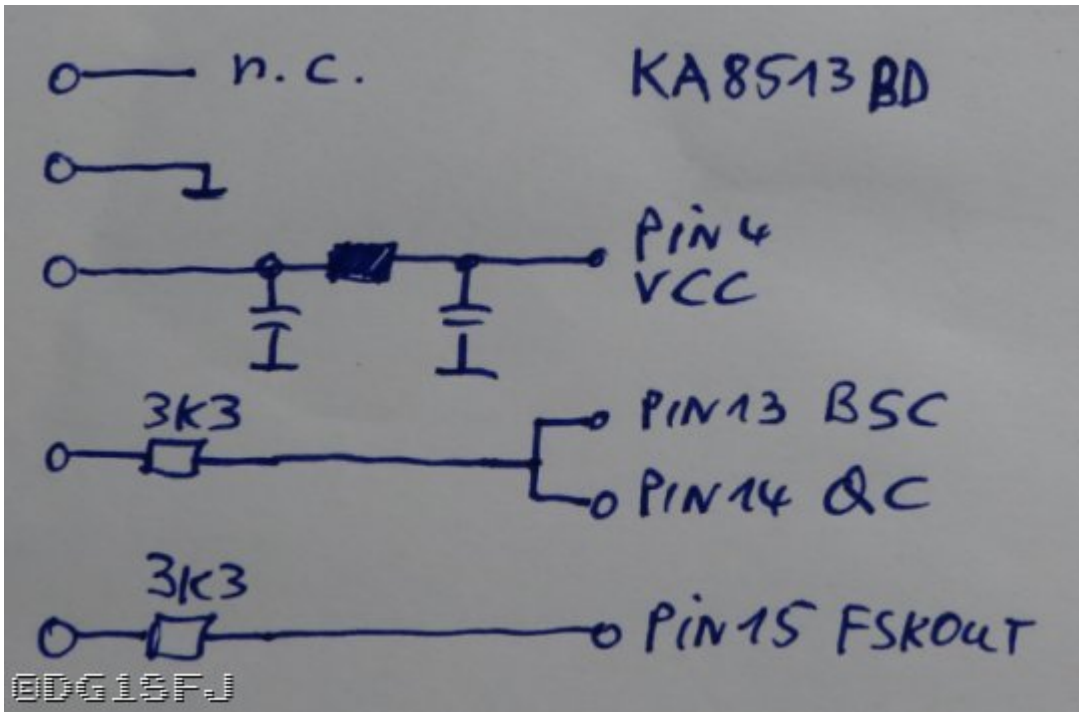
Blockschaltbild des Empfängers mit PLL und zwei Mischern. Vorverstärker nach der Antenne nicht gezeichnet.



Steckerbelegung

Oben links befindet sich die Anschlussleiste, an welcher die Signale des FM ICs anliegen.



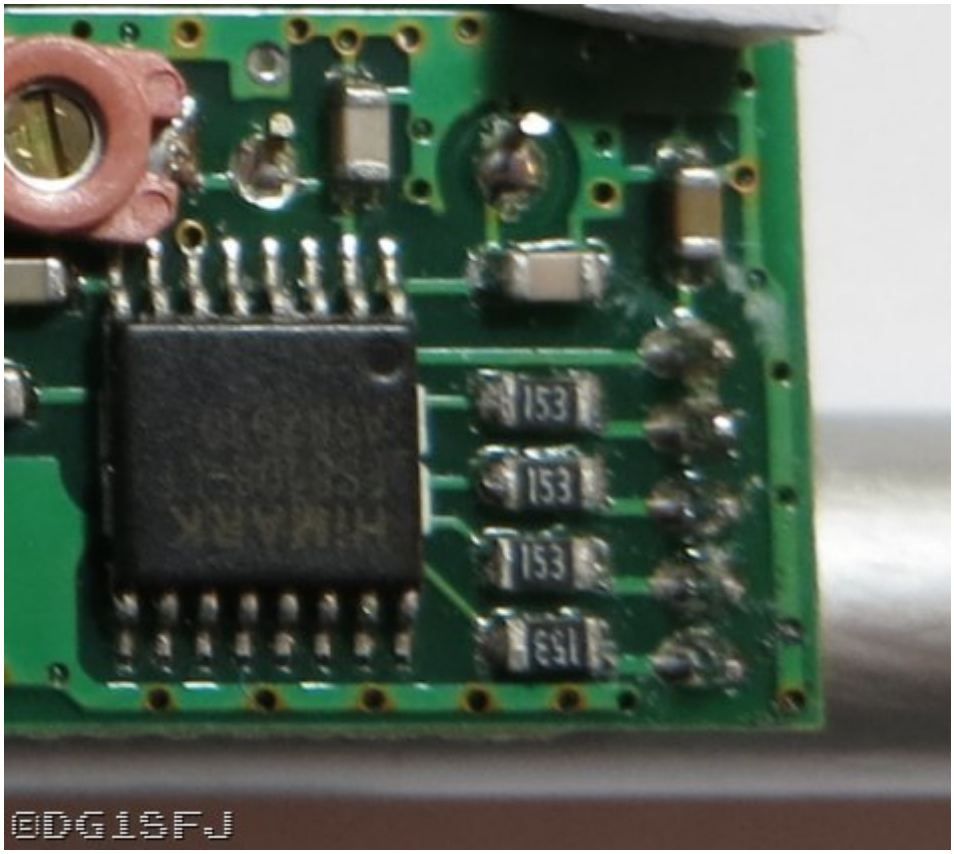


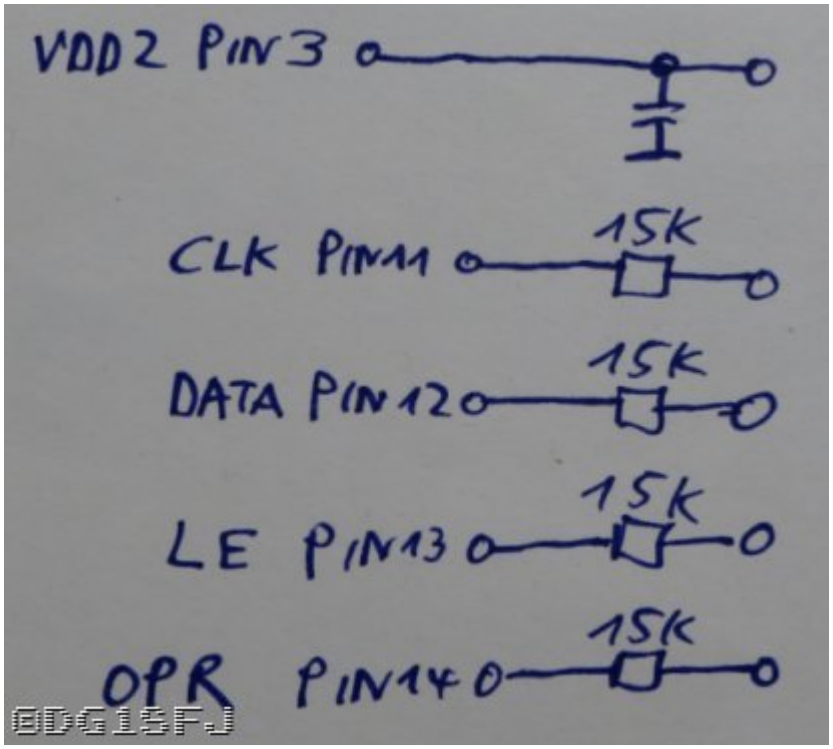
BSC - Pin 13 : Battery Saving Control Pin

QC - Pin 14 : Quickcharge Control Pin

FSKOUT - Pin 15 : Rückgewonnene FSK Information

Unten rechts befindet sich die Anschlussleiste wo die Signale für die PLL anliegen :





CLK - Pin 11 : Shift Register clock input

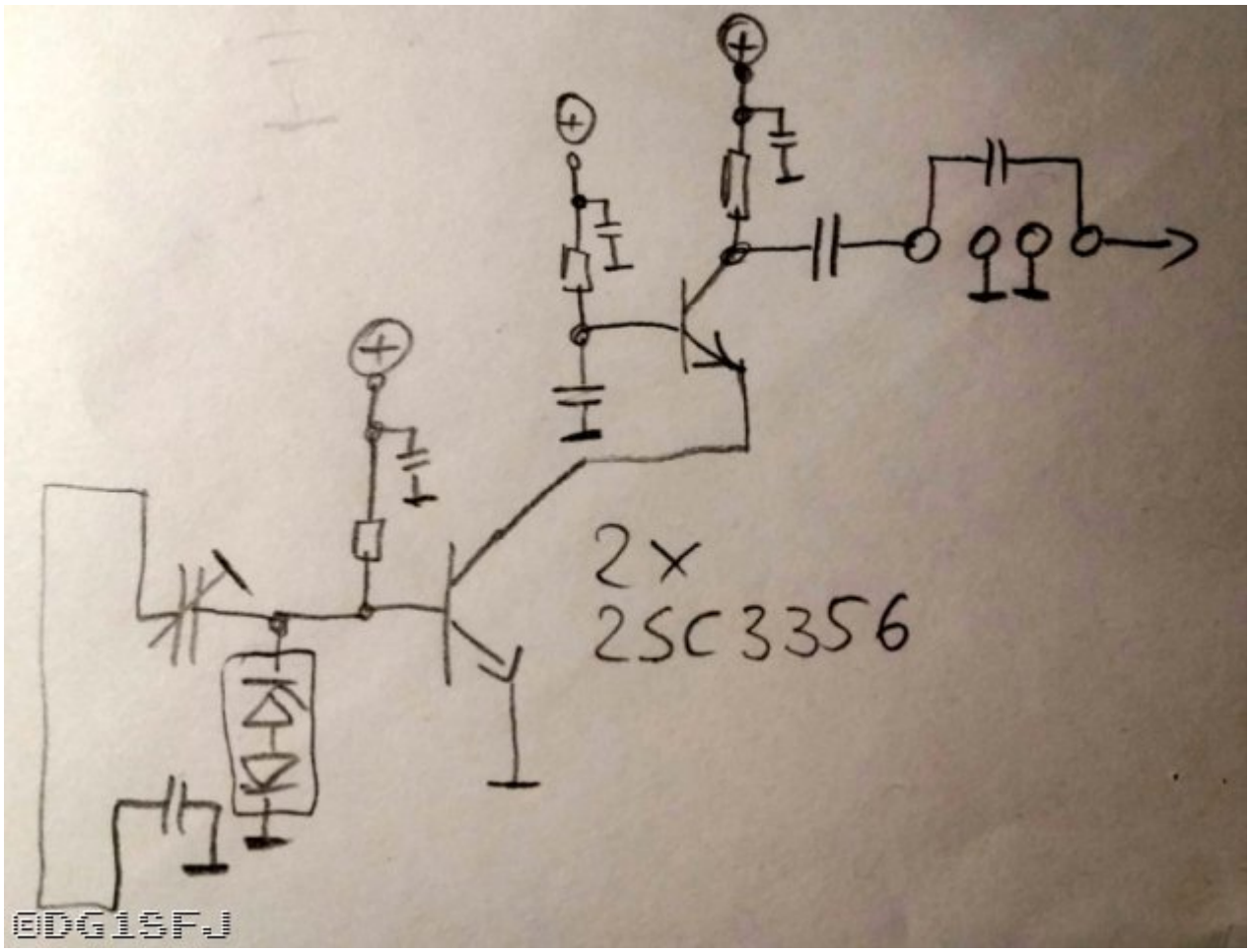
DATA - Pin 12 : Serial Data Input

LE - Pin 13 : Latch enable Input

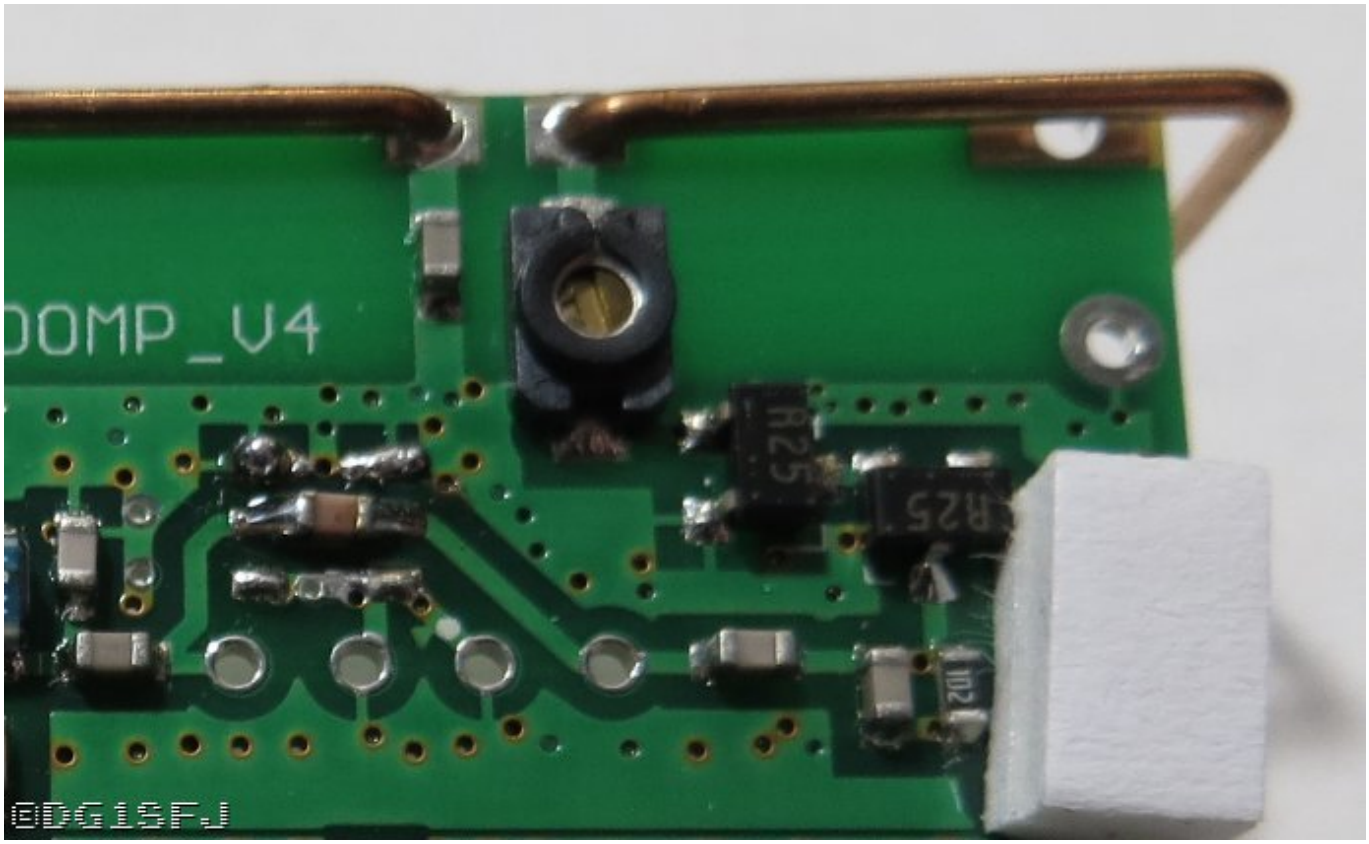
OPR - Pin14 : Battery Save Control Input

Antenne

Kaskode als Antennenverstärker mit 2x2SC3356 „low noise npn“, Package Marking „R25“



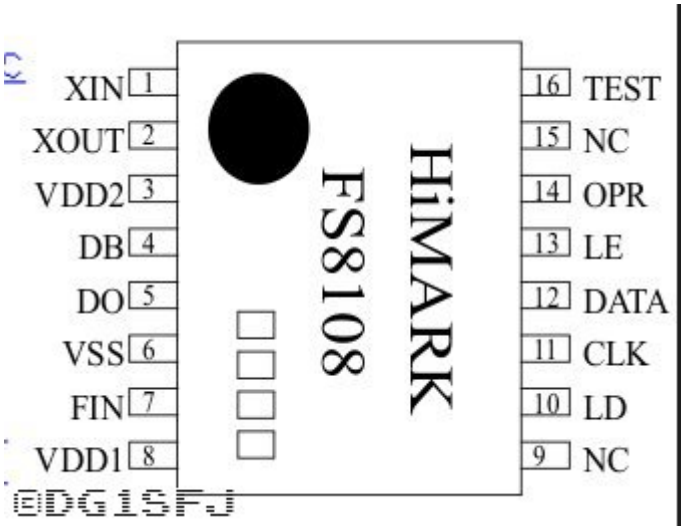
Danach schließt sich eine Einbaumöglichkeit (4 Vias) für einen Filter an, welcher aber mit einem Kondensator überbrückt ist :



PLL-IC

HiMARK FS8108, 16Pin TSSOP, Low Power Phase-Locked Loop IC

PTC Princeton Technology Corp.



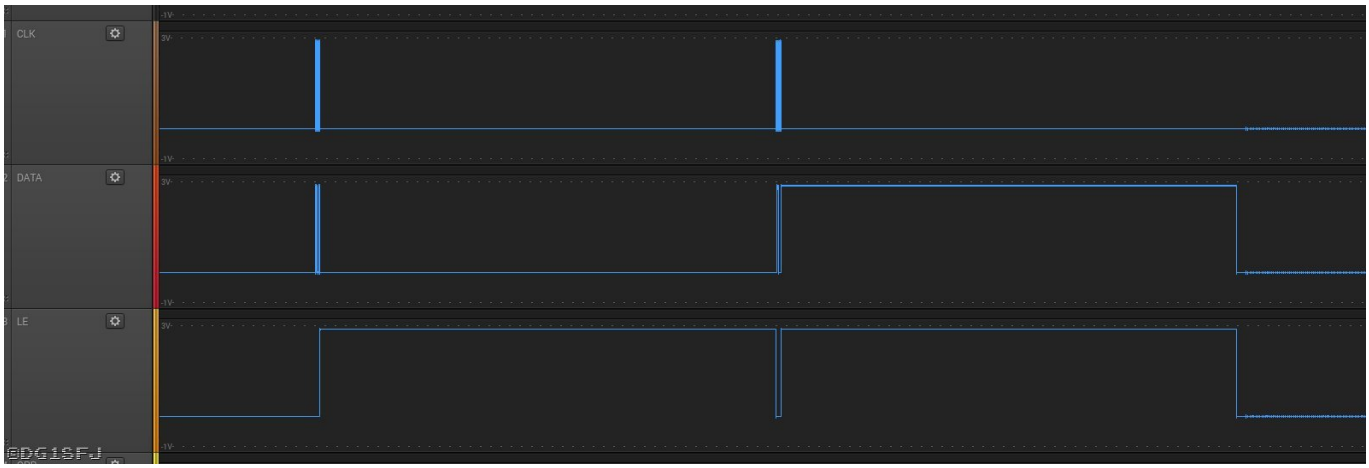
Number	Name	I/O	Description
1	XIN	I	Reference crystal oscillator or external clock input with internally biased amplifier (any external input to XIN must be ac-coupled)
2	XOUT	O	Reference crystal oscillator or external clock output
3	VDD2	POWER	Nominal 3.0 V supply voltage
4	DB	O	Single-ended quick-lock output for faster locking
5	DO	O	Single-ended charge pump output for passive low pass filter
6	VSS	GND	Ground
7	FIN	I	VCO frequency input with internally biased input amplifier (any external input to FIN must be ac-coupled)
8	VDD1	POWER	Nominal 1.0 V supply voltage
9	NC	NC	No connection
10	LD	O	Lock detector output (high when PLL is locked)
11	CLK	I	Shift register clock input
12	DATA	I	Serial data input
13	LE	I	Latch enable input
14	OPR	I	Battery-save control input; normal operation when high, stand-by mode when low
15	NC	NC	No connection
16	TEST	I	Test mode control input with internal pull-down resistor

VDD2 :

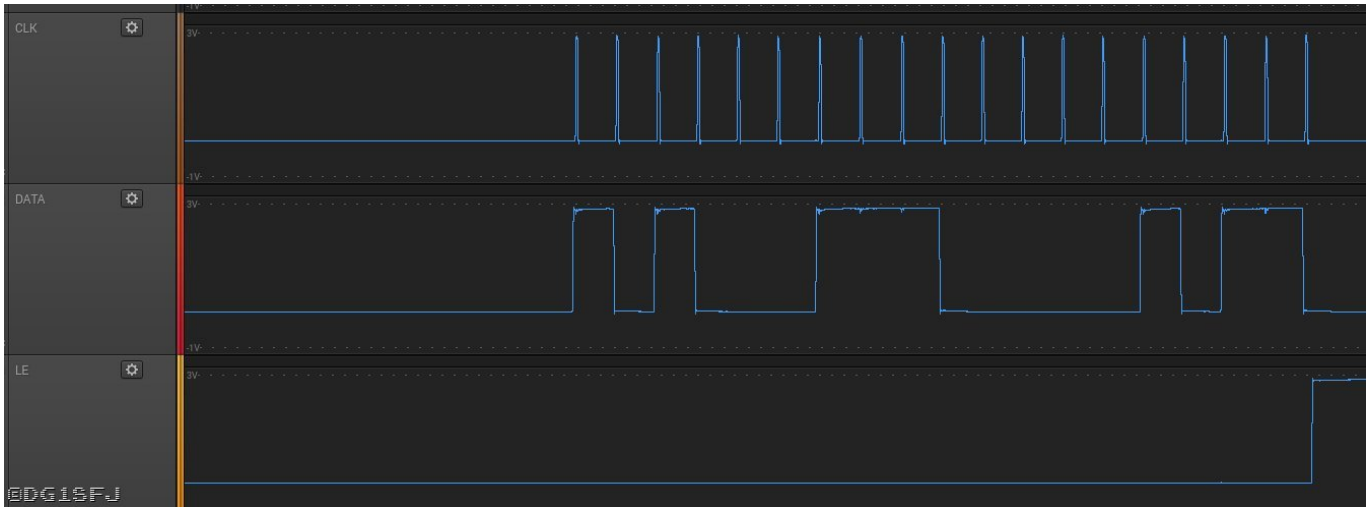
Die VDD2 mit 2,9V liegt dauerhaft an und wird nicht abgeschaltet.

LE/DATA/CLK :

Das IC wird initial einmal programmiert (6,6 Sekunden nach Batterie einlegen) und danach nur noch über OPT aktiviert und deaktiviert. Es werden dazu 2 Sequenzen in einem Abstand von 85ms losgeschickt.



Sequenz 1 :



Bitdauer 50us.

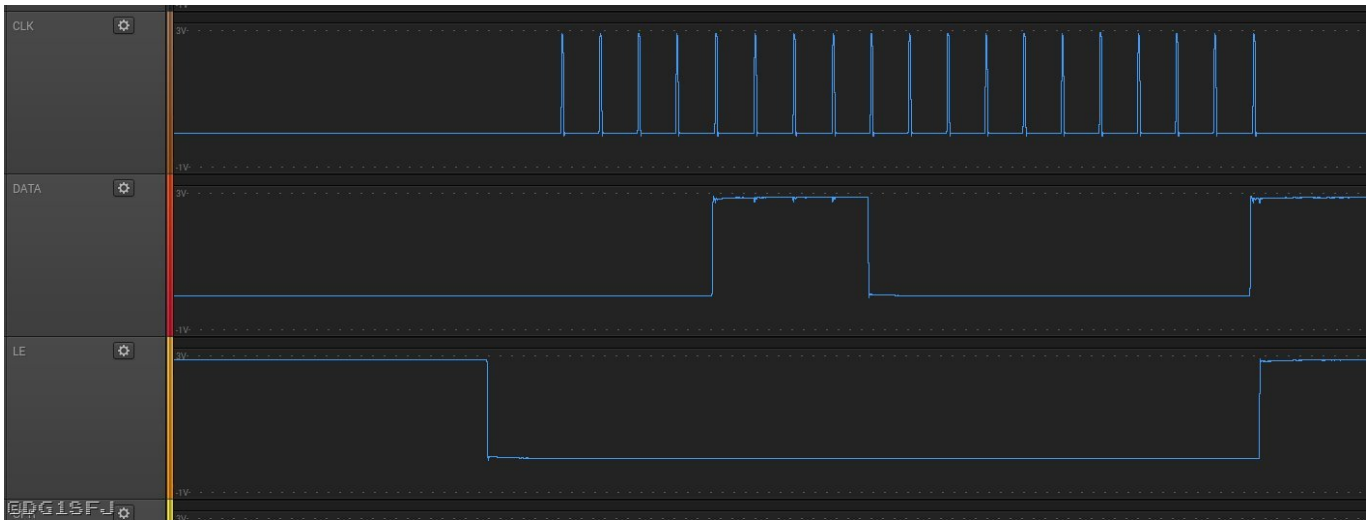
Botschaft „10100011100000010110“ (MSB first, ControlBit19=0 : 18Bit N-counter)

Ergibt die 18Bit Zahl „10100011100000011011“ = 167435

Übernahme bei steigender Flanke

Nach 85ms kommt dann Sequenz 2 :

Sequenz 2 :



Botschaft „00001111000000000001“ (MSB first, ControlBit19=1 : 13Bit R-Counter)

2 MSBs werden ignoriert

13 Bit „0011110000000“ = 1920dez

000 = R14=0 LD Output Disable sowie R15 R16 = 0

1 Control Bit

Funktionsweise :

12,8MHz Quarz wird intern im IC durch 8 geteilt (fix) ergibt 1,6MHz, geteilt durch R counter 1920

ergibt 8,333kHz, also das Kanalraster mit welchen die PLL arbeitet.

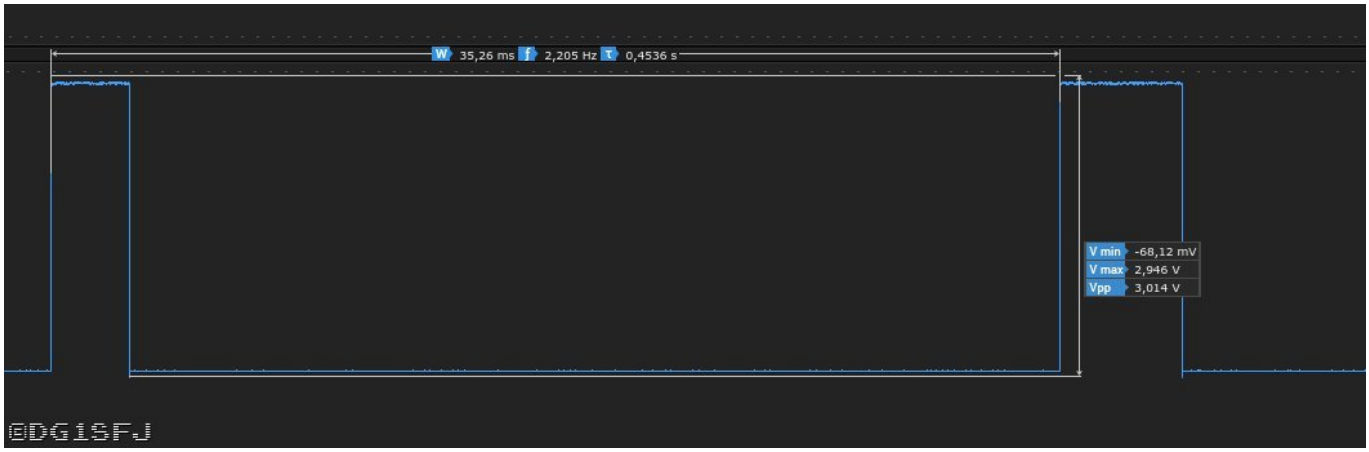
Nimmt man nur die 8,3333kHz mal der 167435 (siehe oben) ergibt sich 139,5292MHz. Auf dieser Frequenz läuft ein Oszillator den die PLL steuert. Diese Frequenz wird verdreifacht auf 418,5875MHz welche im Mischer aus der 439,9875MHz dann die erste ZF auf 21,4MHz erzeugt.

Dann folgen ZF-Filter auf 21,4MHz, Mischung mit 20,945 MHz auf die zweite ZF von 455kHz welche dann in den FM-IC gehen.

OPR :

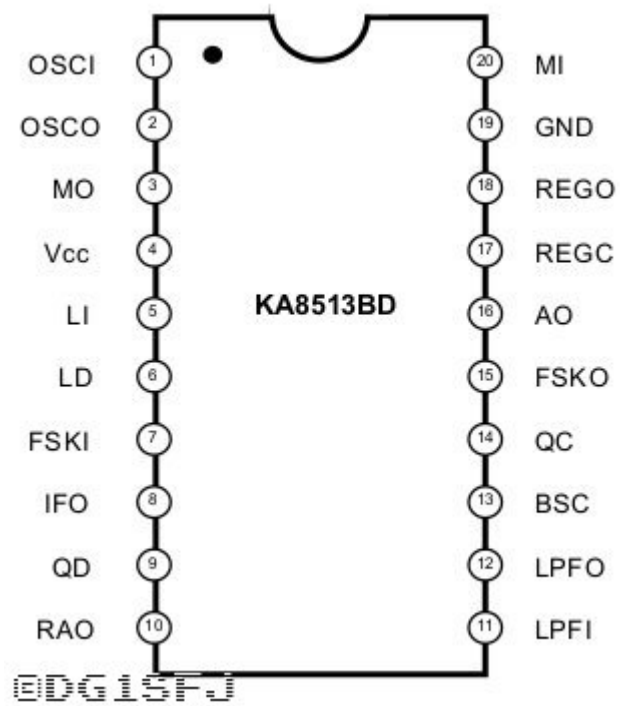
Der Zyklus des OPR ist ca. 450ms. Davon 50..100ms „High“, der Rest auf „Low“.

Bei High ist die PLL aktiv, beim Low im Standby-Mode.



FM-IF IC

KA8513BD, SSOP20, Samsung, FM IF Receiver, Supply 1..4V, Vreg 1.0V



PIN DESCRIPTION

Pin No	Symbol	Description
1	OSCI	Oscillator input. The oscillator is an internally-biased colpitts type.
2	OSCO	Oscillator output.
3	MI	Mixer output pin Output impedance ≒ 2KΩ Connect a 455KHz filter between this pin and the LI.
4	V_{CC}	V_{CC} pin.
5	LI	IF limiter amplifier input. Input impedance ≒ 2KΩ
6	LD	Bypass capacitor connect pin for the IF limiter amplifier.

DG1SFJ

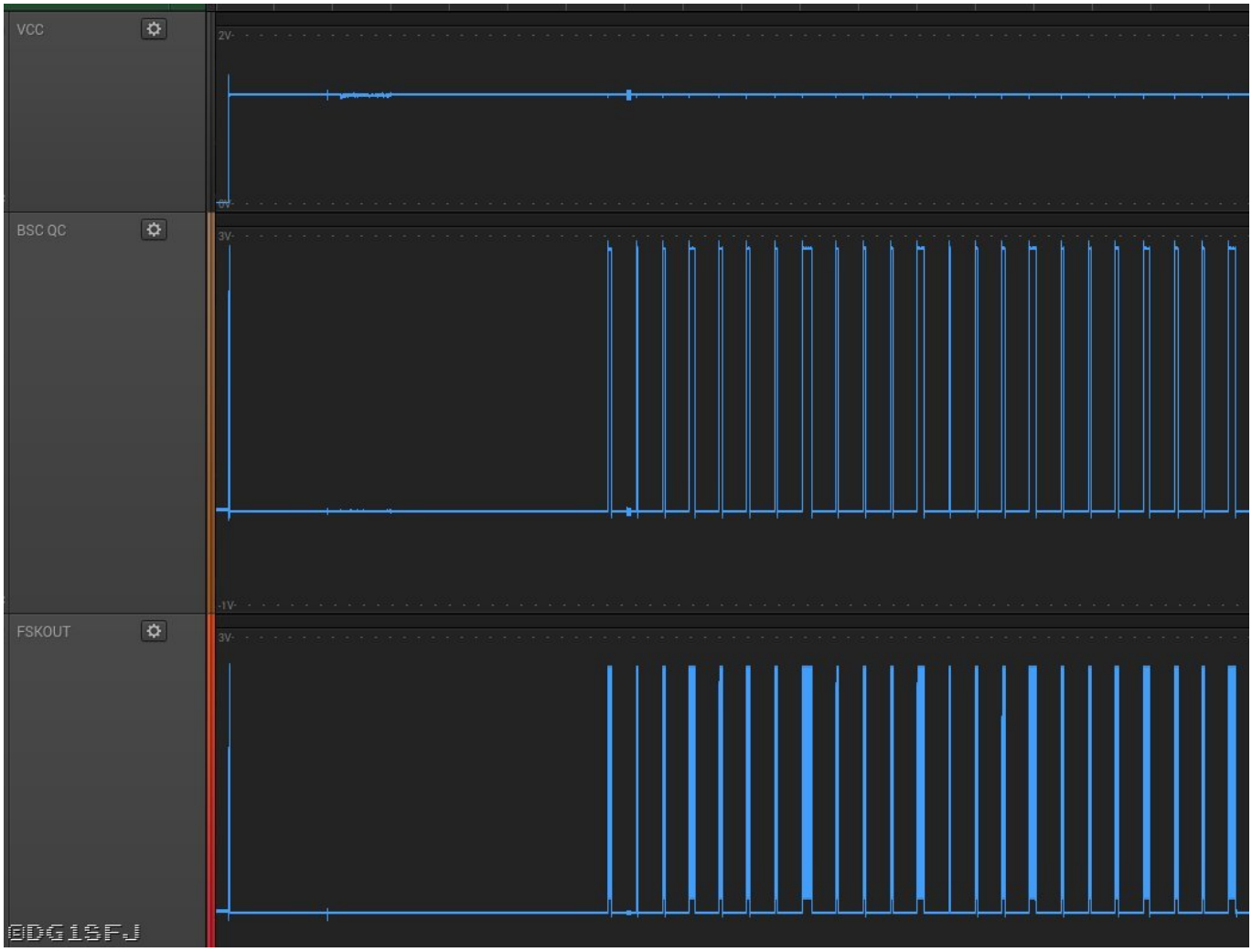
PIN DESCRIPTION (Continued)

Pin No	Symbol	Description
7	FSKI	Differential Amp reference input on the FSK comparator.
8	IFO	IF Amp output.
9	QD	Quadrature detection phase shifter pin.
10	RAO	Recovered audio signal output.
11	LPFI	Low pass filter amplifier input Bias is supplied from pin 10.
12	LPFO	Low pass filter amplifier output.
13	BSC	Battery saving control pin. High : Battery saving off. Low : Battery saving on.
14	QC	Quick charge control pin. High : Quick charge-discharge on. Low : Quick charge-discharge off.
15	FSKO	FSK signal output pin.
16	AO	Alarm output. This pin becomes High when V_{CC} drops below 1.05V
17	REGC	Internal Transistor control pin. Normal mode : open
18	REGO	Regulated voltage output.
19	GND	Ground.
20	MI	Mixer input impedance ≒ 5KΩ.

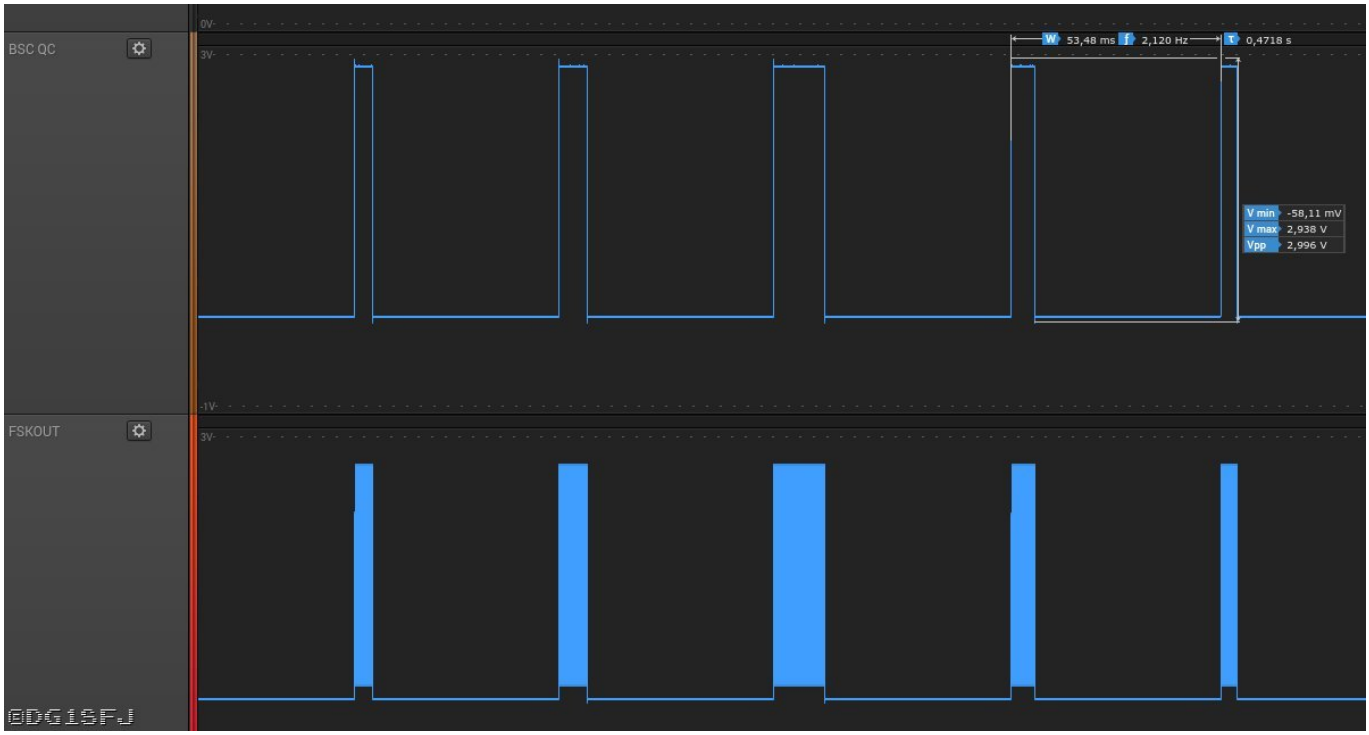
DG1SFJ

VCC ist 1,3V dauerhaft nach dem einlegen der Batterie.

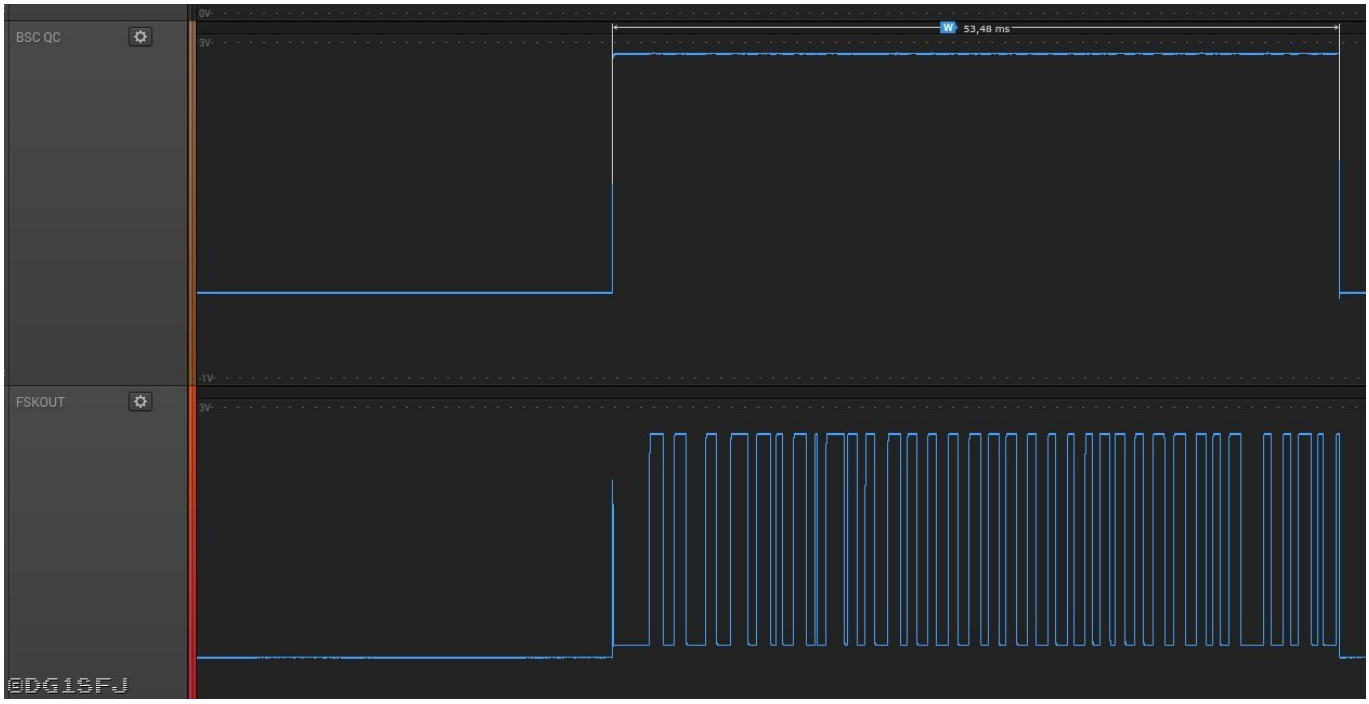
BSC/QC sind getaktet und aktivieren das IC. Im Schnitt 0,5s Zyklus, 40ms an (High) und der Rest der Zeit Low / aus. Ab und an gibt es längere An-Phasen. Das Takten beginnt ca. 6,5Sekunden nach Batterie einlegen.



Taktzyklus der BSC/QC Leitung bei grob 54ms zu 470ms. Also ca. 1/10 der Zeit ist der Empfänger an.



Beispiel eines Empfangszyklus - während des Empfangs geht BSC/QC auf High, die Daten kommen auf der FSKOUT Leitung heraus. Der Empfänger ist aktiv für ca. 54ms :



Diese Daten gehen dann in den POCSAG Decoder IC zur weiteren Verarbeitung.

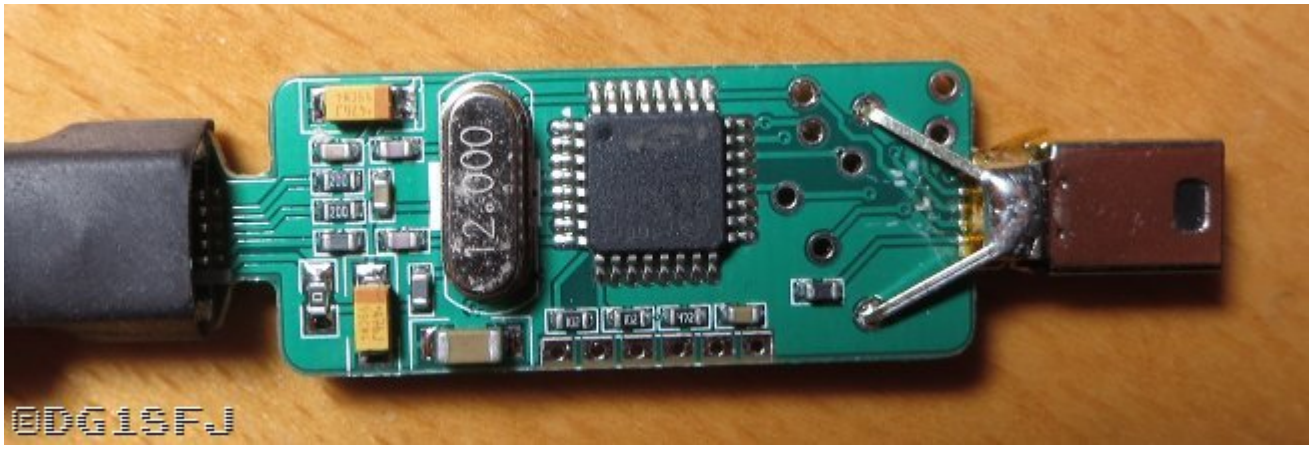
USB Programmierkabel

Für den Alphapoc 602R wird ein spezielles Programmierkabel verkauft. Der Pager hat zwar optisch eine 10pol Mini-USB Schnittstelle es liegt aber kein USB Controller innen vor. An den Pins ist lediglich das interne Flash herausgeführt und kann von dem Programmierkabel mit Software ausgelesen und neu beschrieben werden.

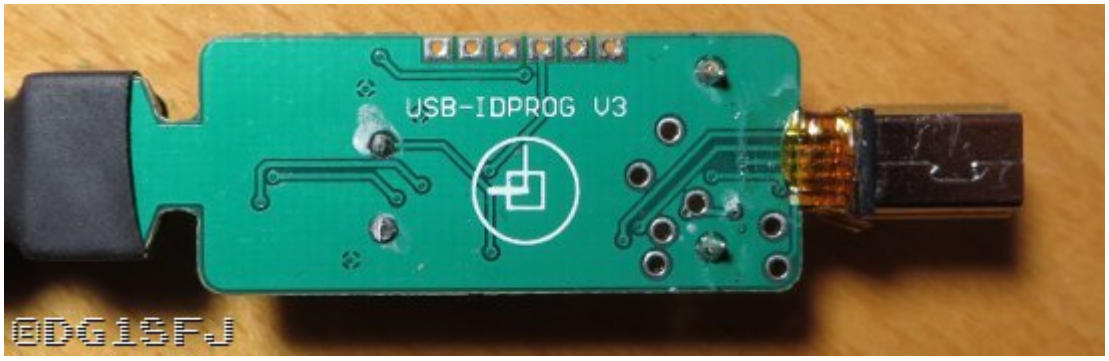
Pager und Kabel zusammengesteckt sieht das dann so aus :

Aussen und Innenansichten

Oberseite der Leiterplatte :

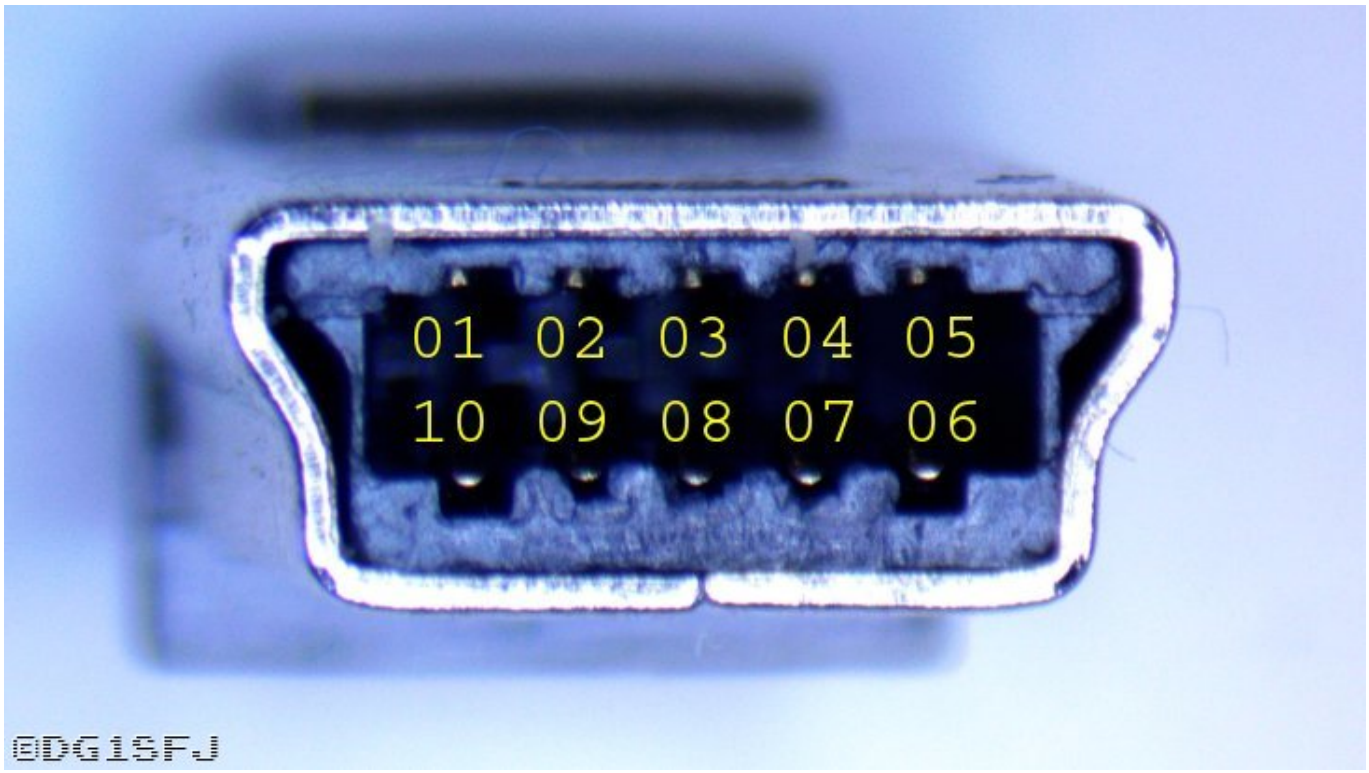


Unterseite der Leiterplatte :



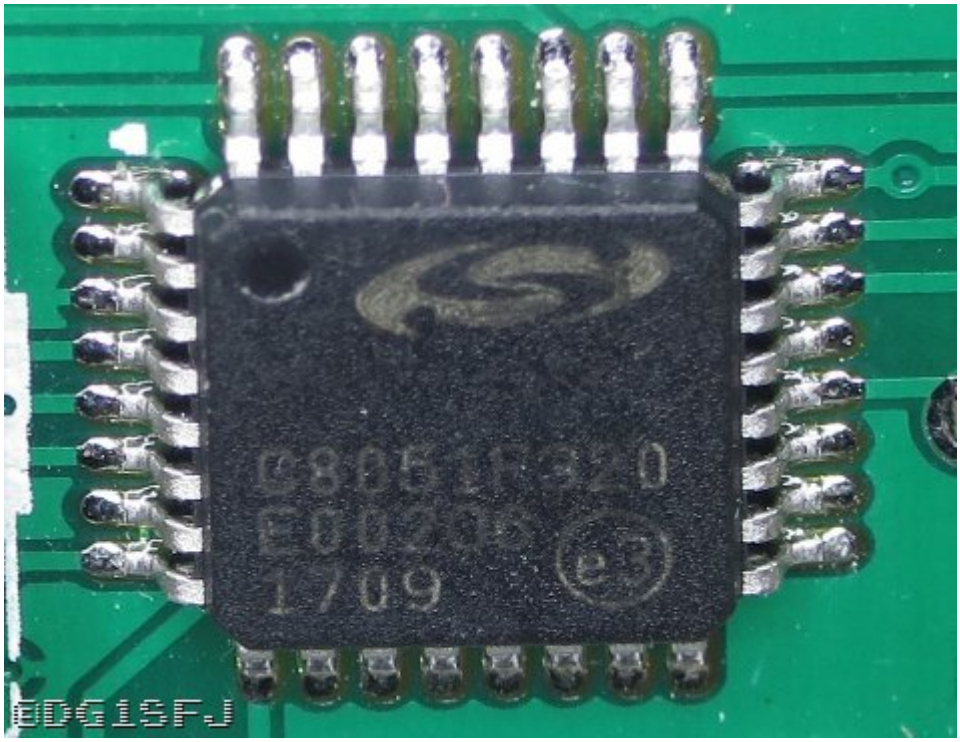
Pinbelegung

Der 10Pin Mini-USB ist wie folgt belegt :




```
01 – via Ferrit auf Masse, also Ground
02 – uC Pin 25 [P1.1] – zum SPI-Flash Pin 2 – Serial Data In
03 – uC Pin 22 [P1.4] - zum SPI-Flash Pin 50 – Serial Data Out
04 – n.c. *
05 – mit Pin 7 verbunden – uC Pin 20 [P1.6]
06 – n.c. *
07 – mit Pin 5 verbunden
08 – n.c. *
09 – uC Pin 23 [P1.3] - zum SPI-Flash Pin 6 – Serial CLK
10 – uC Pin 24 [P1.2] - zum SPI-Flash Pin 1 – /ChipSelect

* Es scheint noch eine Variante zu geben in der die unbenutzten Pins zum
Reprogrammieren des Renesas Controllers im Pager verwendet werden :

04 – TXD-Pin des Renesas – Pin 36 : P112/TXD/SEG18
06 – Flashmode-Pin des Renesas – Pin 17
08 – RXD-Pin des Renesas – Pin 35 : P113/RXD/SEG19
```

Microcontroller

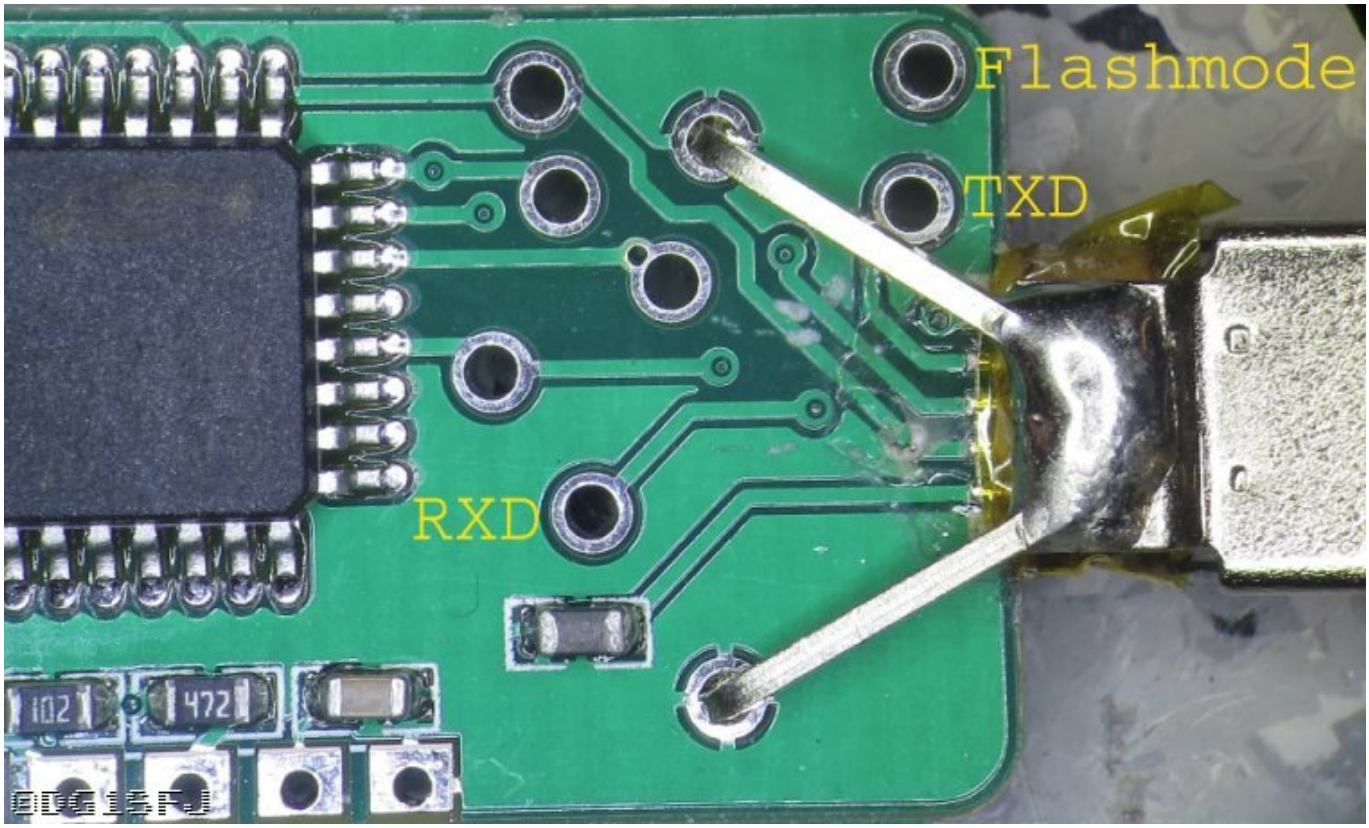


Dies ist ein C8051F320 Controller von Silicon Labs. Sozusagen ein moderner 8051 Clone mit eingebautem USB - das ganze im 24 Pin Gehäuse. Daran angeschlossen wie üblich ein 12 MHz Quarz.

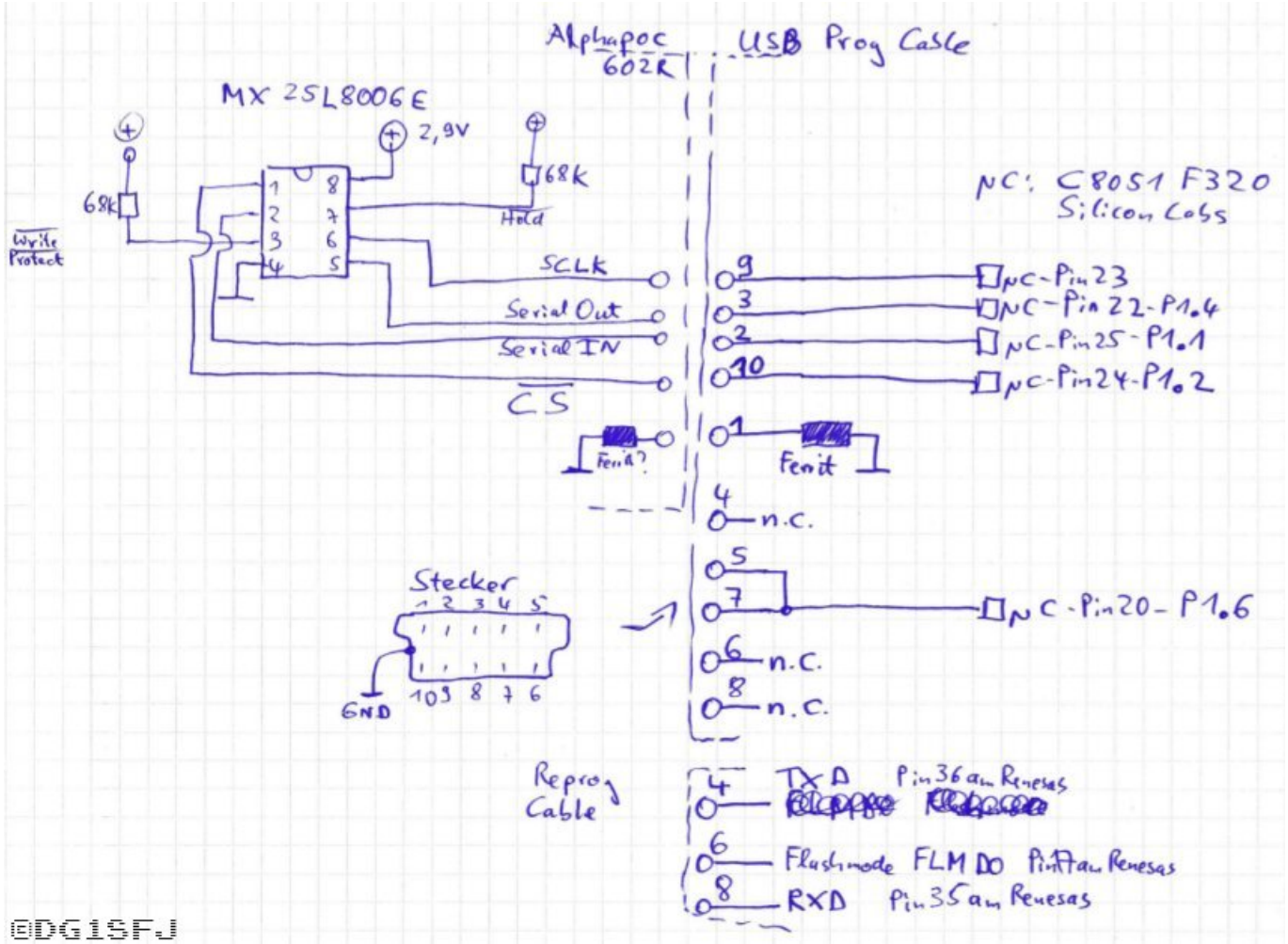
Die ungenutzten Pins auf der unteren Seite sind vermutlich die Reprogrammierpins für den Silicon Labs Controller. Habe ich mir auch nicht näher angeschaut.

Die ungenutzten Vias auf der rechten Seite sind die Reprogrammierpins für den Renesas im Pager :

```
04 – TXD-Pin des Renesas – Pin 36 : P112/TXD/SEG18

06 – Flashmode-Pin des Renesas – Pin 17

08 – RXD-Pin des Renesas – Pin 35 : P113/RXD/SEG19
```



Damit ergibt sich bzgl. der SPI zum Flash folgende Gesamtschaltung :



GP2012 Software

Die GP2012-Software kommuniziert mit dem Controller über USB auf dem Programmierkabel.Die Daten auf dem USB habe ich mir nicht näher angeschaut, nur die Kommunikation zwischen den Controller und dem Flash.

Das „READ“ :

Dieser liest das Flash auf dem Pager via SPI aus per FAST READ (0x0B 0x08 0x80 0x00) :

```
0x080000, ab hier 256 Bytes mit den Konfig-Daten wie oben beschrieben

Danach werden die Speicher mit den empfangenen Botschaften ausgelesen :

0x081000, ab hier 256 Bytes
0x081100, ab hier 256 Bytes
0x081200, ab hier 256 Bytes
…..
0x081F00, ab hier 256 Bytes
dann
0x088000, ab hier 256 Bytes
0x088100, ab hier 256 Bytes
0x088200, ab hier 256 Bytes
dann
0x089000, ab hier 256 Bytes
0x089100, ab hier 256 Bytes
0x089200, ab hier 256 Bytes
dann
0x08A000, ab hier 256 Bytes
0x08A100, ab hier 256 Bytes
0x08A200, ab hier 256 Bytes
dann
0x08B000, ab hier 256 Bytes
0x08B100, ab hier 256 Bytes
0x08B200, ab hier 256 Bytes
dann
0x08C000, ab hier 256 Bytes
0x08C100, ab hier 256 Bytes
0x08C200, ab hier 256 Bytes
dann
0x08D000, ab hier 256 Bytes
0x08D100, ab hier 256 Bytes
0x08D200, ab hier 256 Bytes
dann
0x08E000, ab hier 256 Bytes
0x08E100, ab hier 256 Bytes
0x08E200, ab hier 256 Bytes
dann
0x08F000, ab hier 256 Bytes
0x08F100, ab hier 256 Bytes
0x08F200, ab hier 256 Bytes
dann
```

```
0x090000, ab hier 256 Bytes
0x090100, ab hier 256 Bytes
0x090200, ab hier 256 Bytes
…
0x0A0000,ab hier 256 Bytes
0x0A0100,ab hier 256 Bytes
0x0A0200,ab hier 256 Bytes
```

Nach jedem 256Byte Block wird einmal das Kommando Read Status Register (0x05) ausgeführt.

Der ausgelesene Inhalt findet sich auch in der Datei wieder, die die GP2012 Software beim „Save“ abspeichert.

```
Das "Write" :

0x06 WREN (Write enable)

0x20 SE (Sector erase)
0x08 AD1 (Adresse Teil 1)
0x80 AD2 (Adresse Teil 2)
0x00 AD3 (Adresse Teil 3)

0x05 RDSR (Read Status Register)
```

Solange das Flash beschäftigt ist, kommt 0x03 zurück. Wenn es fertig ist ein 0x00 (dauert ca. 36ms).

Danach 0x06 WREN (Write enable) und dann wird das PageProgramm Kommando geschickt 0x02 0x08 0x80 0x00 sowie die 256 Bytes. Auslesen des 0x05 RDSR (Read Status Register) und solange das Flash beschäftigt ist, kommt 0x03 zurück. Wenn es fertig ist ein 0x00 (dauert hier nur ca. 0,5ms).

Danach liest die Software via FAST READ 0x0B 0x08 0x80 0x00 jeweils 256 Bytes aus, prüft ob korrekt geschrieben wurde.

Dann wiederholt sich das gleiche Spiel mit den anderen Blöcken.

Am Ende wird der Block ab 0x080000 geschrieben und wieder ausgelesen.

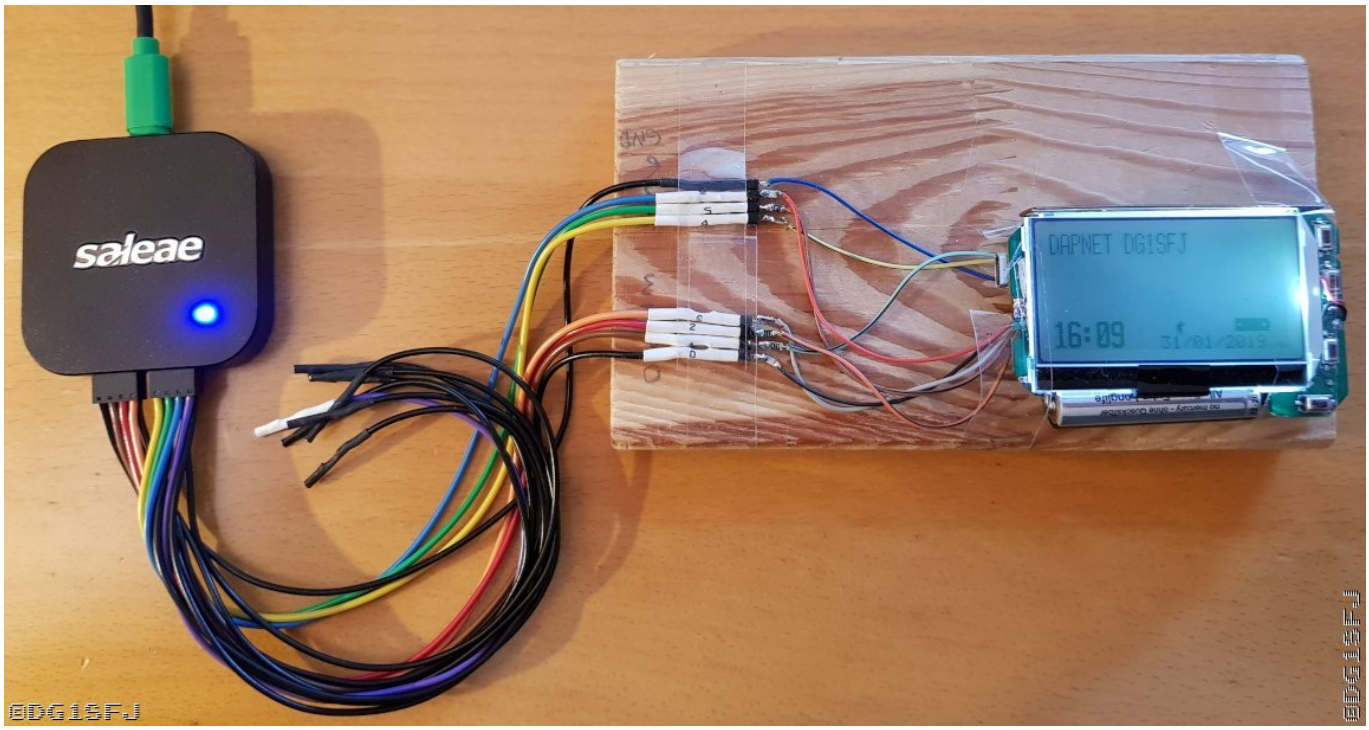
Reihenfolge also zuerst die Botschaften (ab $088000), dann die RIC Konfig (ab $081F00), dann die Pager Konfig (ab $080000).

EEPROM Kommunikation

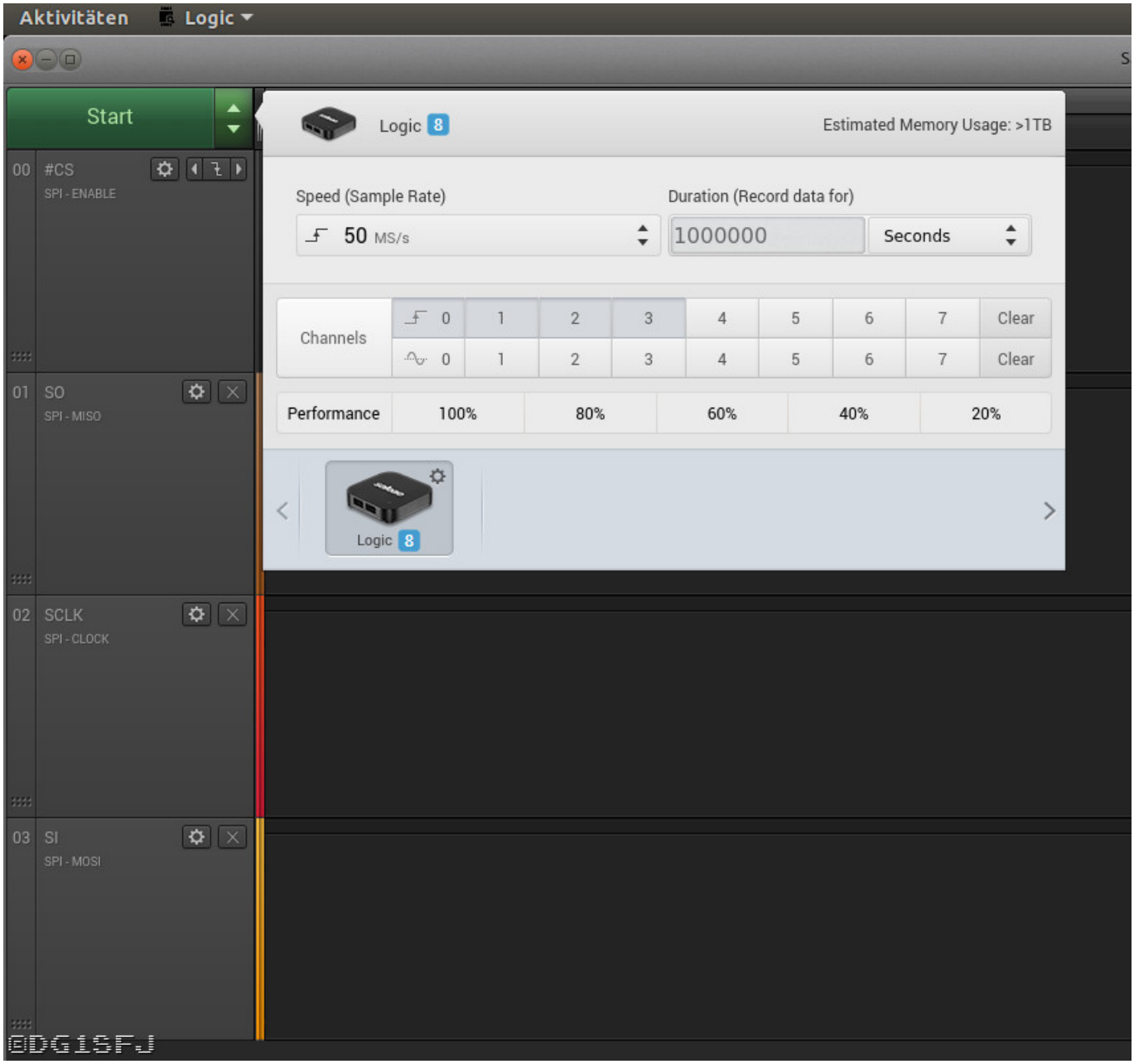
Möchte man dem Pager bei der Arbeit zuschauen reichen ein paar Drähte vom EEPROM zu einem Logic Analyzer (Saleae Logic 8 in meinem Fall).

```
Kanal 0 : Chipselect (SPI Enable)

Kanal 1 : SO - MISO
```

```
Kanal 2 : SCLK - Clock

Kanal 3 : SI - MOSI
```

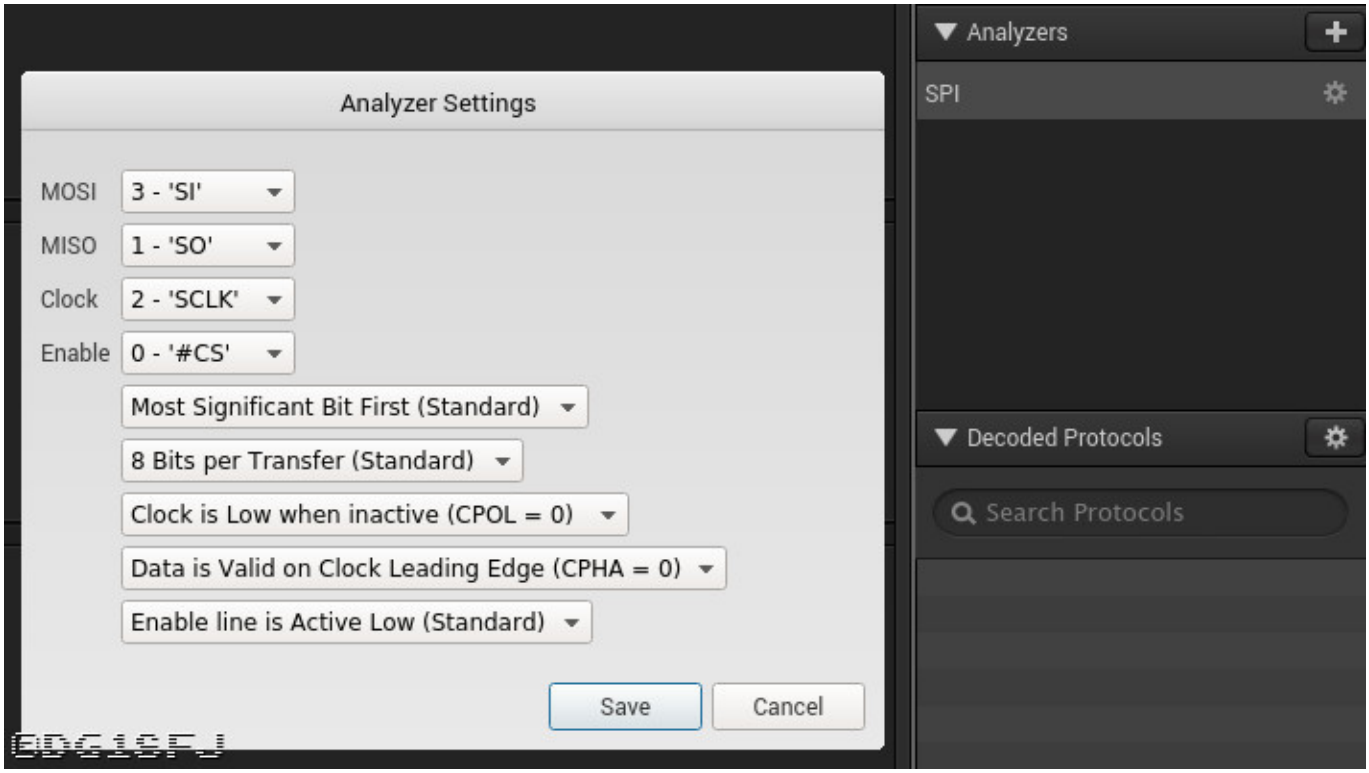


Kleiner Tip noch : Leitungen nicht direkt anschließen sondern noch je 470Ohm in Serie schalten (sieht man hier als SMD Widerstände). Dann ist die Belastung durch den LogicAnalyzer nicht so groß. Und natürlich die Masse-Leitung nicht vergessen...

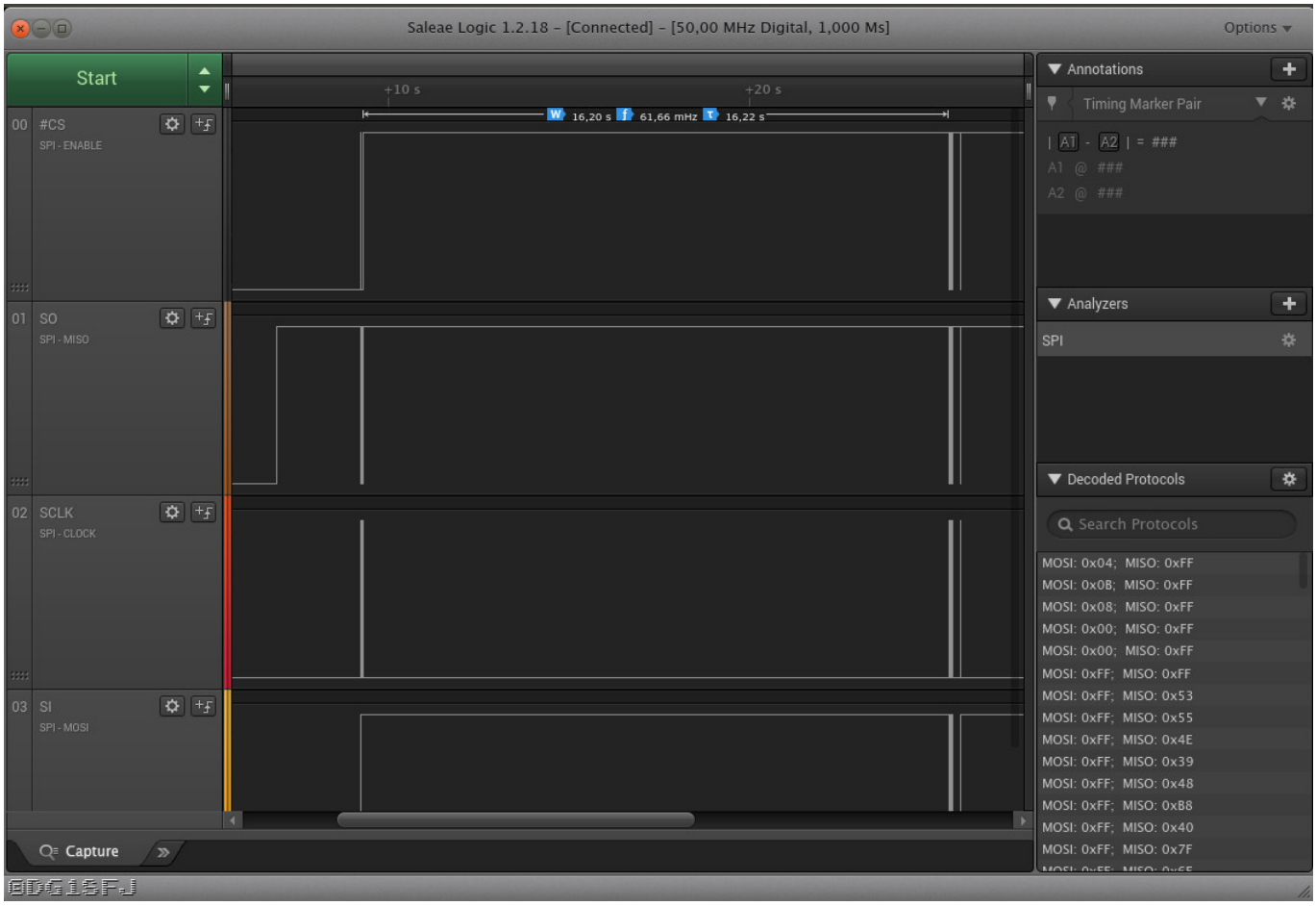


Trigger dann einstellen ob sofort oder erst bei fallender Flanke des Chipselects.

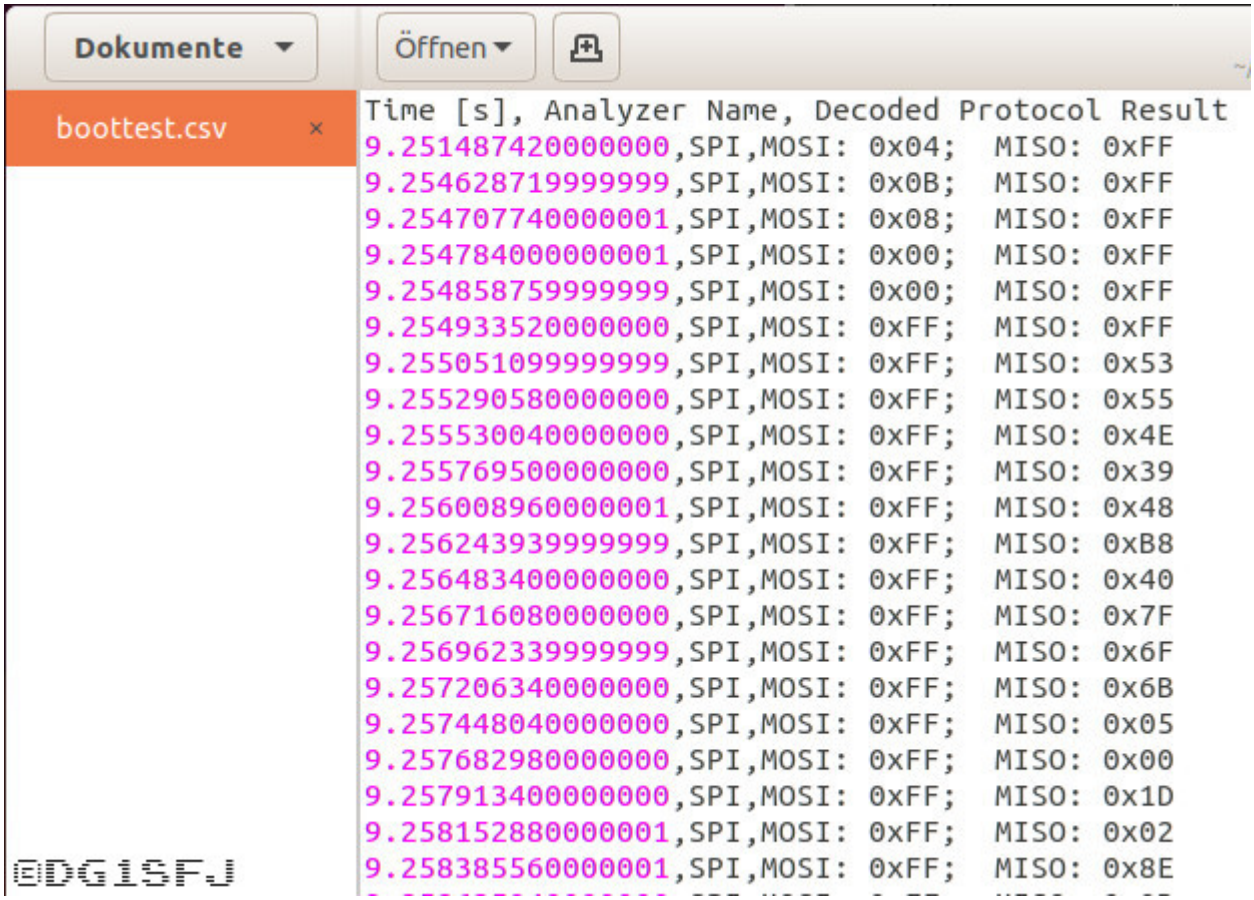
Als nächstes muss der SPI Analyzer aktiviert werden mit folgenden Einstellungen :



Dann kann es auch schon losgehen. Hier als Beispiel beim einlegen der Batterie die ersten 30 Sekunden mal mit geschrieben. Man sieht auch rechts das der Analyzer die SPI dekodiert hat.



Die dekodierten SPI-Kommandos lassen sich dann in einer *.csv Datei exportieren :



Das ist leider sehr unübersichtlich und wenig lesbar für den Menschen ... Anhand des Datenblattes habe ich mir einen Parser in Python geschrieben der die einzelnen Kommandos an das EEPROM dekodieren und lesbar wieder herrausschreiben kann. Folgende Kommandos werden vom Pager benutzet (und nur die habe ich auch umgesetzt) :

COMMAND DESCRIPTION

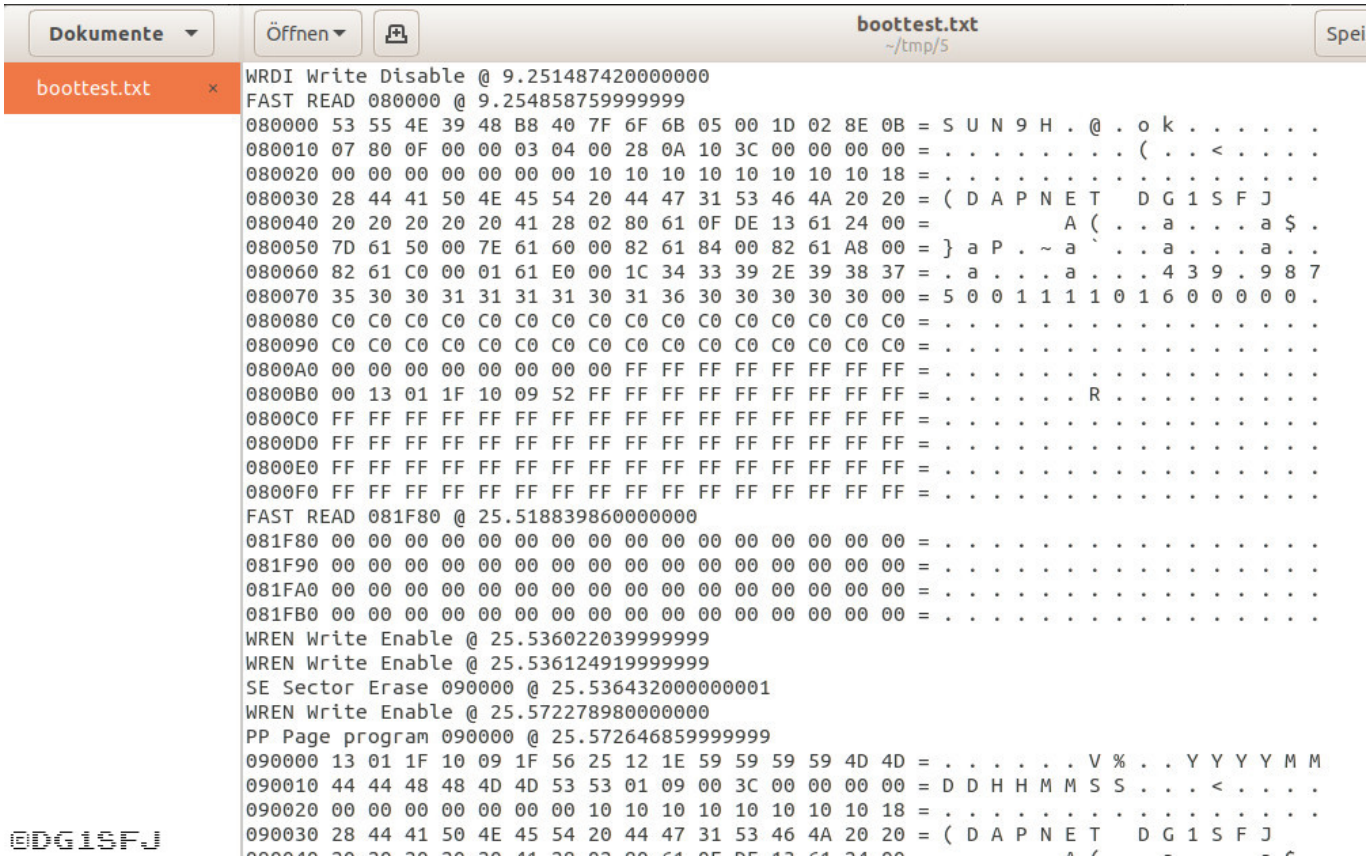
Table 4. COMMAND DEFINITION

Command (byte)	WREN (write enable)	WRDI (write disable)	WRSR (write status register)	RDID (read identific-ation)	RDSR (read status register)	READ (read data)	FAST READ (fast read data)
1st byte	06 (hex)	04 (hex)	01 (hex)	9F (hex)	05 (hex)	03 (hex)	0B (hex)
2nd byte						AD1	AD1
3rd byte						AD2	AD2
4th byte						AD3	AD3
5th byte							Dummy
Action	sets the (WEL) write enable latch bit	resets the (WEL) write enable latch bit	to write new values to the status register	outputs JEDEC ID: 1-byte Manufact-urer ID & 2-byte Device ID	to read out the values of the status register	n bytes read out until CS# goes high	n bytes read out until CS# goes high

Command (byte)	RDSFDP (Read SFDP)	RES (read electronic ID)	REMS (read electronic manufacturer & device ID)	DREAD (Double Output Mode command)	SE (sector erase)	BE (block erase)	CE (chip erase)
1st byte	5A (hex)	AB (hex)	90 (hex)	3B (hex)	20 (hex)	52 or D8 (hex)	60 or C7 (hex)
2nd byte	AD1	x	x	AD1	AD1	AD1	
3rd byte	AD2	x	x	AD2	AD2	AD2	
4th byte	AD3	x	ADD *(Note 1)*	AD3	AD3	AD3	
5th byte	Dummy			Dummy			
Action	Read SFDP mode	to read out 1-byte Device ID	output the Manufacturer ID & Device ID	n bytes read out by Dual Output until CS# goes high	to erase the selected sector	to erase the selected block	to erase whole chip

Command (byte)	PP (page program)	RDSCUR (read security register)	WRSCUR (write security register)	ENSO (enter secured OTP)	EXSO (exit secured OTP)	DP (Deep power down)	RDP (Release from deep power down)
1st byte	02 (hex)	2B (hex)	2F (hex)	B1 (hex)	C1 (hex)	B9 (hex)	AB (hex)
2nd byte	AD1						
3rd byte	AD2						
4th byte	AD3						
5th byte							
Action	to program the selected page	to read value of security register	to set the lock-down bit as "1" (once lock-down, cannot be updated)	to enter the 512 bit secured OTP mode	to exit the 512 bit secured OTP mode	enters deep power down mode	release from deep power down mode

Nach dem Durchlauf mit dem Python Programm sieht das ganze schon schöner aus :



Damit lassen sich dann alle Interaktionen sauber aufschlüsseln :

- Booten

- Empfang von Nachrichten

- Lesen mit dem GP Steuerprogramm

- Schreiben mit dem GP Steuerprogramm

Beim Vergleich der Kommunikation der alten (V1.01) und der neuen (V1.70) Software konnte ich keinen Hinweis finden das diese Software eine Firmware in den Pager programmiert. Insofern ist das berühmte „Antennen-Symbol“ vermutlich nur ein ein/ausknipsen eines Bits irgendwo in der Config....