

# Konzept ADS-B Viewer

9. September 2021

## Designentscheidungen

### ADS-B Empfänger

#### Hardware

- RTL-SDR Dongle
- Raspberry-Pi

#### Software

- Änderung rtl\_adsb.c: Die geänderte Quelle (rtl\_adsb-mein-raspi.c) und die Original-Quelle (rtl\_adsb.c) sind im Verzeichnis „D:\VBSharedFolder\Debian\rtl-adsb changed for RasPi“ abgelegt. Die Änderung umfasst Entfernen der Windows-Code-Anteile und Hinzufügen einer Zeitmarke zu jeder empfangenen ADS-B Message. Diese Änderung ist mit der Zeichenkette  
////////////////////////////////////  
umschlossen. Die Originalquelle ist Teil des Projektes „D:\Projekte\RTL-SDR“ hier auf der Festplatte und „git://git.osmocom.org/rtl-sdr.git“ im Netz.
- Kommandozeile Raspberry-Pi: while true; do ( (echo HTTP/1.0 200 ok; echo; rtl\_adsb -S -Q2; ) | netcat -l 1234); done
- Informationen zur Raspberry-Pi Representanz finden sich im README unter „D:\VBSharedFolder\Debian\rtl-adsb changed for RasPi“.
- ToDo:
  - Erkennung der ADS-B Meldungen verbessern (pufferübergreifende Auswertung)
  - Fehlerkorrekturen von [2] übernehmen

## Visualisierung

#### Hardware

- ThinkPad X1 Laptop Windows 10

#### Software

- C++
- MFC mit Document/View-Konzept benutzen. Immer nur ein Dokument. Zunächst mit nur eine Ansicht auf die Rawmessages und später eine Ansicht der dekodierten ADS-B Meldungen.

The view is responsible for displaying and modifying<sup>1</sup> the document's data but not for storing it.<sup>2</sup> The document provides the view with the necessary details<sup>3</sup> about its data. You can let the view access the document's data members directly, or you can provide member functions in the document class for the view class to call.

When a document's data changes, the view responsible for the changes typically calls the `CDocument::UpdateAllViews` function for the document, which notifies all the other views by calling the `OnUpdate` member function for each. The default implementation of `OnUpdate` invalidates the view's entire client area. You can override it to invalidate only those regions of the client area that map to the modified portions of the document.

- LibCURL für den Netzwerkzugriff benutzen
- Designdetails:
  - Das Document erzeugt einen Worker-Thread (in der Klasse `ADSBThread`), der den ADS-B Rohdatenstrom von der Netzwerkschnittstelle liest.
  - Der Rohdatenstrom kommt in Blöcken variabler Länge an. Die enthaltenden Informationen erstrecken sich evtl. über die Blockgrenzen hinweg. Dies muss beim Parsen (in der Klasse `CombineChunks`) berücksichtigt werden um keine Informationen zu verlieren.
  - Nach jedem neu ankommenden Chunk des Rohdatenstromes werden die darin enthaltenen Rawmessages geparkt, d.h. in Zeitmarken und Rawmessages zerlegt, und in einer Queue des Documents gespeichert. Alle empfangenen ADS-B Messages sind in dieser Queue persistent.
  - Die Klasse `ADSBThread` erbt von der Parser-Klasse `CombineChunks`.
  - Die neu ankommenden ADS-B Messages werden mit der Message `ADSB_MESSAGE_ID` zum View-Window gesendet, dort mit `OnUser()` empfangen, und visualisiert.
  - Dazu wird die View von `CListView` abgeleitet und mittels der `AddItem`-Methode die neuen Queueelemente hinzugefügt.
  - Um Speicherlecks beim Beenden der Applikation zu vermeiden, wird dem Thread im Destructor des Documents signalisiert, dass er sich beenden soll (durch setzen von `loopBreak=true`) dann wird auf das Event `threadFinished` gewartet.
  - Dekodierung der ADS-B Rawmessages: siehe [2]
- Softwaretest: Die pufferübergreifende Auswertung der Informationsblöcke ist komplex. Zum Test habe ich ein Tool geschrieben: `ADSBTest`. Dieses Tool könnte auch für die Empfangsseite genutzt werden.

## Analyse rtl-adsb

- Die Samplezeit ist  $0.5 \mu s$ , die Sampleauflösung ist 8 Bit.
- Die Sampledaten vom Demodulator kommen in einem Byte-Puffer der Länge  $16 * 16384 = 262144$  an.
- Jeder Samplezeitpunkt wird durch 2 Bytes beschrieben: i und q.
- 1. Schritt: die beiden Bytes in ein Wort wandeln:  $i^2 + q^2$ . Die Länge des Puffers reduziert sich dadurch auf 131072 Worte.
- Funktion `manchester()`:
  - Die Wörter der Puffers werden nacheinander daraufhin untersucht, ob sie eine  $8 \mu s$ , d.h. 16 Worte lange Präambel [2] enthalten.
  - Die Präambel ist durch folgende Bit-Abfolge definiert: 1010000101000000, d.h. die Bits 0,2,7 und 9 müssen 1 sein, der Rest 0.
  - Diese Prüfung wird durch die Funktion `preamble()` vorgenommen. Eine 1 wird returned, falls eine Präambel gefunden wurde, sonst 0.
  - Zum Testen der Funktionalität habe ich 2 Dateien generiert: Rawmessages in Textform "ADSB-Messages.txt" und die dazugehörigen binären Rohsamples "ADSB-Dump.bin". Die beiden Dateien befinden sich im Verzeichnis `Projekte/rtl-sdr`.

---

<sup>1</sup>gemeint ist: vom User!

<sup>2</sup>gemeint ist: in eine Datei speichern!

<sup>3</sup>also z.B. die Dekodierung im Dokument machen!

# ADS-B Datenformatbeschreibung

DABS ist die Abkürzung von “Discrete Address Beacon System”.

Es gibt die folgenden Downlink Formate:

DF	Beschreibung
0	Airborne Collision Avoidance System (ACAS), Traffic Alert and Collision Avoidance System (TCAS)
1-3	Reserviert
4	Antworten auf Abfragen von bodengestützten Sekundärradaranlagen, Surveillance altitude
5	Antworten auf Abfragen von bodengestützten Sekundärradaranlagen, Surveillance ident
6-10	Reserviert
11	All-Call Reply (Acq. Squitter if II=0)
12-15	Reserviert
16	Long air-air surveillance (ACAS)
17	Extended Squitter
18	Extended Squitter, supplementary
19	Militärischer Extended Squitter
20	Comm-B, Mode-S EHS, Altitude
21	Comm-B, Mode-S EHS, Ident
22	militärisch genutzt
23	Reserviert
24	Comm-D Extended Length Message (ELM), Format siehe [3]

## DF17/DF18

## DF20/DF21

Diese beiden Downlink-Formate sind Antworten, die von der Bodenstation aktiv abgefragt werden. Welche Daten übertragen werden hängen von einem BDS (Binary Data Selector) ab, der aber nicht in der Antwort enthalten ist.

Es gibt die folgenden BDS-Codes:

BDS 2,0	Aircraft identification
BDS 2,1	Aircraft and airline registration markings
BDS 4,0	Selected vertical intention
BDS 4,4	Meteorological routine air report
BDS 5,0	Track and turn report
BDS 6,0	Heading and speed report

Die DF20/21 Messages sind wie folgt strukturiert:

Bezeichnung	Bits	Bedeutung
DF	5	Downlink Format
FS	3	Flight Status
DR	5	Downlink Request
UM	6	Utility Message
AC/ID	13	Altitude Code (DF20) or Identity (DF21)
MB	56	Comm-B Message
AP/DP	24	Address Parity/Data Parity

## Aircraft identification BDS 2,0

Das Comm-B Messagefeld MB hat die folgende Struktur:

Inhalt	0x20	C8	C7	C6	C5	C4	C3	C2	C1
Bits	8	6	6	6	6	6	6	6	6

Die jeweils 6 Bits  $b_n$ ,  $n = 0, \dots, 5$  eines Zeichens  $C_m = (b_5, \dots, b_0)$ ,  $m = 1, \dots, 8$  werden als 0-basierter Index  $\sum_{n=0}^5 (2^n b_n)$  in die folgende Lookup-Tabelle interpretiert:

“#ABCDEFGHJKLMNPOQRSTUVWXYZ#####\_#####0123456789#####”.

Das #-Zeichen bezeichnet einen ungültigen Index, das ␣-Zeichen ist das Leerzeichen.  
Man erhält die Aircraft Identification die Zeichenkette C8,C7,C6,C5,C4,C3,C2,C1.

## Parity Coding

Für ADS-B wird ein pseudo-zyklischer Code zur Bestimmung der Parität benutzt.

Das Generatorpolynom dieses Codes ist durch

$$G(x) = x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^3 + x$$

gegeben [5], es gilt  $x \in \{0, 1\}$ .

## Polynom-Multiplikation

Ein auf Matrizen basierende Formulierung von Polynom-Multiplikationen/Divisionen ist z.B. in [6] beschrieben und soll im folgenden benutzt werden.

Die Menge der Polynome über dem Körper  $K$  und des Grades  $n$  sei durch  $P_n = \{p_0 + p_1x + \dots + p_nx^n, x \in K, p_n \in K\}$  bezeichnet. Die  $n+1$  Koeffizienten  $p_n$  der Elemente aus  $P_n$  können jeweils zu Spaltenvektoren  $\underline{p} = (p_0, \dots, p_n)^T$  zusammengefasst werden.

Für zwei Polynome  $a(x) = a_0 + a_1x + \dots + a_nx^n \in P_n$  und  $b(x) = b_0 + b_1x + \dots + b_mx^m \in P_m$  erhält man für den Koeffizientenvektor ihres Produktes  $a(x)b(x) \in P_{(n+m)}$  die Beziehung

$$\underline{p}^{ab} = C_a \underline{b}.$$

Dabei ist  $C_a^m$  die Faltungsmatrix der Koeffizienten von  $a(x)$  für ein Produkt mit einem Polynom des Ranges  $m$

$$C_a^m = \begin{pmatrix} a_0 & & & \\ \vdots & \ddots & & \\ a_n & \vdots & a_0 & \\ & \ddots & \vdots & \\ & & & a_n \end{pmatrix} \in K^{(n+m+1) \times (m+1)}.$$

Natürlich gilt ebenso  $\underline{p}^{ab} = \underline{p}^{ba}$  mit

$$\underline{p}^{ba} = C_b^n \underline{a}$$

und

$$C_b^n = \begin{pmatrix} b_0 & & & \\ \vdots & \ddots & & \\ b_m & \vdots & b_0 & \\ & \ddots & \vdots & \\ & & & b_m \end{pmatrix} \in K^{(n+m+1) \times (n+1)}.$$

## Polynom-Division

Ganz ähnlich kann man für  $n > m$  die Division zweier Polynome, nämlich eines Zählerpolynoms  $b(x) \in P_n$  und eines Nennerpolynoms  $a(x) \in P_m$  behandeln

$$\frac{b(x)}{a(x)} = q(x) + \frac{r(x)}{a(x)}$$

dabei ist  $q(x) \in P_{(n-m)}$  der Quotient und  $r(x) \in P_{(m-1)}$  der Divisionsrest.

Durch Multiplikation mit  $a(x)$  erhält man das Polynom

$$b(x) = a(x)q(x) + r(x)$$

und für seinen  $(n+1)$ -dimensionalen Koeffizientenvektor folgt mit dem Ergebnis von oben

$$\underline{b} = C_a^{(n-m)} \underline{q} + \begin{pmatrix} \underline{r} \\ \underline{0} \end{pmatrix}$$

mit der Faltungsmatrix  $C_a \in K^{(n+1) \times (n-m+1)}$  und dem  $(n-m+1)$ -dimensionalen Nullvektor  $\underline{0}$ .

Äquivalent umgeformt ergibt sich daraus

$$\underline{b} = C_a^{(n-m)} \underline{q} + D \underline{r}$$

und weiter

$$\underline{b} = \begin{pmatrix} D, & C_a^{(n-m)} \end{pmatrix} \begin{pmatrix} \underline{r} \\ \underline{q} \end{pmatrix}.$$

Dabei ist

$$D = \begin{pmatrix} E \\ 0 \end{pmatrix} \in K^{(n+1) \times m}$$

eine Matrix,  $E$  die  $(m \times m)$ -Einheitsmatrix und  $0$  die  $((n-m+1) \times m)$ -dimensionale Nullmatrix.

Setzt man nun  $\underline{b}$  als bekannt voraus, so kann man die Koeffizientenvektoren  $\underline{r}$  und  $\underline{q}$  bestimmen, da die obere Dreiecksmatrix  $\begin{pmatrix} D, & C_a^{(n-m)} \end{pmatrix}$  unter der Annahme  $a_n \neq 0$  invertierbar ist.

Damit hat man die Polynomdivision durchgeführt und es gilt

$$\begin{pmatrix} \underline{r} \\ \underline{q} \end{pmatrix} = \begin{pmatrix} D, & C_a^{(n-m)} \end{pmatrix}^{-1} \underline{b}.$$

Die zu invertierende Matrix ist als obere Dreiecksmatrix einfach invertierbar.

## Voraussetzungen

Um die mathematische Methode der Polynomdivision für das Paritätsproblem nutzen zu können, muss der Menge  $\{0, 1\}$  eine Körperstruktur  $K$  zugeordnet werden können. Im Anhang wird bewiesen, dass dies möglich ist, wenn man mit der Körperaddition  $+$  die Exklusiv-Oder-Operation  $\oplus$  identifiziert und mit der Körpermultiplikation  $\circ$  die Und-Operation  $\wedge$ . Damit ist dann  $K = \mathbb{B}$ .

# Anhang

## Körpereigenschaften

Eine Menge  $\Pi$  bildet einen Körper, wenn auf ihr die zweistelligen Operationen der Addition  $+$  und der Multiplikation  $\circ$  definiert sind und die drei Eigenschaften

1.  $(\Pi, +)$  muss eine kommutative Gruppe mit dem neutralen Element  $e = 0$  sein, d.h. es muss gelten
  - Assoziativgesetz:  $\forall a, b, c \in \Pi$  gilt  $a + (b + c) = (a + b) + c$ .
  - Kommutativgesetz:  $\forall a, b \in \Pi$  gilt  $a + b = b + a$ .

- Neutrales Element  $e$ :  $\forall a \in \Pi$  gilt  $a + e = a$ .
  - Inverses Element:  $\forall a \in \Pi$  gibt es ein inverses Element  $a^{-1}$  mit  $a + a^{-1} = e$ .
2.  $(\Pi \setminus \{0\}, \circ)$  muss eine kommutative Gruppe mit dem neutralen Element  $e = 1$  sein, d.h. es muss gelten
- Assoziativgesetz:  $\forall a, b, c \in \Pi$  gilt  $a \circ (b \circ c) = (a \circ b) \circ c$ .
  - Kommutativgesetz:  $\forall a, b \in \Pi$  gilt  $a \circ b = b \circ a$ .
  - Neutrales Element  $e$ :  $\forall a \in \Pi \setminus \{0\}$  gilt  $a \circ e = a$ .
  - Inverses Element:  $\forall a \in \Pi \setminus \{0\}$  gibt es ein inverses Element  $a^{-1}$  mit  $a \circ a^{-1} = e$ .
3. Distributivgesetze:  $\forall a, b, c \in \Pi$  gilt  $a \circ (b + c) = a \circ b + a \circ c$  und  $(a + b) \circ c = a \circ c + b \circ c$  erfüllt sind.

## Behauptungen

1. Identifiziert man mit der Addition  $+$  die Exklusiv-Oder-Operation  $\oplus$ , so ist  $(\Pi, +)$  eine kommutative Gruppe mit dem neutralen Element 0.
2. Identifiziert man mit der Multiplikation  $\circ$  die Und-Operation  $\wedge$ , so ist  $(\Pi \setminus \{0\}, \circ)$  eine kommutative Gruppe mit dem neutralen Element 1.
3. Ausserdem gelten die beiden Distributivgesetze für die Körperelemente.

## Beweise

### Beweis der Behauptung 1

Man erhält für das Assoziativgesetz  $a \oplus (b \oplus c) = (a \oplus b) \oplus c$  und für das Kommutativgesetz  $a \oplus b = b \oplus a$ .

Mit der Wahrheitstabelle 1 der Exklusiv-Oder-Operation und den beiden Tabellen 2 und 3 sieht man, dass das Assoziativgesetz erfüllt ist.

Das Kommutativgesetz ist erfüllt, wie man der Wahrheitstabelle 1 direkt entnehmen kann.

Für das neutrale Element 0 gilt die Beziehung  $a \oplus 0 = a$  (siehe 1).

Das inverse Element zu 0 ist 0, da  $0 \oplus 0 = 0$  gilt, das inverse Element zu 1 ist 1, da  $1 \oplus 1 = 0$  gilt. Die Elemente der Menge  $\Pi$  sind selbstinvers bezüglich der Addition  $\oplus$ , also  $\forall a \in \Pi \mid a \oplus a = 0$ .

$a \oplus b$	a	b
0	0	0
1	1	0
1	0	1
0	1	1

Tabelle 1: Wahrheitstabelle Exklusiv-Oder (XOR)

$a \oplus (b \oplus c)$	$b \oplus c$	a	b	c
0	0	0	0	0
1	0	1	0	0
1	1	0	1	0
0	1	1	1	0
1	1	0	0	1
0	1	1	0	1
0	0	0	1	1
1	0	1	1	1

Tabelle 2: zum Assoziativgesetz der Exklusiv-Oder-Operation (XOR)

$(a \oplus b) \oplus c$	$a \oplus b$	a	b	c
0	0	0	0	0
1	1	1	0	0
1	1	0	1	0
0	0	1	1	0
1	0	0	0	1
0	1	1	0	1
0	1	0	1	1
1	0	1	1	1

Tabelle 3: zum Assoziativgesetz der Exklusiv-Oder-Operation (XOR)

Damit sind alle notwendigen Eigenschaften der Addition nachgewiesen und  $(\mathbb{Z}_2, +)$  ist eine kommutative Gruppe mit dem neutralen Element 0. ■

## Beweis der Behauptung 2

Man erhält für das Assoziativgesetz  $a \wedge (b \wedge c) = (a \wedge b) \wedge c$  und für das Kommutativgesetz  $a \wedge b = b \wedge a$ . Beide Gesetze sind erfüllt (siehe [7]).

Für das neutrale Element 1 gilt die Beziehung  $1 \wedge 1 = 1$  (siehe 4).

Das inverse Element zu 1 ist 1, da  $1 \wedge 1 = 1$  gilt, ein inverses Element zu 0 existiert natürlich nicht.

$a \wedge b$	a	b
0	0	0
0	1	0
0	0	1
1	1	1

Tabelle 4: Wahrheitstabelle der Und-Operation (AND)

Damit sind alle notwendigen Eigenschaften der Multiplikation nachgewiesen und  $(\mathbb{Z}_2, \circ)$  ist eine kommutative Gruppe mit dem neutralen Element 1. ■

## Beweis der Behauptung 3

Für die beiden Distributivgesetze für die Körperelemente erhält man mit den Operationen  $\oplus$  und  $\wedge$  die beiden Bedingungen

$$a \wedge (b \oplus c) = a \wedge b \oplus a \wedge c$$

und

$$(a \oplus b) \wedge c = a \wedge c \oplus b \wedge c.$$

$a \wedge (b \oplus c)$	$b \oplus c$	a	b	c
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
1	1	1	1	0
0	1	0	0	1
1	1	1	0	1
0	0	0	1	1
0	0	1	1	1

Tabelle 5: zur Behauptung 3, 1. Distributivgesetz

$a \wedge b \oplus a \wedge c$	$a \wedge b$	$a \wedge c$	a	b	c
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
1	1	0	1	1	0
0	0	0	0	0	1
1	0	1	1	0	1
0	0	0	0	1	1
0	1	1	1	1	1

Tabelle 6: zur Behauptung 3, 1. Distributivgesetz

$(a \oplus b) \wedge c$	$a \oplus b$	a	b	c
0	0	0	0	0
0	1	1	0	0
0	1	0	1	0
0	0	1	1	0
0	0	0	0	1
1	1	1	0	1
1	1	0	1	1
0	0	1	1	1

Tabelle 7: zur Behauptung 3, 2. Distributivgesetz

$a \wedge c \oplus b \wedge c$	$a \wedge c$	$b \wedge c$	a	b	c
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	1	1	0
0	0	0	0	0	1
1	1	0	1	0	1
1	0	1	0	1	1
0	1	1	1	1	1

Tabelle 8: zur Behauptung 3, 2. Distributivgesetz

Die beiden Distributivgesetze sind erfüllt, wie man den Wahrheitstabellen 5 bis 8 entnehmen kann. ■

## Definition

Die Menge  $\mathbb{B}$  mit der Addition  $\oplus$  und der Multiplikation  $\wedge$  wird als Körper der binären Zahlen  $\mathbb{B}(\mathbb{B}, +, \circ)$  bezeichnet. Der Körper  $\mathbb{B}$  hat die Charakteristik 1.

## Literatur

- [1] <https://osmocom.org/projects/rtl-sdr/wiki>
- [2] <https://github.com/watson/libmodes>
- [3] <https://www.radartutorial.eu/13.ssr/sr24.de.html>
- [4] <https://github.com/junzis/pyModeS/blob/master/pyModeS/decoder/common.py>
- [5] Fundamentals of Mode S Parity Coding; J. L. Gertz; Project Report ATC-117



- [6] Polynomial Division By Convolution; Feng Cheng Chang; Applied Mathematics E-Notes, 11(2011), 249–254
- [7] [https://de.wikipedia.org/wiki/Boolesche\\_Algebra](https://de.wikipedia.org/wiki/Boolesche_Algebra)