ADS-B Aircraft Monitoring

Hans-Gerhard Gross Kai Warendorf

Tuesday 2nd September, 2014

Abstract

ADS-B is a new approach to aircraft monitoring by processing VHF radio transmitted messages. The messages are sent by each aircraft according to predefined pattern. Each message type represents particular information about identification, position, and direction.

This document describes the ADS-B domain, and it is intended to help with the decoding of messages and their correct interpretation.

1 Introduction

The *Automatic Dependent Surveillance Broadcast* (ADS-B) represents the future generation aircraft tracking method, and it will, eventually, replace ground-based radar as most important data source for situational awareness in avionics. From 2017 on, all aircraft above 5.700 Kg, or with a maximum cruise speed above 250 knots will have to be equipped with ADS-B transponders. This widespread use of ADS broadcasting is expected to improve safety and efficiency in avionics considerably [1].

ADS-B enables air-to-air, air-to-ground, and ground-to-ground communication between all stakeholders equipped with suitable transceivers, which use the 1090 MHZ VHF radio frequency. ADS-B follows a periodic broadcasting scheme of so-called extended squitter messages, carriying information about an aircraft's identification, its current position, altitude, speed, and, in the future, also its intent.

2 ADS-B Base Station

Because of its close proximity to Stuttgart Airport, and its location in the main flight path to the airport, the University of Esslingen operates an ADS-B base station that provides the received messages through a web server. This server can be accessed from within the university's network[1] via VPN (or form the Lab computers).

[1]URI: www.flugmon-it.hs-esslingen.de/subscribe/ads.sentence

The ADS-B setup used by the university is the low-budget GNS 5890 ADS-B Receiver USB Stick from Global Navigation Systems [2]. This is linked via USB-serial connection to a low power linux box that timestamps the received messages and publishes them through the web server.

The web server provides the messages in a Javascript Object Notation (JSON) format.[2] The JSON comes from the Redis[3] publish-subscribe middleware which is realizing the web-based message distribution, and is using this format. The JSON sentence shown below is an example as it is coming from the web server.

```
{"subscribe":["message","ads.sentence","1379574427.9127481!ADS-B*8D40675258BDF05CDBFB59DA7D6F;\r\n"]}
```

The JSON sentence could be parsed by appropriate JSON libraries and the content of the ADS-B sentence retrieved. However, since the structure of the sentence is extremely simple, and, moreover, it is fixed, it is probably easier to simply extract the substring carrying the relevant information, e.g. determine the position of the '!' symbol and extract a fixed number of characters on each side. The following example show the ADS-B sentence in the shape as it is coming from the base station.

```
1379574427.9127481!ADS-B*8D40675258BDF05CDBFB59DA7D6F;
```

The number at the beginning represents the time of message reception in the base station in seconds since the *epoch*[4] (left-hand side of the dot), with microsecond accuracy (right-hand side of the dot). The timestamp is relevant in order to assess network delays, so that approriate measures may be taken, in order to re-adjust the true locations of moving aircraft.

The following string shows the real sentence as received by the GNS.

```
*8D40675258BDF05CDBFB59DA7D6F;
```

3 ADS-B Sentence Decoding

Each ADS-B raw extended squitter sentence is comprised of a starting and trailing symbol, i.e. '*' and ';', respectively. A sentence is made up of 28 half-byte nibbles, or 14 bytes in hex representation, leading to 112 bit long messages. The first step in decoding is the knowlege about the originator of the message and the type of the message sent, i.e. who sent the message and what does it contain? The message type determines how the message content should be further treated and interpreted.

[2]http://www.json.org
[3]http://redis.io
[4]https://en.wikipedia.org/wiki/Unix_epoch

The first byte of any sentence contains the *downlink format* (DF) and the *capability* CA. The next 3 bytes [2..4] represent the unique aircraft identification code issued by the International Civil Aviation Organization, the so-called ICAO code. This identifies the sender of the sentence. Bytes [5..11] contain the actual payload of the sentence, including the type of the message. Bytes [12..14] contain the parity number, used in order to check the correct transmission of the sentence. The following raw sentence

```
8D3C6DD6581F97E703EBAB40067F
```

must then be split up according to the following organization.

```
1        2, 3, 4    5, 6, 7, 8, 9, 10, 11    12, 13, 14  [BYTES]
8D       3C6DD6     581F97E703EBAB           40067F
DF+CA    ICAO       ADS-B PAYLOAD            PARITY
```

The first byte contains two individual pieces of information, i.e. the downlink format, bits [0..4], indicating the type of squitter sentence, and the capability of the transmitter, bits [5..7]. By transforming the first byte into its binary representation, the two bit blocks can be extracted individually and decoded, in the following way shown below, indicating a standard 112 bit extended binary squitter sentence (17, 5). The bit order of the binary encoding, and this is the case for all ADS-B encodings, is most significant bit (MSB, left) to least significant bit (LSB, right).

```
8D                 [hex]
10001     101      [binary]
17        5        [decimal]
DF        CA
```

The ICAO code, representing the originating aircraft that sent the sentence, can be directly looked up in appropriate public databases.[5] This particular example, 3C6DD6, refers to a Germanwings Airbus A-319, with registration D-AKNV.

4 Extended Binary Squitter Payload Decoding

The ADS-B payload, i.e. the content of an ADS-B sentence, is encoded in bytes [5..11] of the original hex-encoded sentence, in total 56 bit long. The first step in decoding the payload, is the idenfication of the message type. This is encoded in the first 5 bits [0..4] of the payload. In order to being able to access invidiual bits of the payload, it must be transformed into a binary string. This is demonstrated in the example below.

[5]e.g. http://www.airframes.org

```
DF+CA+ICAO        Hex Payload       Parity
8D 4B16A3         587DD7DA03F289    20503C
Binary Payload
01011000011111011101011111011010000000111111001010001001
Type = 01011 = 11
01011 000011111011101011111011010000000111111001010001001
```

The 5 bits can then be treated as a single number, in this case binary "01011" = 11 decimal, i.e. an Airborne Position message of a BelAir Airbus A-319 with the registration HB-IOX.

4.1 Message Types

The ADS-B standard [3] defines 32 distinct message types [0..31]. They are summarized in Table 1. Some of the messages are not yet used, because the standard is evolving and some of the information is not yet readily available in onboard transmission systems, e.g. trajectory change, or target state. Other messages can only be received in the direct proximity of airports, e.g. surface position. Airborne position and velocity messages are typical, and can be observed everywhere, as well as aircraft identification messages, i.e. types 1..4, 9..18, 20..22, and, sometimes, also test messages, type 23.

4.2 Structure of Static and Dynamic Message Types

It is important to note that the encoding of the first five bits of the binary payload string, i.e. bit 0..4, is always referring to the type of the message. This is consistent throughout all messages. For some of the messages bit 5..7 is also indicating a subtype, but this is not consistent for all messages types. The remainder of the binary payload string, i.e. bit 5..55, is interpreted differently according to each message type. These individual encodings are summarized in the following paragraphs.

Aircraft Identification Message, Type 1..4

- Type code, 5 bits, bit 0..4: indicating the type of the message.
- Emitter category, 3 bits, bit 5..7: indicates the ADS-B emitter category of an aircraft.
- Identification character 1-8 (six bit for each character), 48 bits, bits 8..55: the eight aircraft identification characters contain either flight number, or the aircraft idenfication code, or the radio call sign. They are sixbit-encoded ASCII characters. Table 2 summarizes the sixbit coding.

Table 1: Summary of ADS-B Extended Squitter Message Types

Type Code bit 0..4	Subtype Code bit 5..7	ADS-B Message Type bit 8..55
0	Not present	Airborne Position Message; Surface Position Message
1-4	Not present	Aircraft Identification and Category Message
5-8	Not present	Surface Position Message
9-18	Not present	Airborne Position Message
19	0	Reserved for future use
	1-4	Airborne Velocity Message
	5-7	Reserved for future use
20-22	Not present	Airborne Position Message
23	0	Test Message
	1-7	Reserved for future use
24	0	Reserved for future use
	1	Surface System Status
	2-7	Reserved for future use
25-26	Not present	Reserved for future use
27	Not present	Reserved for Trajectory Change Message
28	0	Reserved for future use
	1	Aircraft Status Message (Emergency Priority Status)
	2	Aircraft Status Message (TCAS RA Message)
	3-7	Reserved for future use
29	0-1	Target State and Status
	2-3	Reserved for future use
30	0-7	Reserved for future use
31	0-1	Aircraft Operational Status
	2-7	Reserved for future use

Table 2: Sixbit ASCII Code

000000 = @	000001 = A	000010 = B	000011 = C
000100 = D	000101 = E	000110 = F	000111 = G
001000 = H	001001 = I	001010 = J	001011 = K
001100 = L	001101 = M	001110 = N	001111 = O
010000 = P	010001 = Q	010010 = R	010011 = S
010100 = T	010101 = U	010110 = V	010111 = W
011000 = X	011001 = Y	011010 = Z	011011 = [
011100 = \	011101 =]	011110 = ˆ	011111 = _
100000 =	100001 = !	100010 = ¨	100011 = #
100100 = $	100101 = %	100110 = &	100111 = ’
101000 = (	101001 =)	101010 = *	101011 = +
101100 = ,	101101 = -	101110 = .	101111 = /
110000 = 0	110001 = 1	110010 = 2	110011 = 3
110100 = 4	110101 = 5	110110 = 6	110111 = 7
111000 = 8	111001 = 9	111010 = :	111011 = ;
111100 = <	111101 = =	101110 = >	111111 = ?

Airborne Position Message, Type 9..18, 20..22

- Type code, 5 bits, bit 0..4: indicating the type of the message.
- Surveillance status, 2 bits, bit 5..6: indicating the setup of the transmitter in an aircraft.
- Nic supplement-B, 1 bit, bit 7: is used in combination with the type code and indicates integrity of an aircraft’s navigation equipment.
- Altitude, 12 bits, bit 8..19: encodes the altitude of an aircraft, following the setting of the so-called ”Q”-bit, located at bit position 15. If the ”Q”-bit is set to zero, the aircraft reports altitude in 100 foot increments, if the ”Q”-bit is set to one, the aircraft reports altitude in 25 foot increments. In order to retrieve altitude, bits 8..14 and 16..19 are combined and treated as one integer representing foot increments. The range of the combined integer goes from -1000 up to +50.175 feet.
- Time flag, 1 bit, bit 20: indicates whether the aircraft uses exact UTC timing (value set to one).
- CPR format, 1 bit, bit 21: indicated the CPR format even = 0, or odd = 1. This is significant in order to decode the cpr-encoded lat-lon position correctly.
- CPR encoded latitude, 17 bits, bit 22..38, and CPR encoded longitude, 17 bits, bit 39..55: the decoding of the aircrafts global latitude and

longitude position requires two consecutive airborne postion messages, one even, and one odd. The cpr-encoding realizes data compression. Its decoding is detailed in Sect. 5.1.

Airborne Velocity Message, Type 19

The ADS-B standard [3] discriminates between two kinds of airborne velocity message subtypes, i.e. subtype 1/2, and subtype 3/4. They use the same bit structure, but their contents are interpreted differently.

- Type code, 5 bits, bit 0..4: indicating the type of the message.
- Subtype, 3 bits, bit 5..7: indicating the subtype of the airborne velocity message, either 1/2, or 3/4.
- Intent change flag, 1 bit, bit 8: indicates change in intent.
- Reserved-A, 1 bit, bit 9: 0 indicates that the aircraft onboard systems comply with these modes of operation (should be zero all the time).
- Navigation accuracy category (NAC), 3 bits, bit 10..12: indicates the horizontal velocity error in the transmitted data; decimal 0 represents an error $\geq$ 10m/s, 1: $e < 10$ m/s, 2: $e < 3$ m/s, 3: $e < 1$ m/s, 4: $e < 0.3$ m/s.
- Subtype 1/2
  - East-west direction bit, 1 bit, bit 13: indicates the movement direction of the velocity vector, i.e. 0 = moving east, 1 = moving west.
  - East-west velocity, 10 bits, bit 14..23: indicates the sub-sonic velocity in knots in east-west direction (subtype 1). Decimal 0 indicates velocity information is not available, 1 indicates velocity = 0 knots, 2 = 1 knot, 3 = 2 knots, ... , 1022 = 1021 knots, 1023 indicates a velocity $\geq$ 1021.5 knots.
  - North-south direction bit, 1 bit, bit 24: indicated the movement direction of the velocity vector, i.e. 0 = moving north, 1 = moving south.
  - North-south velocity, 10 bits, bit 25..34: indicates the sub-sonic velocity in knots in north-south direction (subtype 1). Decimal 0 indicates velocity information is not available, 1 indicates velocity = 0 knots, 2 = 1 knot, 3 = 2 knots, ... , 1022 = 1021 knots, 1023 indicates a velocity $\geq$ 1021.5 knots.
- Subtype 3/4

  - Heading status bit, 1 bit, bit 13: 0 = heading data NOT available, 1 = heading data available.
  - Heading, 10 bits, bit 14..23: decimal 0 = heading is 0 degrees, dec 1 = heading is 0.3515625 degrees, 2 = 0.703125, ..., 1022 = 359.296875, 1023 = 359.6484375 degrees, thus a bit change accounts for 360/1024 degree change.
  - Airspeed type, 1 bit, bit 24: 0 = airspeed is indicated airspeed (IAS), 1 = airspeed is true airspeed (TAS).
  - Airspeed, 10 bits, bit 25..34: indicates subsonic airspeed, 0 = no AS available, 1 = AS is zero, 2 = AS is 1 knot, 3 = 2 knots, ... , 1022 = 1021 knots, 1023 > 1021.5 knots.

- Vertical rate source, 1 bit, bit 35: indicates the source for vertical rate information, i.e. 0 = geometric source, 1 = barometric source.

- Vertical rate sign, 1 bit, bit 36: indicates vertical movement, i.e. 0 = going up, 1 = going down.

- Vertical rate, 9 bits, bit 37..45: indicates the speed of the vertical movement. Decimal 0 indicates vertical rate informtion is not available, 1 indicates vertical rate = 0, 2 = 64 ft/min, 3 = 128 ft/min, ..., 510 = 32.575 ft/min, 511 > 32608 ft/min.

- Reserved, 2 bits, bit 46..47: 0 indicates that the aircraft onboard systems comply with these modes of operation (should be zero all the time).

- Difference from baro-altitude sign, 1 bit, bit 48: 0 indicates geometric (GNSS or INS) altitude source data is greater than (above) barometric; 1 indicates geometric (GNSS or INS) altitude source data is less than (below) barometric.

- Difference from baro-altitude, 7 bits, bit 49..55: reports the difference between Geometric (GNSS or INS) Altitude Source data and Barometric Altitude when both types of Altitude Data are available and valid. Decimal 0 indicates difference information is not available, 1 = difference is zero, 2 diff = 25 ft, 3 diff = 50 ft, ..., 126 diff = 3125 ft, 127 diff > 3137.5 ft.

The subtypes of airborne velocity messages are specified as follows:

- Decimal 0: reserved.

- Decimal 1: velocity over ground under normal speed conditions (non-supersonic speed).

- Decimal 2: velocity over ground under supersonic-speed conditions.
- Decimal 3: airspeed and heading Information when velocity over ground information is not available and airspeed conditions are normal, i.e., non-supersonic.
- Decimal 4: airspeed and heading information when velocity over ground information is not available and airspeed conditions are supersonic.
- Decimal 5..7: reserved.

5 Interpretation of Airborne Messages

5.1 Global Airborne Position Calculation

The decimal values for latitude and longitude found in position messages are encoded according to compact position reporting (CPR). CPR has been developed for ADS-B in order to reduce the number of bits required for expressing latitude and longitude [5]. Consequently, CPR represents a data compression technique.

The CPR coordinate system defines two sets of differently sized latitude and longitude zones, i.e. a set of even zones, and a set of odd zones. Each zone is identified by a zone number; there are 60 (0..59) even zones and 59 (0..58) odd zones. Latitude zones start with zone count 0 expand north and wrap around the globe. Longitude zones start with zone count 0, just east of the prime meridian. The number of longitude zones around the globe decreases with incrasing latitude value, i.e. fewer longitude zones towards the poles.

The height of a latitude zone is

$$Dlat_i = \frac{360^\circ}{4 * NZ - i} \tag{1}$$

with $NZ = 15$ (60/4) number of zones per quadrand, i.e. north-east, south-east, north-west, south-west, and $i = 0$ for an even latitude zone, and $i = 1$ for an odd latitude zone, i.e. even zones are about 360 nmi high, and odd zones about 366 nmi, measured north-south [5]. Each even zone spans 6° in height, each odd zone 6.1°. Further, each latitude zone is divided into so-called *bins*, i.e. $2^{Nb} = 2^{17} = 131072$ bins, with Nb being the number of bits used to encode CPR latitude or longitude for airborne positions. Each bin is identified by its bin number YZ_i, with $i = 0$ for an even zone bin, and $i = 1$ for an odd zone bin. Based on YZ_i, CPR produces a latitude value at the centerline of the bin, i.e. the bin centerline latitude. The bin height, which corresponds to the encoding resolution (2^{17}), is the latitude zone height divided by the number of bins per zone. Hence, the CPR encoding transforms latitude into even and odd bin numbers.

5.1.1 Global Latitude Decoding

The first step in latitude decoding is the calculation of the so-called zone index j. Equation 2 defines j for even (Lat_0) and odd (Lat_1) CPR-encoded latitude values coming from two subsequent airborne position messages.

$$j = floor\left(\frac{59 * Lat_0 - 60 * Lat_1}{2^{Nb}} + \frac{1}{2}\right) \tag{2}$$

The recovered latitude values for even ($Rlat_{i=0}$) and odd ($Rlat_{i=1}$) messages can then be calculated as defined in Equation 3 by applying the `MOD` function defined in Equation 4.

$$Rlat_i = Dlat_i * \left(MOD(j, 60 - i) + \frac{Lat_i}{2^{Nb}}\right) \tag{3}$$

$$MOD(x, y) \to x - y * floor\left(\frac{x}{y}\right) \tag{4}$$

One of the two recovered latitude values, $Rlat_0$ or $Rlat_1$, is chosen as true current latitude in degrees, i.e. the one with the same CPR format as the message that was received second (last).

5.1.2 Global Longitude Decoding

Longitude decoding is similar to latitude decoding, in that an even and an odd CPR message is used for calculating the zone index, m in this case. The main difference lies in the fact that the number of longitude zones varies with the latitude [5]. There are fewer longitude zones towards the poles, i.e. between 1 and 59 zones between the equator and either of the poles. The first step in longitude decoding is to ascertain that both recovered latitude positions used in the calculation are associated with the same number of longitude zones (NL). NL can be determined either through a formula [5], or through a lookup table [3]. The lookup table is presented at the end of this document.

If both $NL(Rlat_0) and NL(Rlat_1)$ are the same, the longitude zone index m can be calculated from the two CPR encoded longitude values, Lon_0 and Lon_1, as defined in Equation 5. If the number of longitude zones of both messages differ, they cannot be used for global position decoding. In this case, decoding must be postponed until two messages with the same NL become available.

$$m = floor\left(\frac{(NL - 1) * Lon_0 - NL * Lon_1}{2^{Nb}} + \frac{1}{2}\right) \tag{5}$$

The recovered longitude values for even ($Rlon_{i=0}$) and odd ($Rlon_{i=1}$) messages can then be calculated as defined in Equation 6 by using the longitude zone size $Dlon_i$ defined in Equation 7.

$$Rlon_i = Dlon_i * \left(MOD(m, NL - i) + \frac{Lon_i}{2^{Nb}} \right) \tag{6}$$

$$Dlon_i = \frac{360}{max(NL - i, 1)} \tag{7}$$

One of the two recovered longitude values, $Rlat_0$ or $Rlat_1$, is chosen as true current longitude in degrees, i.e. the one with the same CPR format as the message that was received second (last).

5.2 Local Airborne Position Calculation

Local position decoding uses a *reference-position* with Lat_s and Lon_s, and the contents of one position message, with Lat_i and Lon_i, an even ($i = 0$) or an odd one ($i = 1$), in order to recover latitude and longitude of an airborne aircraft [5]. Prerequisite is that the difference in both latitude and longitude of the reference position and the target position is less than $\frac{1}{2}$ a zone in both directions. The reference position is usually the last successfully decoded position, e.g. an earlier global position, or an earlier local position of the target aircraft. Since $\frac{1}{2}$ of a zone is approximately 180 nmi, it is unlikely that the position change exceeds this limit, i.e. given that the aircraft abides by the reporting intervals specified.

Similar to the global decoding, latitude must be determined first, and then with the knowledge of the NL at this latitude position, longitude can be determined. Here, the zone index j denotes the difference in zone fractions between the reference position latitude and the target position latitude, and it is defined for the CPR-encoded latitude value Lat_i and the latitude of the reference position Lat_s in Equation 8. The recovered latitude value $Rlat_i$ for a particular CPR type message i is, thus, defined in Equation 9.

$$j = floor\left(\frac{Lat_s}{Dlat_i}\right) + floor\left(\frac{MOD(Lat_s, Dlat_i)}{Dlat_i} - \frac{Lat_i}{2^{Nb}} + \frac{1}{2}\right) \tag{8}$$

$$Rlat_i = Dlat_i * (j + \frac{Lat_i}{2^{Nb}}) \tag{9}$$

The decoding of the longitude is performed accordingly. The zone index m denotes the difference in zone fractions between the reference position longitude and the target postion longitude, and is defined for the CPR-encoded longitude of the target position Lon_i, and the longitude of the reference position Lon_s in Equation 10. The recovered longitude value $Rlon_i$ for a particular CPR type message i is, thus, defined in Equation 11.

$$m = floor(\frac{Lon_s}{Dlon_i}) + floor(\frac{MOD(Lon_s, Dlon_i)}{Dlon_i} - \frac{Lon_i}{131072} + \frac{1}{2}) \tag{10}$$

$$Rlon_i = Dlon_i * \left(m + \frac{Lon_i}{2^{Nb}} \right) \tag{11}$$

Finally, the width of longitude zone $Dlon_i$ is defined in the following equation (Eq. 12).

$$Dlon_i = \begin{cases} \frac{360^\circ}{NL(Rlat_i)-i} & \text{if } NL(Rlat_i) - i > 0 \\ 360^\circ & \text{if } NL(Rlat_i) - i = 0 \end{cases} \tag{12}$$

5.3 Heading and Velocity for Subtype 1 and 2 Velocity Messages

In contrast to subtype 3 and 4 of airborne velocity messages that encode the heading and velocity directly in degrees and knots, subtype 1 and 2 airborne velocity messages are more complicated to decipher. Here, the heading as well as the velocity must be calculated from the combination of horizontal (east-west) and vertical (north-south) velocities, plus the directions east, west, north or south. Figure 1 illustrates, how, through the application of the appropriate trigonometric functions, both, the direction angle β, and the length of vector c can be determined from offset a and offset b. This translates into heading and velocity.

It is important to note that, depending on the quadrant determined through the movement direction bits, the resulting angle values must be adjusted accordingly, i.e. $90^\circ - \beta$ in the north-east quadrant, $90^\circ + \beta$ in the south-east quadrant, $270^\circ - \beta$ in the south-west quadrant, and $270^\circ + \beta$ in the north-west quadrant.

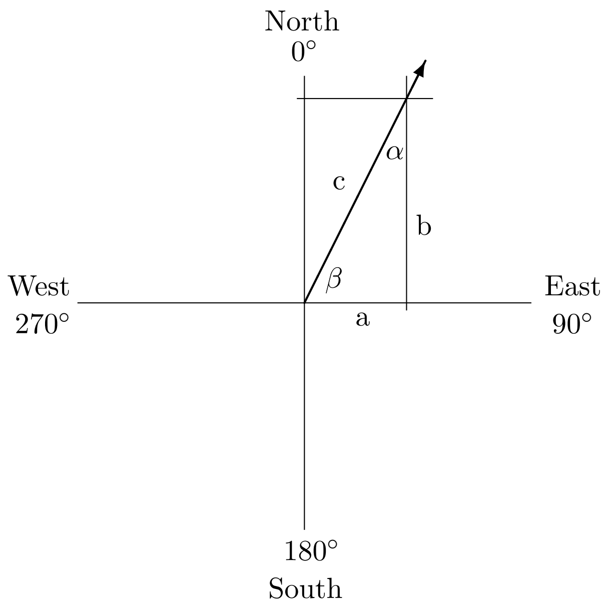


Figure 1: Application of trigonometry for calculating heading (β) and velocity (c)

References

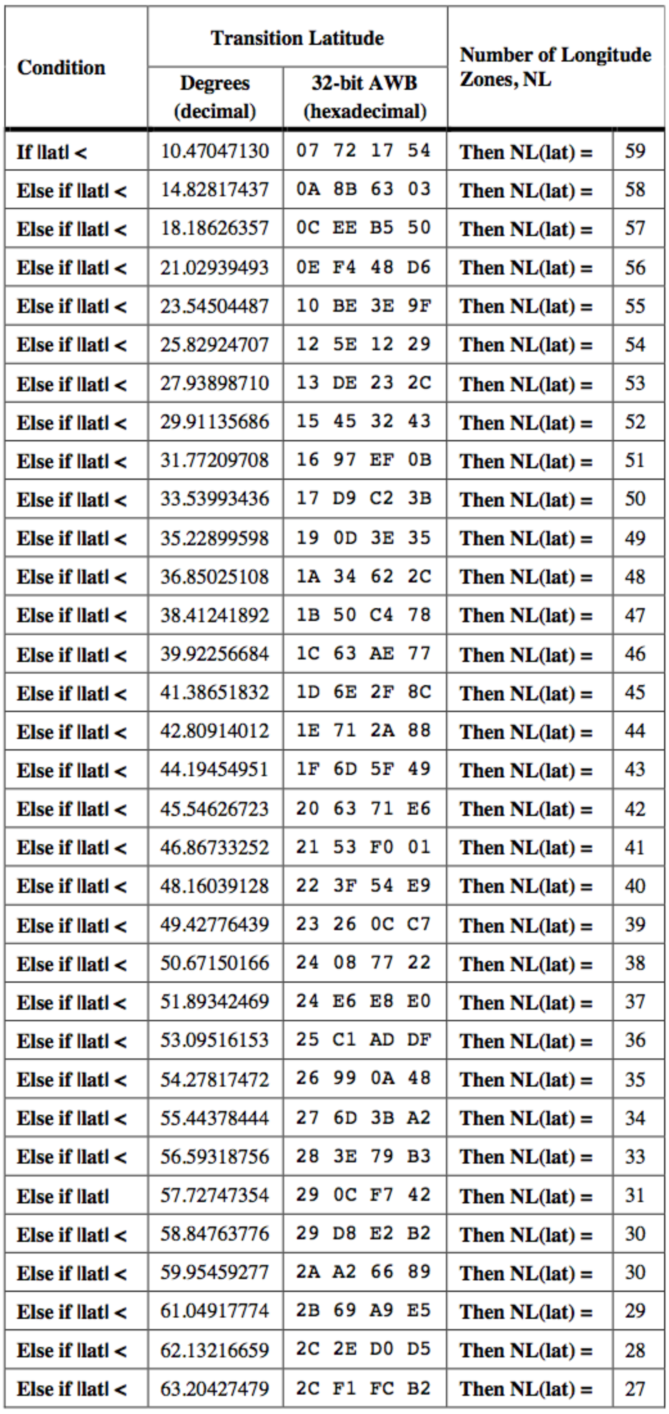
[1] Wikipedia. *Automatic Dependent Surveillance Broadcast.* https://en.wikipedia.org/wiki/Automatic_dependent_surveillance-broadcast.

[2] Global Navigation Systems (GNS) GmbH. GNS 5890 ADS-B Receiver. http://www.gns-gmbh.com.

[3] RTCA. *Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance Broadcast (ADS-B) andTraffic Information Services Broadcast (TIS-B).* RTCA DO-260B.

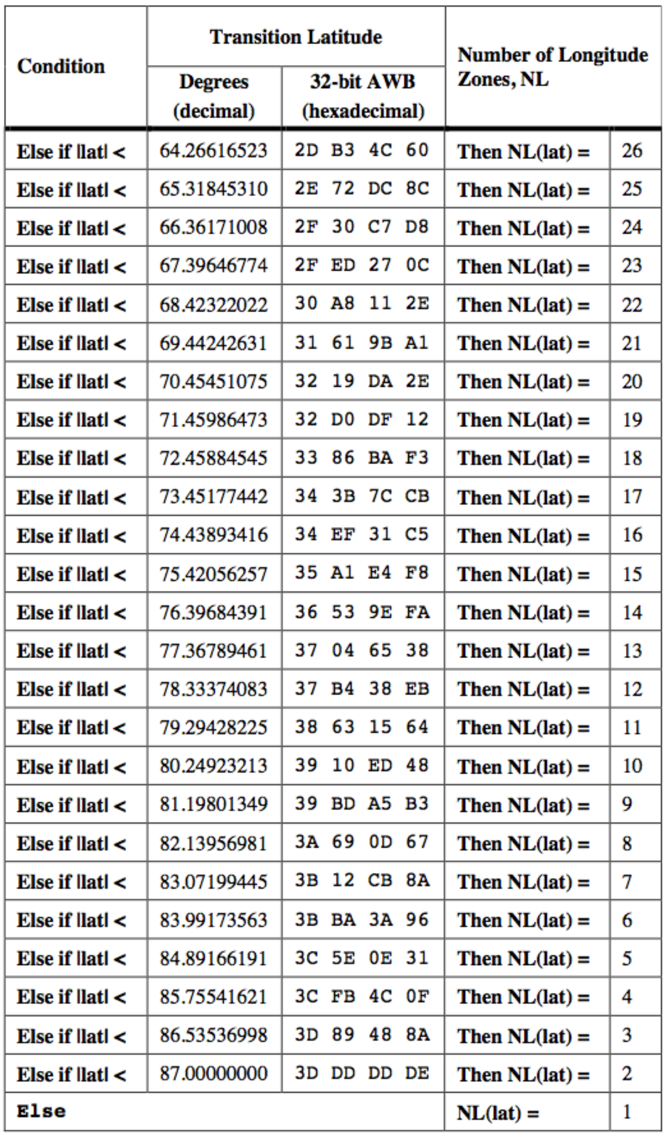
[4] RTCA. *Transition Table for NL(lat) Function.* ADS-B 1090 MHz MOPS, Meeting #9, 1090-WP-9-14, Fort Lauderdale, January 9th, 2002.

[5] Marshall, A. *An Expanded Description of the CPR Algorithm.* RTCA Special Committee 186, Working Group 3, July 24, 2009.



Condition	Transition Latitude		Number of Longitude Zones, NL	
	Degrees (decimal)	32-bit AWB (hexadecimal)		
If |lat| <	10.47047130	07 72 17 54	Then NL(lat) =	59
Else if |lat| <	14.82817437	0A 8B 63 03	Then NL(lat) =	58
Else if |lat| <	18.18626357	0C EE B5 50	Then NL(lat) =	57
Else if |lat| <	21.02939493	0E F4 48 D6	Then NL(lat) =	56
Else if |lat| <	23.54504487	10 BE 3E 9F	Then NL(lat) =	55
Else if |lat| <	25.82924707	12 5E 12 29	Then NL(lat) =	54
Else if |lat| <	27.93898710	13 DE 23 2C	Then NL(lat) =	53
Else if |lat| <	29.91135686	15 45 32 43	Then NL(lat) =	52
Else if |lat| <	31.77209708	16 97 EF 0B	Then NL(lat) =	51
Else if |lat| <	33.53993436	17 D9 C2 3B	Then NL(lat) =	50
Else if |lat| <	35.22899598	19 0D 3E 35	Then NL(lat) =	49
Else if |lat| <	36.85025108	1A 34 62 2C	Then NL(lat) =	48
Else if |lat| <	38.41241892	1B 50 C4 78	Then NL(lat) =	47
Else if |lat| <	39.92256684	1C 63 AE 77	Then NL(lat) =	46
Else if |lat| <	41.38651832	1D 6E 2F 8C	Then NL(lat) =	45
Else if |lat| <	42.80914012	1E 71 2A 88	Then NL(lat) =	44
Else if |lat| <	44.19454951	1F 6D 5F 49	Then NL(lat) =	43
Else if |lat| <	45.54626723	20 63 71 E6	Then NL(lat) =	42
Else if |lat| <	46.86733252	21 53 F0 01	Then NL(lat) =	41
Else if |lat| <	48.16039128	22 3F 54 E9	Then NL(lat) =	40
Else if |lat| <	49.42776439	23 26 0C C7	Then NL(lat) =	39
Else if |lat| <	50.67150166	24 08 77 22	Then NL(lat) =	38
Else if |lat| <	51.89342469	24 E6 E8 E0	Then NL(lat) =	37
Else if |lat| <	53.09516153	25 C1 AD DF	Then NL(lat) =	36
Else if |lat| <	54.27817472	26 99 0A 48	Then NL(lat) =	35
Else if |lat| <	55.44378444	27 6D 3B A2	Then NL(lat) =	34
Else if |lat| <	56.59318756	28 3E 79 B3	Then NL(lat) =	33
Else if |lat|	57.72747354	29 0C F7 42	Then NL(lat) =	31
Else if |lat| <	58.84763776	29 D8 E2 B2	Then NL(lat) =	30
Else if |lat| <	59.95459277	2A A2 66 89	Then NL(lat) =	30
Else if |lat| <	61.04917774	2B 69 A9 E5	Then NL(lat) =	29
Else if |lat| <	62.13216659	2C 2E D0 D5	Then NL(lat) =	28
Else if |lat| <	63.20427479	2C F1 FC B2	Then NL(lat) =	27

Figure 2: NL lookup table part 1 [3, 4].



Condition	Transition Latitude		Number of Longitude Zones, NL	
	Degrees (decimal)	32-bit AWB (hexadecimal)		
Else if |lat| <	64.26616523	2D B3 4C 60	Then NL(lat) =	26
Else if |lat| <	65.31845310	2E 72 DC 8C	Then NL(lat) =	25
Else if |lat| <	66.36171008	2F 30 C7 D8	Then NL(lat) =	24
Else if |lat| <	67.39646774	2F ED 27 0C	Then NL(lat) =	23
Else if |lat| <	68.42322022	30 A8 11 2E	Then NL(lat) =	22
Else if |lat| <	69.44242631	31 61 9B A1	Then NL(lat) =	21
Else if |lat| <	70.45451075	32 19 DA 2E	Then NL(lat) =	20
Else if |lat| <	71.45986473	32 D0 DF 12	Then NL(lat) =	19
Else if |lat| <	72.45884545	33 86 BA F3	Then NL(lat) =	18
Else if |lat| <	73.45177442	34 3B 7C CB	Then NL(lat) =	17
Else if |lat| <	74.43893416	34 EF 31 C5	Then NL(lat) =	16
Else if |lat| <	75.42056257	35 A1 E4 F8	Then NL(lat) =	15
Else if |lat| <	76.39684391	36 53 9E FA	Then NL(lat) =	14
Else if |lat| <	77.36789461	37 04 65 38	Then NL(lat) =	13
Else if |lat| <	78.33374083	37 B4 38 EB	Then NL(lat) =	12
Else if |lat| <	79.29428225	38 63 15 64	Then NL(lat) =	11
Else if |lat| <	80.24923213	39 10 ED 48	Then NL(lat) =	10
Else if |lat| <	81.19801349	39 BD A5 B3	Then NL(lat) =	9
Else if |lat| <	82.13956981	3A 69 0D 67	Then NL(lat) =	8
Else if |lat| <	83.07199445	3B 12 CB 8A	Then NL(lat) =	7
Else if |lat| <	83.99173563	3B BA 3A 96	Then NL(lat) =	6
Else if |lat| <	84.89166191	3C 5E 0E 31	Then NL(lat) =	5
Else if |lat| <	85.75541621	3C FB 4C 0F	Then NL(lat) =	4
Else if |lat| <	86.53536998	3D 89 48 8A	Then NL(lat) =	3
Else if |lat| <	87.00000000	3D DD DD DE	Then NL(lat) =	2
Else			NL(lat) =	1

Figure 3: NL lookup table part 2 [3, 4].