The 1090 Megahertz Riddle

A Guide to Decoding Mode S and ADS-B Signals

Junzi SUN

Faculty of Aerospace Engineering
Delft University of Technology

Title: The 1090 Megahertz Riddle

Subtitle: A Guide to Decoding Mode S and ADS-B Signals

Author: Junzi Sun

Edition: 2

Keywords: Aircraft Surveillance, Radar, Decoding, Mode A/C, Mode S, ADS-B, Comm-B, ELS, EHS, MRAR.

Publisher: TU Delft OPEN Publishing

Cover design: Junzi Sun

ISBN: 978-94-6366-402-8

An electronic version of this dissertation is available at
https://doi.org/10.34641/mg.11

This book is dedicated to my sons: William and Vincent

Contents

Preface

This book provides researchers, engineers, and students a practical guide to decoding ADS-B and other types of common Mode S messages. It consolidates the information from various ICAO documentation and other literature to provide readers easy access to key knowledge of Mode S and ADS-B and related topics. The book extensively uses examples and sample Python code to explain the decoding process.

Back in 2015, I joined TU Delft to undertake PhD research on analyzing and modeling aircraft performance using open aircraft surveillance data. ADS-B data served as my primary data source. Frustrated with the lack of open literature on ADS-B and Mode S, I created a live online project to document my experience in decoding ADS-B data, entitled the *ADS-B Decoding Guide*. As the guide grew in popularity, I started receiving questions, feedback, and suggestions from the research community all over the world. This input greatly helped me to fix and improve the content of the decoding guide.

By then, I also started to incorporate Mode S Enhanced Surveillance data into my research, which required me to further develop tools for inferring and decoding new types of messages. At the same time, I created more content for the online book. Due to the increasing interest expressed and demand from readers, I started writing a more comprehensive book focused on the decoding practice of the data, which was the starting point for this book, *the 1090 Megahertz Riddle*.

During these years of work with ADS-B and Mode S data, I created a Python decoding library, *pyModeS*, which welcomed contributions from GitHub users from all over the world. In this book, some decoding examples are shown using *pyModeS*. However, knowledge of Python is not required to understand the topics covered in this book.

In 2019, I completed my PhD and continued as a faculty member in the aerospace engineering faculty of TU Delft. By then, I had time to reflect on aircraft surveillance data from a new perspective and decided to make a major update of the book's content.

The result is this text, which is both the second edition of *the 1090 Megahertz Riddle* and one of the first books from TU Delft's OPEN publishing initiative. It is published as an open access book, under the CC-BY-NC-SA 4.0 license. The LaTeX source of the book is also shared on GitHub, where future comments and pull requests are greatly appreciated.

Junzi Sun
Delft, the Netherlands
April, 2021

Part I

Getting Started

1 Introduction

1.1 Background: The “death ray” that saves lives

Like many modern technologies such as computers, the internet, and GPS whose origins can be traced back to the military, aviation radar is no exception.

The fundamental theory of radar started in late 19th century. Since the 1860s, when the electromagnetic theory was discovered by James Clerk Maxwell, the foundation for many science and technology fields was laid out. In the late 19th century, Heinrich Hertz, who proved the existence of electromagnetic waves, also confirmed that metals could reflect radio waves. In the first decades of the 20th century, several systems for using radio waves to provide short-range directional information of objects were developed. German inventor Christian Hülsmeyer is often considered as the first person to use radio waves to detect metal objects in 1904.

However, not until the Second World War, was the concept of *RAdio Detection And Ranging* (RADAR) developed. The technology was simultaneously researched by both major Allies and Axis countries. However, the United Kingdom led the race in developing a functional radar system.

Originally, British Air Ministry officials, who were concerned about falling behind the technology race with the Germans, advised physicist Robert Watson Watt to propose a set of abstract technical challenges to his colleague Skip Wilkins:

> "Suppose, just suppose, that you had eight pints [17kg] of water, 3,000ft [1km] above the ground. And suppose that water was at 98°F [37°C], and you wanted to heat it to 105°F [41°C]. How much radio frequency power would you require, from a distance of 5km?"

Both scientists understood very well that the average adult human has 17 kg blood and a critical body temperature of 41°C. This represented an idea of the *death ray* that could be used to disable enemy aircraft pilots.

Without even developing such a system, they figured out that the power requirement was far beyond practice at that time. However, they realized an opportunity for funding a different project. Given the amount of power that can be transmitted, it was possible to detect the reflection of aircraft, and, thus calculate the position of aircraft. This proposal was approved by the air ministry. Later on, this technology was developed and shared with Americans during the war. After that, the widespread usage of RADAR (which later became radar) had begun.

After the war ended, radar became more and more important for civil aviation. With the rapid growth of commercial flights, it developed into a prominent technology for aircraft surveillance in the aviation industry.

1.2 The primary radar

Aircraft surveillance in the early days relied only on, what is known today as, the primary surveillance radar (PSR). The concept of PSR is fairly simple. It is a rotating radio transponder with an omnidirectional antenna. Commonly, the radar transmits a one-microsecond pulse for every one millisecond and listens to the reflections from the airplanes. The position of the aircraft is measured by the distance and angle to the radar. The distance is known as the *slant distance*, which is the line-of-sight distance between an aircraft and the radar. It can be calculated by measuring the time difference between the original signal and the reflection received, since the speed of the radio wave (speed of light) is known. The azimuth angle[1] of the aircraft is determined by the rotation angle of the radar. By processing radar signals, technologies like phase filter and Doppler filter can be used to filter out moving targets like aircraft and remove static objects, such as mountains, buildings, and other obstacles.

The slant distance of an aircraft does not always correspond to the horizontal distance to the radar. This is because all points having the same slant distance are located on a sphere that has the radius equal to the distance. Since the civil radar usually does not provide elevation information on the target, it is not possible to accurately convert the slant distance to the horizontal distance. Historically, it is sufficient to use primary radar for separating airplanes without considering these altitude differences. However, other systems have to be in place to provide air traffic controllers more accurate positions of the aircraft.

1.3 The secondary radar

The secondary surveillance radar (SSR), also known as the air traffic control radar beacon system (ATCRBS), was designed to provide air traffic controllers more information than what is provided by the primary radar. The secondary radar can be installed separately or installed on top of the primary radar. It uses a different radio frequency to actively interrogate the aircraft and receive information transmitted by the aircraft.

The SSR transmits interrogations using the 1030 MHz radio frequency and the aircraft transponder transmits replies using the 1090 MHz radio frequency. In the early design of SSR, two civilian communication protocols (Mode A and Mode C) were introduced. Mode A and Mode C respectively allow the SSR to continuously interrogate the identity (squawk code) and the altitude of an aircraft. The squawk code in Mode A is a unique 4-octal digit code given by air traffic controllers to aircraft in their flight information region for identification. The altitude in Mode C refers to the barometric altitude obtained from the aircraft's air data system.

Although Mode A and Mode C provide additional information to air traffic con-

[1] Azimuth angle is the horizontal angle measured clockwise to the north of the observer.

trollers, they present several design challenges [12]. The secondary surveillance radar initiates Mode A and C interrogations with two different pulse patterns, which are shown in Figure 1.1.

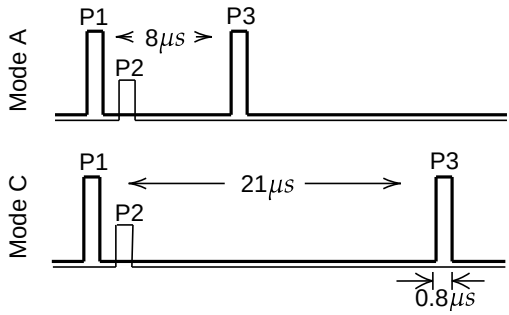


Figure 1.1: Mode A/C interrogation pulses

The pulses are about 0.8 μs wide. P1 and P3 are the two main pulses sent by the directional antenna. They are separated by 8 μs and 21 μs, respectively for Mode A and C. P2 is a pulse emitted through the omnidirectional antenna right after P1. Pulse P2 is introduced for sidelobe suppression [20]. When the power of P2 is higher than P1, the interrogation is likely from the side lobes of the directional antenna and should be ignored by the aircraft. This is can happen when the aircraft is close to the radar.

Figure 1.2 shows an example of a Mode A/C reply. Each reply consists of two persistent pulses, F1 and F2, separated by 20.3 μs. Within this period, either the identity code or the altitude code is encoded using 13 pulses of 0.45 μs. The pulses are separated by gaps of 1 μs. The pulse at the center serves as a verification pulse and is always absent. The presence or absence of any of the other 12 pulses represents a 1 or 0 bit. When required by air traffic controllers for identification purposes, a special purpose identification (SPI) may follow F2 after two absent pulses.

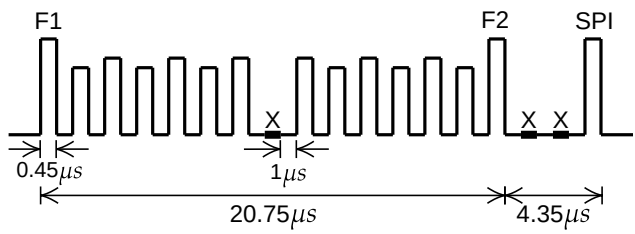


Figure 1.2: Mode A/C reply pulses

The response type (A or C) cannot be identified based on the reply signal itself. The SSR determines the content based on the synchronization with interrogations. This

design of ATCRBS works sufficiently well with low-density air traffic, but it cannot efficiently cope with higher flight densities since all aircraft replies are transmitted on the same frequency. When several aircraft are in the same direction of the radar beam, reply signals can overlap and introduce errors for decoding. This is known as synchronous garbling.

When there are multiple secondary radars in the vicinity, replies originated by other radars may be considered as valid responses of one radar, which in turn, causes errors and confusion. This syndrome is called *FRUIT* (False Replies Unsynchronized In Time).

Though some of the garbling and the FRUIT problems can be mitigated with the reduction of interrogation frequency and improvements in signal processing ability, the information transmitted in Mode A and Mode C is still very limited. The number of identity codes available in Mode A communication is limited to a maximum of 4096 (8^4 given the 4 octal digits) unique codes, which poses another clear limitation.

Hence, more advanced communication protocols needed to be developed to acquire more information from a large number of aircraft.

1.4 Mode S

Mode S (Mode Select Beacon System) was designed by Lincoln Laboratory at Massachusetts Institute of Technology in the 1970s. Based on different iterations of hardware and software design in the 1980s, the implementation of Mode S in air traffic control began in the 1990s. Since then, Mode S has become one of the main sources for aircraft surveillance.

The main characteristic of Mode S is its selective interrogation, which allows the SSR to interrogate different information from different aircraft separately. By using selective interrogation, it largely mitigated the problem of garbling in Mode A/C and thus greatly improved the capacity of the communication channel.

Unlike the limited number (4096) of unique identification codes in Mode A communication, the Mode S transponder is identified by a 24-bit transponder code, which can support up to 16,777,216 (2^{24}) unique addresses. In addition to these two major advantages, the Mode S protocol introduced many different types of information that could be interrogated and downlinked.

1.4.1 Mode S interrogations

The Mode S uplink signal contains parameters that indicate which information is desired by the air traffic controller. There are two types of Mode S interrogations, which are shown in Figure 1.3. The short interrogation has 56 bits of information contained in the data block, while the long interrogation contains 112 bits of information. The P2 pulse acts as the sidelobe suppression for Mode A/C transponders so that they will ignore the rest of the interrogation pulses. Information in the Mode S

interrogation data block uses the Differential Phase-Shift Keying (DPSK) modulation [17].

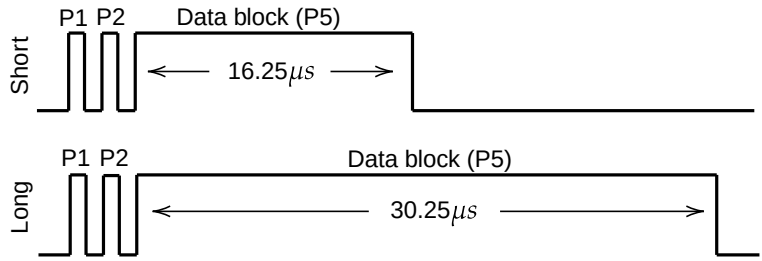


Figure 1.3: Mode S uplink pulses

1.4.2 Mode S replies

There are two types of Mode S downlink signals, the short reply and the long reply, which correspond to the short and long interrogations from the SSR. For each microsecond, one bit is transmitted. All Mode S replies start with an 8 μs fixed preamble and continue with 56- or 112 μs data block. The structure of the downlink message is shown in Figure 1.4.

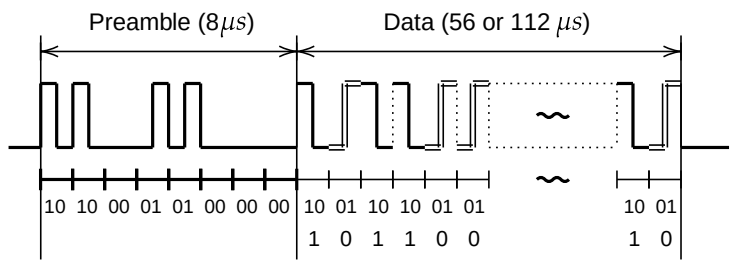


Figure 1.4: An example of Mode S reply message

The 16-bit fix preamble can be represented as `1010000101000000` in binary. The information contained in the data block is modulated using the Pulse Position Modulation (PPM), which is a type of amplitude modulation. In PPM, the `1` bit is represented by a 0.5 μs of pulse followed by a 0.5 μs flat signal. The `0` bit is reversed compared to the `1` bit, which is represented by a 0.5 μs flat signal and followed by a 0.5 μs pulse.

Note

Mode S uplink and downlink signals use different modulation methods. The uplink uses phase modulation, while the downlink uses amplitude modulation. Phase modulation requires slightly more complicated hardware implementation but has a better performance in terms of error tolerance.

1.4.3 Compatibility with Mode A/C

Similar to Mode A/C, Mode S SSR can send *All-Call* interrogations to all aircraft in the vicinity. The All-Call interrogation is designed to be compatible with Mode A/C transponders. Figure 1.5 shows the pulses of an All-Call interrogation. There are 8 or 21 μs between P1 and P3, which is the same as Mode A/C. P4 is a wider pulse lasting for 1.6 μs.

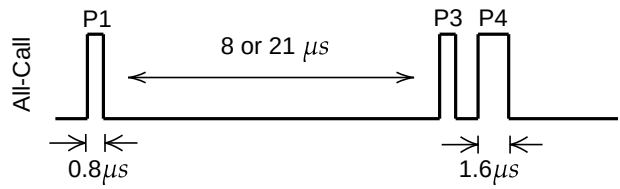


Figure 1.5: Mode A/C/S All-Call interrogation

When a Mode A/C transponder receives this interrogation, the last pulse P4 is ignored. Hence, a normal Mode A or Mode C reply is transmitted. A Mode S transponder would detect P4, and thus, produce a Mode S All-Call reply. The Mode A/C/S All-Call reply is designed for SSR to identify which aircraft are in the vicinity, including Mode A/C only aircraft.

This is one of the several different interrogation pulse patterns from a Mode A/C compatible Mode S radar. In addition, Mode S SSRs can perform both Mode A/C and Mode S interrogations.

Note

The Mode A/C/S All-Call is being phased out, as new transponders stop replying to this type of interrogations. It is common for Mode S SSR makes use of Mode S only All-Call interrogation with the standard Mode S uplink (format 11) as shown in Figure 1.3.

1.4.4 Mode S format

The Mode S communication protocol is designed to handle different types of uplink and downlink message formats. The first five bits of the message define the uplink format (UF) or downlink format (DF) number of the message. Based on the UF/DF

number, different structures of the data block are presented. Table 1.1 shows all available Mode S formats. Currently, 11 Mode S formats are being used. Numbers not in this table are reserved for future use.

Table 1.1: Mode S uplink and downlink formats

UF/DF	Bits	Uplink type	Downlink type
0	56	Short air-air surveillance (ACAS)	Short air-air surveillance (ACAS)
4	56	Surveillance, altitude request	Surveillance, altitude reply
5	56	Surveillance, identity request	Surveillance, identity reply
11	56	Mode S All-Call	All-Call reply
16	112	Long air-air surveillance (ACAS)	Long air-air surveillance (ACAS)
17	112	-	Extended squitter
18	112	-	Extended squitter/non transponder
19	112	-	Military extended squitter
20	112	Comm-A, altitude request	Comm-B, altitude reply
21	112	Comm-A, identity request	Comm-B, identity reply
24	112	Comm-C (ELM)	Comm-D (ELM)

Note

Format number 24 is an exception. It is identified using only the first two bits, which must be `11` in binary. All following bits are used for encoding other information.

We can see that the short 56-bit data block is used to encode messages with format numbers from 0 to 11. Messages with format numbers above 16 are encoded with the long 112-bit data block. Among all uplink formats, UF 17, 18, and 19 are not used. This is because the corresponding downlink messages (extended squitter messages) are designed to be broadcast automatically without the need for SSR interrogations.

One of the most common applications for the extended squitter is the Automatic Dependent Surveillance-Broadcast service, which is commonly known as ADS-B.

1.5 ADS-B

Automatic Dependent Surveillance-Broadcast (ADS-B) is a surveillance technology designed to allow aircraft to broadcast their flight state periodically without the need for interrogation. The word *automatic* refers to the fact that no inputs from controllers or pilots are required. The word *dependent* indicates this technology depends on information from other onboard systems, such as air data systems and navigation systems. ADS-B only become possible because of Global Navigation Satellite Systems (GNSS), such as Global Positioning System (GPS), which is the first GNSS system made available for civil usage by the USA after the Korean Air Lines Flight 007 accident in 1983.

Common aircraft state parameters included in ADS-B are position, altitude, and

speed. The position is determined by GNSS and air data systems. The velocity is derived from the GNSS position and the inertial measurement system. The altitude information includes both barometric altitude and GNSS altitude. The barometric altitude is provided by the air data system. In addition to these primary state parameters, ADS-B also allows other information to be broadcast, for example, aircraft call-sign, accuracy indicators, integrity indicators, and operational status.

It is worth emphasizing that ADS-B is a broadcast-based surveillance system. Unlike other types of Mode S surveillance, no surveillance interrogation is required. The decoding of ADS-B messages is also easier than other Mode S messages because the message type and structure are clear for each individual message.

Different types of ADS-B messages are identified by Type Codes. The structures of these messages are all defined in ICAO documents, such as in [13] and [21]. In Part I of this book, we will explain how these different types of ADS-B messages can be decoded.

Note

In this book, ADS-B only refers to the ADS-B standard implemented through Mode S extended squitter. Originally, there were three candidates to build ADS-B: Mode S Extended Squitter, VHF Data Link - Mode 4 (VDL4), and Universal Access Transceiver (UAT). Different specifications are designed for these frequency channels [21, 22]. However, the Mode S extended squitter implementation of ADS-B is the most adopted one among all candidates. UAT is only partially implemented in some countries like the USA, while VDL4 did not go beyond the testing phase.

1.6 Other Mode S services

The most common data format number used in Mode S for information downlink are 4, 5, 20, and 21. Format 4 and 5 are short messages designed to acquire aircraft altitude and identity (similar to Mode A/C messages). Downlink messages with format 20 or 21, as known as Comm-B messages, contain other information desired by air traffic controllers, in addition to altitude and identity.

In theory, up to 255 different message types are supported by the Mode S Comm-B. A new parameter, Comm-B Data Selector (BDS), is designed to identify which additional information is included in Mode S messages. It functions similarly to the Type Code of ADS-B message. In practice, not all 255 BDS codes are defined or used.

A subgroup of these codes is commonly interrogated by surveillance radars. By combining different BDS codes into groups, several Mode S services are defined. The two most used services are Mode S Elementary Surveillance (ELS) and Mode S Enhanced Surveillance (EHS) [8]. For example, in the European airspace, aircraft above a certain takeoff weight category are required to have these capabilities enabled.

The Mode S Elementary Surveillance consists of four BDS codes (10, 17, 20, and

30). It provides basic information such as Mode S capabilities and identification (callsign), in addition to the altitude or identity (squawk code) information.

The Mode S Enhanced Surveillance consists of three BDS codes (40, 50, and 60). It provides much more additional information on aircraft states, such as selected altitudes, true airspeed, indicated airspeed, Mach number, bank angle, and turn rate.

In addition to the common ELS and EHS, the meteorological routine air report (MRAR), and the meteorological hazard report (MHR) are also interrogated in some controlled airspaces. These reports provide weather-related information such as wind and temperature that are measured by the aircraft sensors. Another important service, Airborne Collision Avoidance System (ACAS), also uses the Mode S transponder for comminucation. All these services will be explained in details in this book.

1.7 Summary

Figure 1.6 illustrates all aforementioned Mode S services and their relationships with different Mode S downlink formats. This book covers all Mode S messages except DF19 and DF24 from this figure.

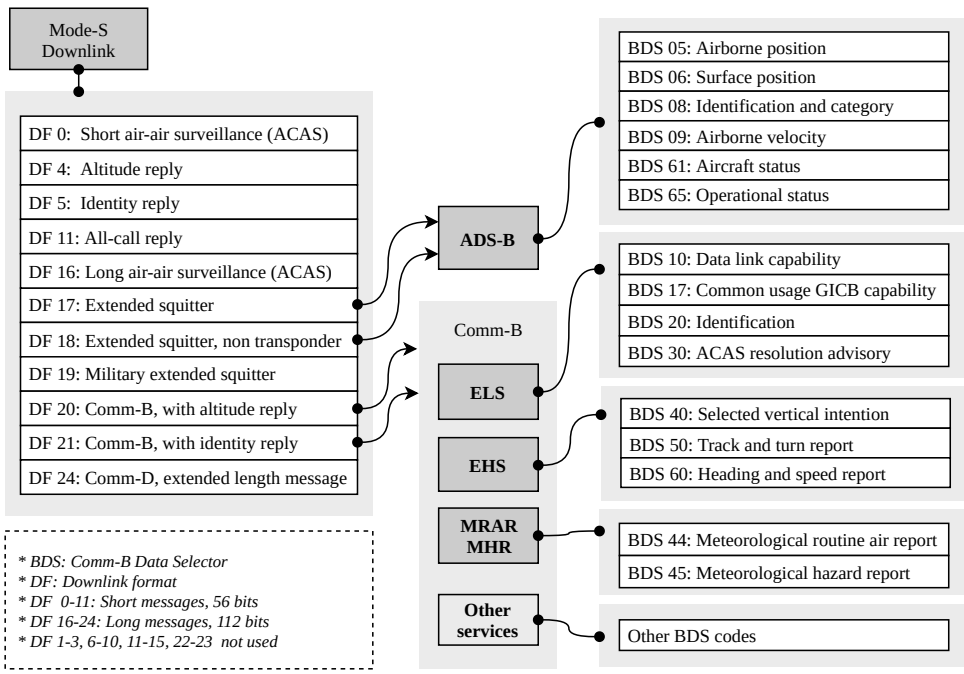


Figure 1.6: Relationship between Mode S downlink formats and different services

In the rest of this book, the details of these services are discussed in separate chap-

ters. In Chapter 2 (the rest of part I), common hardware and software setups are shown. In Chapters 3 to 10 (part II), the focus is on ADS-B messages. Detailed guides on how to decode different types of messages are presented. In Chapters 11 to 19 (part III), the focus is on Mode S ELS, EHS, and meteorological services. Accordingly, the decoding and inference of the messages are provided.

2 Quick Start: Hardware and Software to Receive Mode S Signals

Chapter 1 introduced several fundamental concepts about Mode S communication. This chapter focuses on the practical aspects of acquiring Mode S data, specifically, how to receive and obtain messages that are transmitted in Mode S downlink communications. Mode S signals are transmitted using 1090 MHz radio waves. Hence, the receiver and antenna have to be designed to work with this frequency. Nowadays, all necessary components required to obtain the Mode S data can be easily made with low-cost off-the-shelf components. Several open-source software tools are available to support the extraction and decoding of data from Mode S signals. In this chapter, we explain how to step up a functional hardware and software system to receive Mode S data.

2.1 Range

Since Mode S uses L band signals that follow the line-of-sight propagation, any obstructions between the transmitter and receiver can cause a significant amount of signals to be blocked. Assuming that no obstacle exists between the aircraft and the receiver, and the transmitter has a sufficient amount of power, the maximum range of the receiver is determined by the curvature of the earth. Figure 2.1 illustrates how the maximum range of the receiver can be obtained.

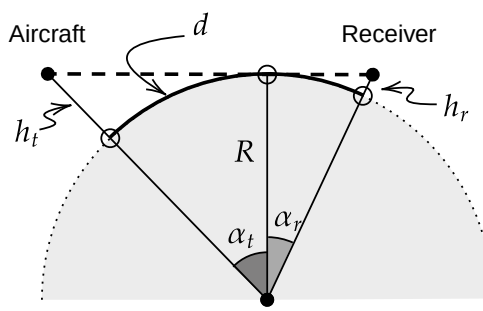


Figure 2.1: Maximum receiver range

Knowing the altitude of the receiver antenna, it is possible to determine the maximum range (d) of a Mode S receiver as:

$$d = (\alpha_r + \alpha_t)R \tag{2.1}$$

$$= \left(\arccos\frac{R}{R+h_r} + \arccos\frac{R}{R+h_t}\right)R \tag{2.2}$$

where R is the radius of the earth, while h_t and h_r are the height of the aircraft and receiver above the sea level. Using this equation, we can calculate the maximum receiving range for aircraft flying at different altitudes. Figure 2.2 shows the maximum range curves calculated using several receiver heights.

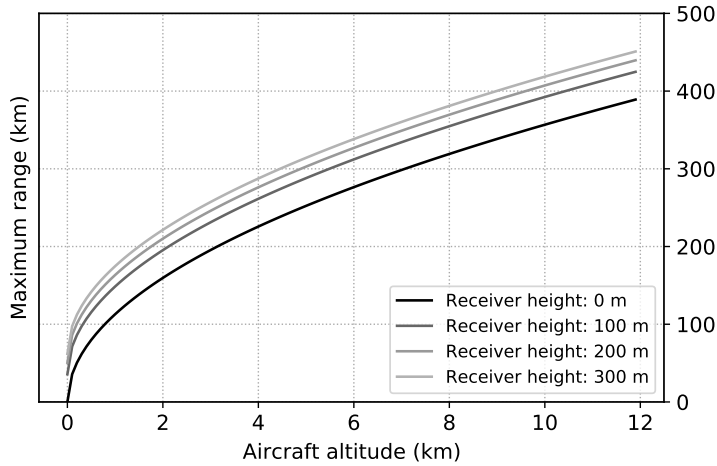


Figure 2.2: Maximum receiver range curve

It should be noted that this figure shows the maximum radio range due to the curvature of the Earth. However, in real-life applications, Mode S signal follows the Friis transmission model, which states that the maximum distance also depends on the power of the transmitter, as well as the directivities of the transmission and receiving antennas. When considering these factors, the actual radio range for the receiver is typically lower than the theoretical values shown in the previous figure.

2.2 Antenna

In principle, any antenna designed for the radio frequency around 1 GHz can be used for receiving Mode S signals. A large variety of commercial off-the-shelf antennas can be found nowadays.

However, it is not difficult to design your own antenna. The carrier frequency of Mode S is 1090 MHz, which corresponds to the wavelength of 27.5 centimeters. In order to have an antenna that is tuned to this specific frequency, one can design the

antenna simply using a piece of a conductor (metal wire) and a coaxial feeder cable. Figure 2.3 shows a few common antenna designs.

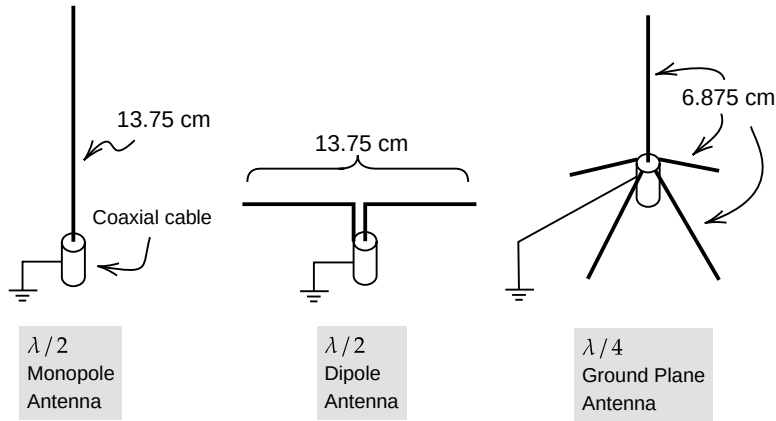


Figure 2.3: Common antenna designs

The monopole antenna and the dipole antenna both are half-wavelength ($\lambda/2$) antennas with a total conductor length of 13.75 cm. The ground plane antenna is a quarter-wavelength ($\lambda/4$) antenna, where the main pole is 6.875 cm.

All these previous antennas are omnidirectional. Some time is also desirable to make use of directional antennas (such a Yagi antenna), for example, to receive messages coming from the airport direction with a higher receiving gain.

For a remote installation where the antenna is far from the receiver (i.e., long coaxial cable between the antenna and receiver), the high-frequency signal of Mode S can attenuate quickly. When the signal has arrived at the receiver end, the signal-to-noise ratio can become quite high. To overcome this limitation, an active antenna may be used.

The active antenna (see Figure 2.4) has a low-noise amplifier integrated on the antenna side. Sometimes, it also includes a band-pass filter that is designed for 1090 MHz. The active antenna requires additional powering. Power is commonly supplied through a bias-tee device connected at the receiver end, which is on the left-hand side of the diagram. On the right-hand side, the radio frequency is mixed with the DC component, which allows the electronic circuit to be powered close to the receiver end.

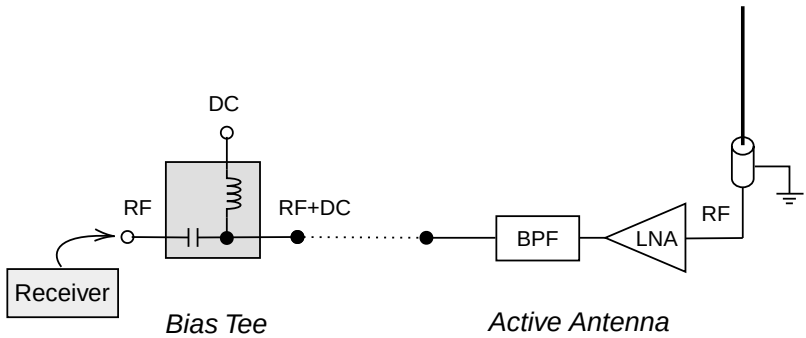


Figure 2.4: Diagram of an active antenna powered through a bias-tee

2.3 Receiver

Many modern Mode S receivers are built using software-defined radio (SDR). An SDR usually can be tuned to a wide range of radio frequencies. The user can conveniently select the center frequency of the receiver and the sampling rate, as well as defined the bandwidth of the onboard low-pass filter if it is supported.

For receiving ADS-B (and other types of Mode S message), a RTL-SDR is the most common type of low-cost receiver to choose. Many different brands of RTL-SDR receivers exist; they are often built with the Realtek RTL2832U chipset. It can work with radio frequencies from 24 to 1766 MHz. The maximum sampling rate is 2.8 million samples per second (MSPS). Given that a Mode S bit consists of a pulse cycle of 0.5 μs (see Figure 1.3), the minimum SDR sampling rate required for Mode S is 2 MSPS. RLT-SDR devices just barely manage to meet this requirement.

Note

The maximum frequency range of RTL-SDR can be from 0 to 2220 MHz. However, the sensitivity drops off significantly outside the 24 – 1766 MHz range. The sampling rate can actually go up to a maximum of 3.2 million samples per second, but some samples may be dropped at this sampling rate. So, it is not stable.

SDRs that offer a higher performance also exist. They often support a larger frequency range with higher sampling rates. Some offer multiple transmitting (TX) and receiving (RX) channels. Table 2.1 lists a few examples of common SDR devices. The performance of RTL-SDR is listed as a comparison.

Table 2.1: Examples of common software defined radio devices

	RTL-SDR	AirSpy R2	HackRF 1	LimeSDR	BladeRF 2
Frequency (from)	24 MHz	24 MHz	1 MHz	100 KHz	47 MHz
Frequency (to)	1766 MHz	1700 MHz	6000 MHz	3800 MHz	6000 MHz
Max sample rate	2.8 MSPS	10 MSPS	20 MSPS	61.44 MSPS	61.44 MSPS
RX channels	1	1	1	2	2
TX channels	0	1	1	2	2
Open hardware	No	Partially	Full	Full	Partially

2.4 Software tools

Several software tools are available to work with some of these SDR devices and decode Mode S and ADS-B signals directly. In this section, we explain how to use two open-source tools, which are `dump1090` and `pyModeS`.

2.4.1 dump1090

`dump1090` is the most well-known open-source Mode S decoder currently available. The software is written in C programming language and was originally developed by Salvatore Sanfilippo [1] under the BSD-3-Clause license.

Since its release in 2013, the repository has been forked and extended intensively on GitHub, creating more than 900 forks so far. Many of the branched versions contain different variations, such as support for additional hardware, additional decoded Mode S messages, and additional output format. Currently, one of the most comprehensive forks is the repository maintained by FlightAware.[2] The following examples will be based on the FlightAware version of `dump1090`, using a Debian-based Linux system.

Installation of dump1090

The latest version of `dump1090` source code can download as follows:

```
$ git checkout https://github.com/flightaware/dump1090.git
```

Before compiling the code, following dependencies must be installed:

```
sudo apt-get install build-essential debhelper librtlsdr-dev \
  pkg-config dh-systemd libncurses5-dev libbladerf-dev
```

Next, we can compile the source code as:

[1] https://github.com/antirez/dump1090
[2] https://github.com/flightaware/dump1090

```
$ cd dump1090
$ make
```

It is worth noting that the FlightAware version of `dump1090` supports both RTL-SDR and BladeRF devices by default.

Default usage of dump1090

Once it is compiled and the RTL-SDR receiver is connected, we can run the following command to start receiving and decoding signals:

```
$ ./dump1090
```

With the default option, Mode S messages and decoded information are displayed in the terminal. For example:

```
*8d451dbd9905b5018004005979c5;
CRC: 000000
RSSI: -20.6 dBFS
Score: 1800
Time: 9214131.08us
DF:17 AA:451DBD CA:5 ME:9905B501800400
 Extended Squitter Airborne velocity over ground, subsonic (19/1)
  ICAO Address:  451DBD (Mode~S / ADS-B)
  Air/Ground:    airborne
  Ground track   271.4
  Groundspeed:   436.1 kt
  Geom rate:     0 ft/min
  NACv:          0
```

Raw Mode S messages

If we only want to display the raw messages, the `--raw` option can be provided, as follows:

```
$ ./dump1090 --raw
```

In this case, the terminal output will only contain the raw messages, for example:

```
*5d4074358ad00c;
*8d407435990dbd01900484f66c3c;
*8d40743558af828cd326fe0c2fe9;
*a80011b1e0da112fe0140060939f;
*a00015b8c2680030a80000318667;
*5d4074358ad030;
*5d4074358ad030;
```

Interactive mode

`dump1090` can also provide a live view of all aircraft seen by the receiver through the use of the `--interactive` option:

```
$ ./dump1090 --interactive
```

An example of the terminal output is shown in Figure 2.5.

Hex	Mode	Sqwk	Flight	Alt	Spd	Hdg	Lat	Long	RSSI	Msgs	Ti
40750D	S	3573	WUK2SH	38000	435	285	51.707	4.833	-27.6	452	0
471EA8	S	5205	WZZ9CS	37025	446	105	51.650	4.404	-32.6	2928	1
471F33	S	2246	WZZ1751	37025	435	106			-34.3	7279	37
0CA56D	S	1347	RYR74PQ	40000	436	266	52.029	4.579	-29.7	7400	0

Figure 2.5: Example of a dump1090 interactive output

2.4.2 pyModeS

`pyModeS` is an open-source Mode S decoder project that I started in 2015. It is a library that was originally designed to focus on lower level inference and decoding of individual raw Mode S messages. Later on, hardware support and live decoding were added, which made it more akin to `dump1090` of late.

> **Note**
>
> Some technical terms related to ADS-B and Mode S are used in this section. They will be explained in detail in later chapters.

Installation of pyModeS

As a Python library, it is possible to install the most recent stable version of `pyModeS` as:

```
pip install --upgrade pyModeS
```

The latest development version for the GitHub can be installed as:

```
pip install --upgrade git+https://github.com/junzis/pyModeS
```

Live traffic view

`pyModeS` provides the possibility to view live traffic through the `modeslive` command. The usage of this command is shown as follows:

```
$ modeslive [-h] --source SOURCE [--connect SERVER PORT DATAYPE]
            [--latlon LAT LON] [--show-uncertainty] [--dumpto DUMPTO]

arguments:
 -h, --help            Show this help message and exit
 --source SOURCE       Choose data source, "rtlsdr" or "net"
 --connect SERVER PORT DATATYPE
                       Define server, port and data type. Supported data
                       types are: ['raw', 'beast', 'skysense']
 --latlon LAT LON      Receiver latitude and longitude, needed for the surface
                       position, default none
 --show-uncertainty    Display uncertainty values, default off
 --dumpto DUMPTO       Folder to dump decoded output, default none
```

When an RTL-SDR device is connected, the following command can be used to display live traffic:

```
$ modeslive --source rtlsdr
```

Data source can also be supplied through the network (for example, from `dump1090` TCP output stream) as:

```
$ dump1090 --net --quiet
$ modeslive --source net --connect localhost 30002 raw
```

An example of the live view is shown in Figure 2.6.

```
Online aircraft [49] ('Ctrl+C' to exit, 'Enter' to lock one)
  icao      call       lat       lon    alt   gs  tas  ias   mach    roc       trk       hdg  live
-------------------------------------------------------------------------------------------------------
06A0AF  QTR7DH__  52.12263   7.88057  40975  494  486  251  0.844    -64    106.31   103.008  -14s
0CA56D  RYR74PQ_   52.2496   8.68286  40000  438  434  229   0.76      0    263.98    266.66  -25s
3C0C99  TUI150__  49.75224   4.25416  36000  438  452  263  0.792    -64     53.44    48.516   -1s
3C4969  TUI2417_  49.23816   4.00952  38000  456  464  256  0.808      0     45.89      45.0    0s
3C4B2B  DLH506__  49.40909   3.88867  33000  481  474  294  0.824      0    256.19   255.059   -4s
3CC187  DCAPB___  49.97695   5.82468  38000  415  420  233  0.744      0     46.75    42.539    0s
-------------------------------------------------------------------------------------------------------
(1 / 7)
```

Figure 2.6: Example of a live view of `modeslive` from `pyModeS`

Real-time flight states are shown, including, for example, ICAO transponder code, call sign, position, altitude, ground speed, true airspeed, indicated airspeed, Mach number, rate of climb, track angle, and heading.

Programming interface

Since `pyModeS` is originally designed as a library aimed for research, and thanks to the use of Python programming language, one can easily make use of its low-level decoding functionalities.

For example, core functions of `pyModeS` can be used to decode Downlink Format,

ICAO address, ADS-B Type Code, as well as to perform parity check:

```
import pyModeS as pms

pms.df(msg)         # Downlink Format
pms.icao(msg)       # Infer the ICAO address from the message
pms.crc(msg)        # Perform parity check
pms.typecode(msg)   # Obtain ADS-B message Type Code
```

ADS-B related functions allow information such as identity, position, and velocities to be decoded. For example:

```
# position messages
pms.adsb.position(msg_even, msg_odd, t_even, t_odd)
pms.adsb.altitude(msg)

# velocity messages
pms.adsb.velocity(msg)
```

There are also a number of functions designed to infer and decode Mode S downlink messages. For example:

```
pms.common.altcode(msg)   # Mode S altitude code (DF=4/20)
pms.common.idcode(msg)    # Mode S squwak code (DF=5/21)
pms.bds.infer(msg)        # Infer Modes S BDS code
```

Once a Mode S message type is identified, type specific functions can be used to decode corresponding parameters.

```
# BDS 4,0
pms.commb.selalt40mcp(msg)  # MCP/FCU selected altitude (ft)
pms.commb.selalt40fms(msg)  # FMS selected altitude (ft)
pms.commb.p40baro(msg)      # Barometric pressure setting (mb)

# BDS 5,0
pms.commb.roll50(msg)       # Roll angle (deg)
pms.commb.trk50(msg)        # True track angle (deg)
pms.commb.gs50(msg)         # Ground speed (kt)
pms.commb.rtrk50(msg)       # Track angle rate (deg/sec)
pms.commb.tas50(msg)        # True airspeed (kt)

# BDS 6,0
pms.commb.hdg60(msg)        # Magnetic heading (deg)
pms.commb.ias60(msg)        # Indicated airspeed (kt)
pms.commb.mach60(msg)       # Mach number (-)
pms.commb.vr60baro(msg)     # Barometric altitude rate (ft/min)
pms.commb.vr60ins(msg)      # Inertial vertical speed (ft/min)
```

These are some commonly used functions related to Mode S decoding. A complete list of the APIs can be found in the `pyModeS` library API documentation.

These functionalities may seem complicated at this moment. In the rest of this book, I will explain all the details related to the decoding of these ADS-B and Mode S messages. Throughout the rest of chapters, I will make use of `pyModeS` to demonstrate the decoding.

Part II

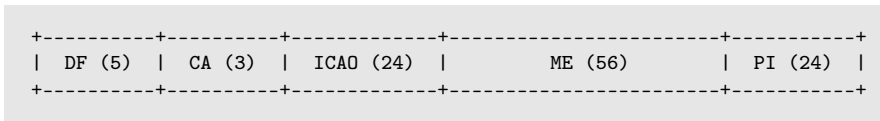
Decoding ADS-B

3 | ADS-B Basics

ADS-B is short for Automatic Dependent Surveillance-Broadcast. It is a satellite-based surveillance system. Parameters such as position, velocity, and identification are transmitted through Mode S Extended Squitter (1090 MHz). Nowadays, the majority of aircraft broadcast ADS-B messages constantly. Starting from the year 2020, civil aviation aircraft in Europe and United States are required to be ADS-B compliant. Old aircraft which are not compliant with ADS-B requirements are required to be retrofitted or phased out within a number of years.

3.1 Message structure

An ADS-B frame is 112 bits long and consists of five main parts, shown as follows:



Civil aircraft ADS-B message starts with the Downlink Format 17. It corresponds to `10001` in binary for the first 5 bits. Bits 6–8 indicated the transponder capability. After that, the 24-bit transponder code (also known as ICAO code) is included. The last two segments are 56-bit payload and 24-bit parity. Table 3.1 lists the key information of an ADS-B message.

Table 3.1: Structure of ADS-B frame

Bit	No. bits	Abbreviation	Information
1–5	5	DF	Downlink Format
6–8	3	CA	Transponder capability
9–32	24	ICAO	ICAO aircraft address
33–88 (33–37)	56 (5)	ME (TC)	Message, extended squitter (Type code)
89–112	24	PI	Parity/Interrogator ID

It is worth noting that the ADS-B Extended Squitter sent from a Mode S transponder uses Downlink Format 17 (`DF=17`). Non-Transponder-based ADS-B Transmitting Subsystems and TIS-B Transmitting equipment use Downlink Format 18 (`DF=18`). By using Downlink Format 18 instead of 17, an ADS-B/TIS-B Receiving Subsystem will know that the message comes from equipment that cannot be interrogated.

3.2 Capability

The second field of an ADS-B message consists of three bits that indicate the transponder level. The capability value can be a decimal value between 0 and 7. The definitions of these values are shown in Table 3.2.

Table 3.2: Mode S transponder capability (CA)

CA	Definition
0	Level 1 transponder
1–3	Reserved
4	Level 2+ transponder, with ability to set CA to 7, on-ground
5	Level 2+ transponder, with ability to set CA to 7, airborne
6	Level 2+ transponder, with ability to set CA to 7, either on-ground or airborne
7	Signifies the Downlink Request value is 0, or the Flight Status is 2, 3, 4, or 5, either airborne or on the ground

3.3 ICAO address

In each ADS-B message, the sender aircraft can be identified using the Mode S transponder code assigned according to ICAO regulations [1]. The Mode S transponder code is also often referred as *ICAO address*, or *hex code*.

The ICAO address is located from 9 to 32 bits in binary (or 3 to 8 in hexadecimal positions). A unique ICAO address is assigned to each Mode S transponder of an aircraft and serves as the unique identifier for each aircraft.

In principle, this code does not change over the lifetime of the aircraft. However, it is possible to reprogram a transponder so that the messages contain a different address. This has been observed for some military aircraft, as well as some private airplanes opt-in for the FAA Privacy ICAO Address System [9].

3.4 ADS-B message types

To identify what information is contained in an ADS-B message, we need to take a look at the Type Code of the message. The Type Code is located at bits 33–37 (or the first 5 bits of the `ME` segment). In the following Table 3.3, the relationships between each Type Code and its information contained in the `ME` segment are shown.

Table 3.3: ADS-B Type Code and content

Type Code	Data frame content
1–4	Aircraft identification
5–8	Surface position
9–18	Airborne position (w/Baro Altitude)
19	Airborne velocities
20–22	Airborne position (w/GNSS Height)
23–27	Reserved
28	Aircraft status
29	Target state and status information
31	Aircraft operation status

3.5 Example of ADS-B message structure

Let us use an example to illustrate the decoding process. First, a raw message is received, which is represented in hexadecimal format:

```
8D4840D6202CC371C32CE0576098
```

It can be converted into binary conveniently. The structure of the binary message is shown as follows:

```
+-----+------------+--------------+----------------------+--------------+
| HEX | 8D         | 4840D6       | 202CC371C32CE0       | 576098       |
+-----+------------+--------------+----------------------+--------------+
| BIN | 10001  101 | 010010000100 | [00100]0000010110011 | 010101110110 |
|     |            | 000011010110 | 0000110111000011100000 | 000010011000 |
|     |            |              | 110010110011100000   |              |
+-----+------------+--------------+----------------------+--------------+
| DEC |  17    5   |              | [4] ..............   |              |
+-----+------------+--------------+----------------------+--------------+
|     |  DF    CA  |   ICAO       |          ME          | PI           |
|     |            |              | [TC] ..............  |              |
+-----+------------+--------------+----------------------+--------------+
```

The first five bits show that the downlink format is `17` (or `10001` in binary), which indicates the message is an ADS-B message. The first five bits of the `ME` field shows that the type code is `4` (or binary `00100`), which indicates the message is an identification message.

In this example, The ICAO address is `4840D6` (`010010000100000011010110` in binary format). Various online tools can be used to find out more about the aircraft with a given ICAO address.[1] For instance, using the previous ICAO `4840D6` example, it will return the result of a `Fokker 70` with the registration of `PH-KZD`.

[1] For example, an online database from OpenSky can be used: https://opensky-network.org/aircraft-database

Try it out

Using pyModeS, we can find out what information is contained in this ADS-B message:

```
import pyModeS as pms
pms.tell("8D4840D6202CC371C32CE0576098")
```

Output:

```
    Message: 8D4840D6202CC371C32CE0576098
ICAO address: 4840D6
Downlink Format: 17
   Protocol: Mode~S Extended Squitter (ADS-B)
       Type: Identitification and category
   Callsign: KLM1023_
```

3.6 Availability and transmission rate

Different ADS-B messages have different transmission rates. The update frequency also differs depending on whether the aircraft is on-ground or airborne, as well as whether the aircraft is still or moving when on the ground. Table 3.4 indicates the transmission rate of these messages.

Table 3.4: ADS-B message transmission rates (ADS-B version 2)

Messages	**TC**	**Ground (still)**	**Ground (moving)**	**Airborne**
Aircraft identification	1–4	0.1 Hz	0.2 Hz	0.2 Hz
Surface position	5–8	0.2 Hz	2 Hz	-
Airborne position	9–18, 20–22	-	-	2 Hz
Airborne velocity	19	-	-	2 Hz
Aircraft status	28	0.2 Hz (*no TCAS RA and Squawk Code change*)		
		1.25 Hz (*change in TCAS RA or Squawk Code*)		
Target states and status	29	-	-	0.8 Hz
Operational status	31	0.2 Hz	0.4 Hz (*no NIC/NAC/SIL change*)	
			1.25 Hz (*change in NIC/NAC/SIL*)	

It is worth noting that for Target states and Operational status messages, when there is a change in some key parameters, the transmission is changed to a higher rate for approximately 24 seconds.

3.7 ADS-B versions

Since ADS-B was first introduced, there have been three different versions of implementations. The main reason for these updates is to include more information in ADS-B. The documentation available on these versions and differences is quite far from user-friendly. The official ICAO 9871 document [13] is confusing to read. In this section, I have put the difference pieces of scattered information together.

There are three ADS-B versions implemented so far, starting from version 0 (specification defined in RTCA document DO-260). Version 1 was introduced around 2008 (DO-260A), and version 2 around 2012 (DO-260B). Version 3 is currently being developed. Next, major changes in version 1 and version 2 are summarized.

3.7.1 From version 0 to 1

The most significant changes in version 1 are new message types and replacing the previous uncertainty indicators with the new integrity indicators. Details of all major changes in ADS-B version 1 are:

- Added Type Code 28 and 31 messages.
  - TC=28: Aircraft status – Emergency/priority status and ACAS RA Broadcast.
  - TC=31: Operational status.
- Removed the *Navigational uncertainty categories* (NUC). Introduced the new *Navigation integrity category* (NIC) and *Surveillance integrity level* (SIL).
  - Both the Type Code and a NIC Supplement bit (NICs) are used to define the NIC.
  - NIC Supplement bit is included in operation status message (TC=31).
- The ADS-B version number is now indicated in operation status message (TC=31).

3.7.2 From version 1 to 2

In version 2, the integrity categories are refined comparing to version 1. Details of major changes in version 2 are:

- Re-defined the structure and content of TC=28 and TC=31 messages.
- Introduced two additional NIC supplement bits.
- `NICa` is defined in operational status messages. (TC=31)
- `NICb` is defined in airborne position messages. (TC=9–18)
- `NICc` is defined in operational status messages. (TC=31)
- Introduced an additional *Horizontal Containment Radius* (Rc) level within NIC=6 of the airborne position message (TC=13).

3.7.3 Identification of the ADS-B Version

There are two steps to check the ADS-B version, due to the fact that ADS-B version information in version 0 is not included in any message.

1. Step 1: Check whether an aircraft is broadcasting ADS-B messages with TC=31 at all. If no message is ever reported, it is safe to assume that the version is version 0.

2. Step 2: If messages with TC=31 are received, check the version numbers located in the bits 41–43 in ME (or bits 73–75 in the message).

After identifying the correct ADS-B version for an aircraft (which does not change often), one can decode related TC=28 and TC=31 messages accordingly.

4 | Aircraft identification and category

Within the group of ADS-B messages, the *Aircraft Identification and Category* message is designed to broadcast the identification (also known as the 'callsign'), and the wake vortex category of the aircraft.

In this message, the Type Code can be from 1 to 4. The 56-bit ME filed consists of 10 parts and is structured as follows:

```
+------+------+------+------+------+------+------+------+------+------+
| TC,5 | CA,3 | C1,6 | C2,6 | C3,6 | C4,6 | C5,6 | C6,6 | C7,6 | C8,6 |
+------+------+------+------+------+------+------+------+------+------+

TC: Type code
CA: Aircraft category
C*: A character
```

Here, number 6 represents the number of bits used to encode each of the characters.

4.1 Identification (call sign)

The aircraft identification included in the message is the callsign. Be aware that callsign is not a unique identifier of an aircraft, since different aircraft flying the same route at different times would share the same callsign.

The last eight fields in the previous structure diagram represent the callsign characters. In order to decode each character, a lookup table is needed to map the corresponding decimal number (represented in binary code) to each character.

The character mapping is shown as follows:

```
#ABCDEFGHIJKLMNOPQRSTUVWXYZ##### ################0123456789######
```

Firstly, the `#` symbols represent characters that are not used. In summary, characters and their decimal representations are as follows, where the `␣` symbol refers to a space character.

```
A - Z :   1 - 26
0 - 9 :  48 - 57
    ␣ :  32
```

If you are familiar with the ASCII (American Standard Code for Information Interchange) code, it is easy to identify that a callsign character is encoded using the lower six bits of the same character in ASCII.

4.2 Wake vortex category

The `CA` value in combination with `TC` value defines the wake vortex category of the aircraft. Table 4.1 lists the definitions and related `TC` and `CA` codes.

Table 4.1: Wake vortex in ADS-B identification and category message

TC	CA	Category
1	ANY	Reserved
ANY	0	No category information
2	1	Surface emergency vehicle
2	3	Surface service vehicle
2	4–7	Ground obstruction
3	1	Glider, sailplane
3	2	Lighter-than-air
3	3	Parachutist, skydiver
3	4	Ultralight, hang-glider, paraglider
3	5	Reserved
3	6	Unmanned aerial vehicle
3	7	Space or transatmospheric vehicle
4	1	Light (less than 7000 kg)
4	2	Medium 1 (between 7000 kg and 34000 kg)
4	3	Medium 2 (between 34000 kg to 136000 kg)
4	4	High vortex aircraft
4	5	Heavy (larger than 136000 kg)
4	6	High performance (>5 g acceleration) and high speed (>400 kt)
4	7	Rotorcraft

It is worth noting that ADS-B has its own definition of wake categories, which is different from the ICAO wake turbulence category definition commonly used in aviation. The relationships of ICAO wake turbulence category (WTC) and ADS-B wake vortex category are:

- ICAO WTC L (Light) is equivalent to ADS-B (TC=4, CA=1).
- ICAO WTC M (Medium) is equivalent to ADS-B (TC=4, CA=2 or CA=3).
- ICAO WTC H (Heavy) or J (Super) is equivalent to ADS-B (TC=4, CA=5).

4.3 Decoding example

Let us use the following raw message as an example to demonstrate the decoding:

```
8D4840D6202CC371C32CE0576098
```

The ME field is:

```
HEX: 202CC371C32CE0
BIN: 00100000001011001100001101110001110000110010110011100000
```

The structure of the ME field can be decomposed as follows:

```
00100 000 001011 001100 001101 110001 110000 110010 110011 100000

 TC   CA    11     12     13     49     48     50     51     32
  4    0     K      L      M      1      0      2      3      _
```

With `TC=4`, we can confirm it is an identification and category message. The decoded identity (callsign) of the aircraft is `KLM1023` (with the trailing space ignored). With `CA=0`, we can see that the aircraft did not transmit any information on the wake vortex category.

Try it out

Using pyModeS, we can decode the callsign as follows:

```
import pyModeS as pms

pms.adsb.category("8D4840D6202CC371C32CE0576098")
# Output: 0

pms.adsb.callsign("8D4840D6202CC371C32CE0576098")
# Output: KLM1023_
```

5 | Airborne position

The aircraft airborne position message is used to broadcast the position and altitude of the aircraft. It has the Type Code 9–18 and 20–22. When Type Code is from 9 to 18, the encoded altitude represents the barometric altitude of the aircraft. When the Type Code is from 20 to 22, the encoded altitude contains the GNSS altitude of the aircraft. The Type Code value is related to the uncertainties in the position, which will be discussed in a later chapter.

The structure of the ADS-B airborne position message ME field is shown as follows:

```
+-------+-------+--------+---------+------+------+-------------+-------------+
| TC, 5 | SS, 2 | SAF, 1 | ALT, 12 | T, 1 | F, 1 | LAT-CPR, 17 | LON-CPR, 17 |
+-------+-------+--------+---------+------+------+-------------+-------------+
```

There are eight fields, and the details of all fields are listed in Table 5.1.

Table 5.1: Airborne position message structure

FIELD		**MSG**	**ME**	**BITS**
Type Code 9–18: with barometric altitude 20–22: with GNSS altitude	TC	33–37	1–5	5
Surveillance status 0: No condition 1: Permanent alert 2: Temporary alert 3: SPI condition	SS	38–39	6–7	2
Single antenna flag	SAF	40	8	1
Encoded altitude	ALT	41–52	9–20	12
Time	T	53	21	1
CPR Format 0: even frame 1: odd frame	F	54	22	1
Encoded latitude	LAT-CPR	55–71	23–39	17
Encoded longitude	LON-CPR	72–88	40–56	17

It is important to emphasize that the encoded latitude and longitude are not the actual latitude and longitude values. Instead, the position information is encoded in a Compact Position Reporting (CPR) format, which requires fewer bits to encode positions with higher resolution. The CPR offers a trade-off between global position ambiguity and local position accuracy. Two types of position messages (identified by the odd and even frame bit) are broadcast alternately. There are two different ways to decode an airborne position based on these messages:

1. **Globally unambiguous position decoding**: Without a known position to start with, using both types of messages to decode the position.
2. **Locally unambiguous position decoding**: Knowing a reference position from previous sets of messages, using only one message for the decoding.

5.1 An over-simplified example

First of all, we will use a simple example to explain the basic ideas behind the CPR position encoding. CPR divides 2D space into two different grids. Two message types are used to encode the positions with different grids.

In the following simple example, we want to encode position $(9, 7)$ in a 16×16 discrete world. Normally, this would require four bits each to encode the x and y coordinates, which is $(1001, 0111)$.

For this simplified algorithm, we first define two different grids. The even grid has a size of 4×4, while the odd grid has a size of 5×5. This is illustrated in Figure 5.1.

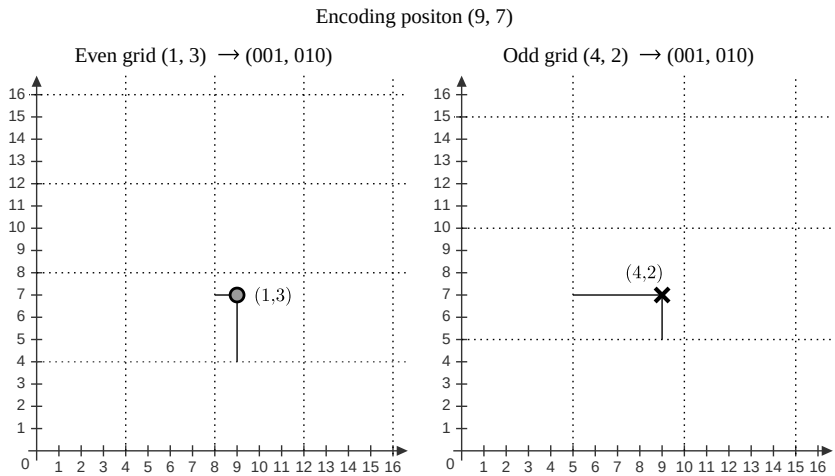


Figure 5.1: Simplified CPR position coding (encoding)

With these two grids, we can encode the local positions within each grid systems, which are $(1, 3)$ and $(4, 2)$, respectively. Now, the position can be encoded only using three bits.

When the odd and even messages are received, each message alone has different possible global positions, which are shown in Figure 5.2.

Combining the possible positions from both even and odd messages, we can recover the global position where the solutions from both grids overlap with each other. This is shown in Figure 5.3.

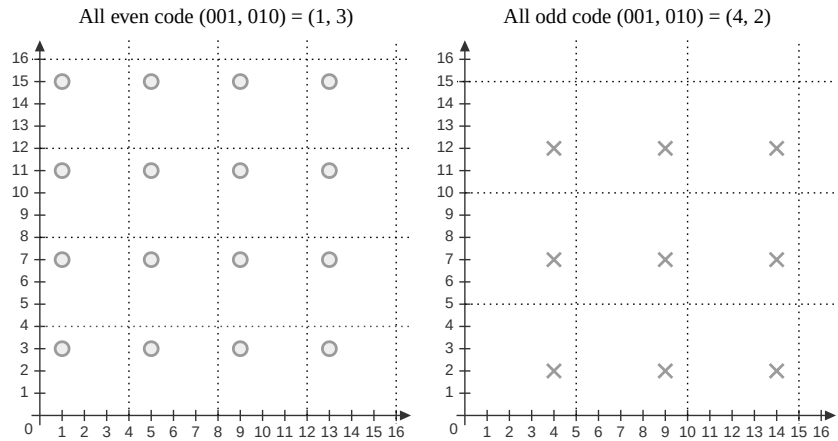


Figure 5.2: Simplified CPR position coding (all position possibilities)

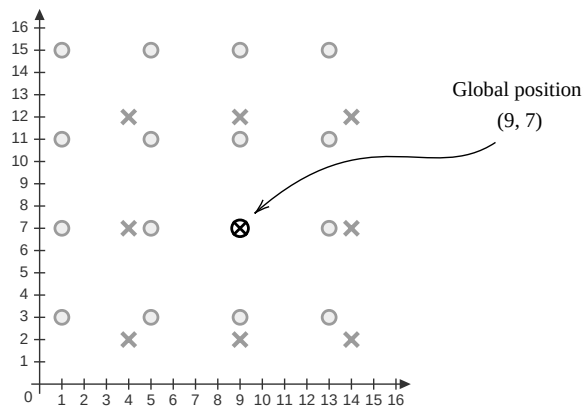


Figure 5.3: Simplified CPR position coding (final global position)

5.2 Compact position reporting

5.2.1 CPR zones

The actual CPR algorithm is more sophisticated. First of all, more zones are defined. There are 15 latitude zones defined for each hemisphere. Up to 59 longitude zones are used, and the number of longitude zones is different at different altitudes.

Figure 5.4 illustrates the latitude zones. The solid and dashed lines represent the latitude zone size of even and odd messages.

Figure 5.5 illustrates the latitude zones. The second plot of the figure is a zoomed-in view of the grid of northern Europe. The black and gray lines represent the

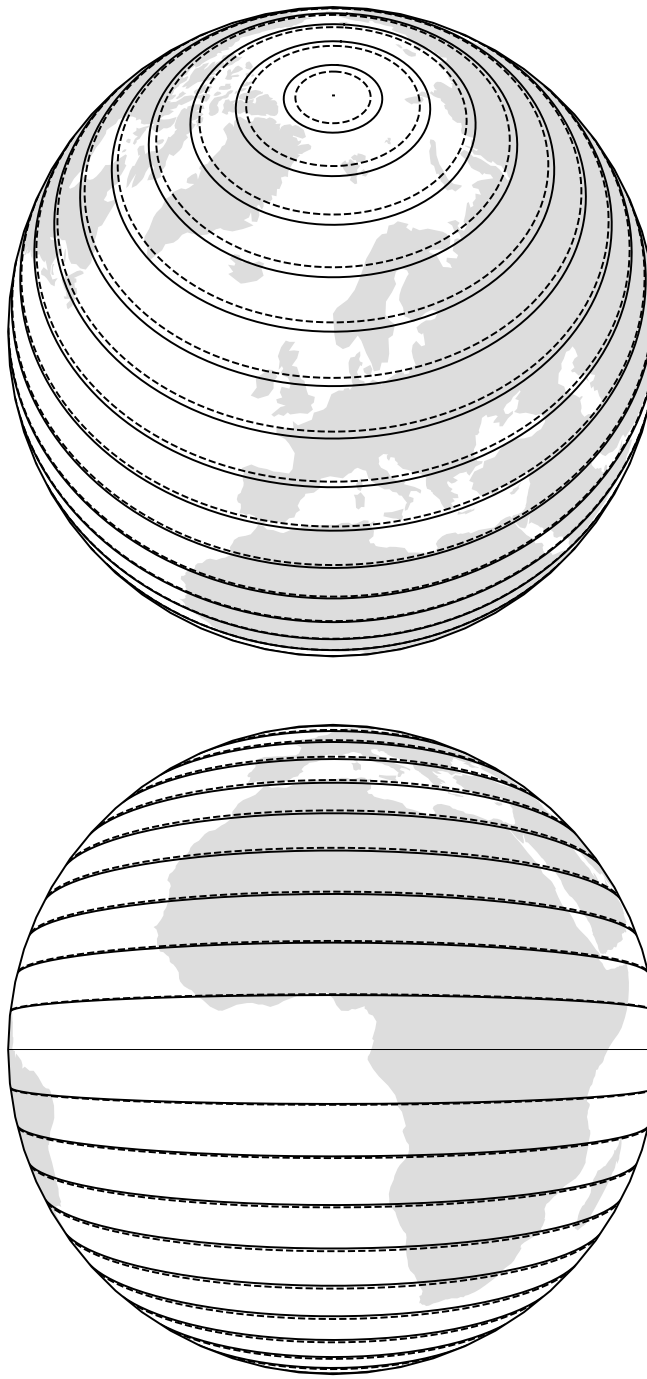


Figure 5.4: Latitude zones in compact position reporting, from different view points.

longitude zone size of even and odd messages respectively. We can see that the number of longitude zones (and the sizes) also differs depending on the latitude.

To report a position, the fractions of latitude and longitude in the respective zones are encoded using 17 bits. These bits are transmitted in the position messages.

5.2.2 Core functions and parameters

To decode the CPR positions, we first introduce relevant parameters and common functions.

Nz

N_Z represents the number of latitude zones between the equator and a pole. In Mode S, N_Z is defined to be 15.

floor(x)

The floor function $floor(x)$ returns the greatest integer value k, where $k \leq x$. For example:

$$\begin{aligned} floor(5.6) &= 5 \\ floor(-5.6) &= -6 \end{aligned} \tag{5.1}$$

mod(x, y)

The modulo function is defined as:

$$mod(x, y) = x - y \cdot floor\left(\frac{x}{y}\right), \qquad y \neq 0 \tag{5.2}$$

NL(lat) - Longitude zone number

Given the latitude, this function yields the number of longitude zones between 1 and 59. The function is expressed as:

$$\mathrm{NL(lat)} = floor\left\{\frac{2\pi}{\arccos\left[1 - \dfrac{1 - \cos\left(\dfrac{\pi}{2\ N_Z}\right)}{\cos^2\left(\dfrac{\pi}{180} \cdot \mathrm{lat}\right)}\right]}\right\} \tag{5.3}$$

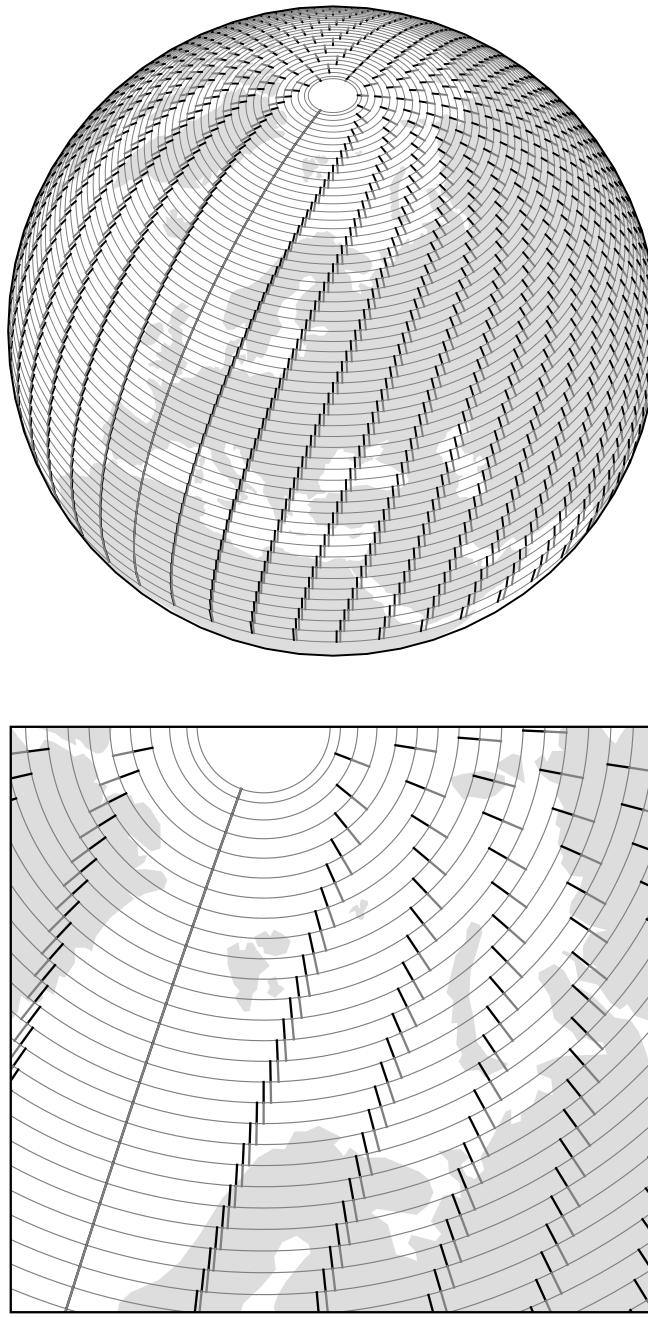


Figure 5.5: Longitude zones in position reporting

For latitudes that are close to the equator or the poles, the following values are used:

```
lat = 0     ->    NL = 59
lat = +87   ->    NL = 2
lat = -87   ->    NL = 2
lat > +87   ->    NL = 1
lat < -87   ->    NL = 1
```

5.3 Globally unambiguous position decoding

5.3.1 Calculation of latitude

In each position message, bit 54 determines whether it is an odd or even frame. `0` indicates an even message, while `1` indicates an odd message.

Bits 55–71 and bits 72–88 represent the fractions of the latitude and longitude within the latitude and longitude grid, denoted as $\mathrm{lat}_{\mathrm{cpr}}$ and $\mathrm{lon}_{\mathrm{cpr}}$:

$$\begin{aligned} \mathrm{lat}_{\mathrm{cpr}} &= \frac{N_{cpr,lat}}{2^{17}} \\ \mathrm{lon}_{\mathrm{cpr}} &= \frac{N_{cpr,lon}}{2^{17}} \end{aligned} \tag{5.4}$$

For even and odd messages, the latitude zone sizes are defined as follows:

$$\begin{aligned} \mathrm{dLat}_{\mathrm{even}} &= \frac{360^\circ}{4N_Z} \\ \mathrm{dLat}_{\mathrm{odd}} &= \frac{360^\circ}{4N_Z - 1} \end{aligned} \tag{5.5}$$

To decode the latitude, first, use the following equation to calculate the latitude zone index, denoted as j:

$$j = floor\left(59 \cdot \mathrm{lat}_{\mathrm{cpr,even}} - 60 \cdot \mathrm{lat}_{\mathrm{cpr,odd}} + \frac{1}{2}\right) \tag{5.6}$$

Based on both even and odd frames, two latitudes are computed as follows:

$$\begin{aligned} \mathrm{lat}_{\mathrm{even}} &= \mathrm{dLat}_{\mathrm{even}}\Big(mod(j, 60) + \mathrm{lat}_{\mathrm{cpr,even}}\Big) \\ \mathrm{lat}_{\mathrm{odd}} &= \mathrm{dLat}_{\mathrm{odd}}\Big(mod(j, 59) + \mathrm{lat}_{\mathrm{cpr,odd}}\Big) \end{aligned} \tag{5.7}$$

For the southern hemisphere, values returned from previous equations range from 270 to 360 degrees. Hence, we need to make sure the latitude is within the range of $[-90, +90]$ by applying the following equations:

$$\begin{aligned} \mathrm{lat}_{\mathrm{even}} &= \mathrm{lat}_{\mathrm{even}} - 360, \quad \text{if } \mathrm{lat}_{\mathrm{even}} \geq 270 \\ \mathrm{lat}_{\mathrm{odd}} &= \mathrm{lat}_{\mathrm{odd}} - 360, \quad \text{if } \mathrm{lat}_{\mathrm{odd}} \geq 270 \end{aligned} \tag{5.8}$$

Before proceeding to the longitude calculation, we need to compute $\mathrm{NL}(\mathrm{lat}_{\mathrm{even}})$ and $\mathrm{NL}(\mathrm{lat}_{\mathrm{odd}})$ to check if both values are the same. If not, this means the pair of messages are from different longitude zones, and it is not possible to compute the correct global position.

In this case, decoding should be stopped, and it is necessary to wait for a pair of messages that are from the same latitude zone. This situation happens when aircraft are flying across the boundaries of longitude zones.

The final latitude is chosen according to the time stamps of the messages as:

$$\mathrm{lat} = \begin{cases} \mathrm{lat}_{\mathrm{even}} & \text{if } T_{\mathrm{even}} \geq T_{\mathrm{odd}} \\ \mathrm{lat}_{\mathrm{odd}} & \text{otherwise} \end{cases} \tag{5.9}$$

where the current latitude is the more recent of these two latitudes.

5.3.2 Calculation of longitude

First, the longitude index, m, can be calculated as:

$$m = floor\left(\mathrm{lon}_{\mathrm{cpr,even}} \cdot \Big[\mathrm{NL}(\mathrm{lat}) - 1\Big] - \mathrm{lon}_{\mathrm{cpr,odd}} \cdot \mathrm{NL}(\mathrm{lat}) + \frac{1}{2}\right) \tag{5.10}$$

We also need to calculate the longitude zone size, which is dependent on the latitude. For even and odd messages, the number of longitude zones is defined as:

$$\begin{aligned} n_{\mathrm{even}} &= max[\mathrm{NL}(\mathrm{lat}), 1] \\ n_{\mathrm{odd}} &= max[\mathrm{NL}(\mathrm{lat} - 1), 1] \end{aligned} \tag{5.11}$$

The longitude zone sizes are defined as follows:

$$\begin{aligned} \mathrm{dLon}_{\mathrm{even}} &= \frac{360^\circ}{n_{\mathrm{even}}} \\ \mathrm{dLon}_{\mathrm{odd}} &= \frac{360^\circ}{n_{\mathrm{odd}}} \end{aligned} \tag{5.12}$$

Then, the longitude is calculated as:

$$\begin{aligned} \mathrm{lon}_{\mathrm{even}} &= \mathrm{dLon}_{\mathrm{even}} \Big[mod(m, n_{\mathrm{even}}) + \mathrm{lon}_{\mathrm{cpr,even}} \Big] \\ \mathrm{lon}_{\mathrm{odd}} &= \mathrm{dLon}_{\mathrm{odd}} \Big[mod(m, n_{\mathrm{odd}}) + \mathrm{lon}_{\mathrm{cpr,odd}} \Big] \end{aligned} \tag{5.13}$$

Similarly, the final longitude is chosen according to the timestamps of the messages as:

$$\mathrm{lon} = \begin{cases} \mathrm{lon}_{\mathrm{even}} & \text{if } T_{\mathrm{even}} \geq T_{\mathrm{odd}} \\ \mathrm{lon}_{\mathrm{odd}} & \text{otherwise} \end{cases} \tag{5.14}$$

It is worth noting that the longitudes in position messages are between 0 and 360 degrees. We often need to convert them to the range between -180 and 180 degrees, which is consistent with aviation conventions. We can convert them as:

$$\mathrm{lon} = \mathrm{lon} - 360, \quad \text{if } \mathrm{lon} \geq 180. \tag{5.15}$$

5.3.3 Decoding example

The example contains two position messages that are received within 10 seconds. The first message is the most recent position message transmitted by the aircraft.

```
8D40621D58C382D690C8AC2863A7   (most recent)
8D40621D58C386435CC412692AD6
```

Following the common ADS-B message structure, we can identify the ICAO address and ME field as follows:

```
+----+--------+----------------+--------+
|    | ICAO   |       ME       |  PI    |
+----+--------+----------------+--------+
| 8D | 40621D | 58C382D690C8AC | 2863A7 |
| 8D | 40621D | 58C386435CC412 | 692AD6 |
+----+--------+----------------+--------+
```

We can covert the message data into binary format:

```
+-------+-----+--------------+---+---+-------------------+-------------------+
| TC    |     | ALT          | T | F | CPR-LAT           | CPR-LON           |
+-------+-----+--------------+---+---+-------------------+-------------------+
| 01011 | 000 | 110000111000 | 0 | 0 | 10110101101001000 | 01100100010101100 |
| 01011 | 000 | 110000111000 | 0 | 1 | 10010000110101110 | 01100010000010010 |
+-------+-----+--------------+---+---+-------------------+-------------------+
```

It is possible to extract the encoded CPR latitude and longitude binary and convert them to decimal format. Then, these values are divided by the 2^{17}, representing the fractions of the positions within the latitude and longitude grids:

$$\begin{aligned} \text{lat}_{\text{cpr,even}} &= \frac{93000}{2^{17}} \\ \text{lon}_{\text{cpr,even}} &= \frac{51372}{2^{17}} \\ \text{lat}_{\text{cpr,odd}} &= \frac{74158}{2^{17}} \\ \text{lon}_{\text{cpr,odd}} &= \frac{50194}{2^{17}} \end{aligned} \tag{5.16}$$

We can calculate the latitude index, j, as:

$$j = floor\left(59 \times \frac{93000}{2^{17}} - 60 \times \frac{51372}{2^{17}} + \frac{1}{2}\right) = 8 \tag{5.17}$$

Then, we can decode the latitudes from both even and odd messages:

$$\begin{aligned} \text{lat}_{\text{even}} &= \frac{360}{4 \times 15}\left[mod(8, 60) + \frac{93000}{2^{17}}\right] = 52.25720214843750 \\ \text{lat}_{\text{odd}} &= \frac{360}{4 \times 15 - 1}\left[mod(8, 59) + \frac{74158}{2^{17}}\right] = 52.26578017412606 \end{aligned} \tag{5.18}$$

After validating the longitude zone of both messages:

$$\begin{aligned} \text{NL}(\text{lat}_{\text{even}}) &= \text{NL}(52.25720214843750) = 36 \\ \text{NL}(\text{lat}_{\text{odd}}) &= \text{NL}(52.26578017412606) = 36 \end{aligned} \tag{5.19}$$

we can continue to calculate the global position. Since the even message is the most recent message, the latitude is:

$$\text{lat} = \text{lat}_{\text{even}} = 52.25720214843750 \tag{5.20}$$

The final longitude (based on the even message) is calculated as:

$$m = floor\left(\frac{51372}{2^{17}} \times (36-1) - \frac{50194}{2^{17}} \times 36 + \frac{1}{2}\right) = 0 \tag{5.21}$$

$$n = max\Big(36-0, 1\Big) = 36 \tag{5.22}$$

$$\text{lon} = \frac{360}{36}\Big[mod(0,36) + \frac{51372}{2^{17}}\Big] = 3.91937255859375 \tag{5.23}$$

Try it out

Using pyModeS, we can perform the previous calculation of globally unambiguous position as:

```
import pyModeS as pms

msg0 = "8D40621D58C382D690C8AC2863A7"
msg1 = "8D40621D58C386435CC412692AD6"
t0 = 1457996402
t1 = 1457996400

pms.adsb.position(msg0, msg1, t0, t1)
```

Output:

```
(52.2572, 3.91937)
```

5.4 Locally unambiguous position decoding

Previously, a globally unambiguous position was decoded using a pair of odd and even messages. This section explains how to decode the ADS-B position with one message and a previous reference position that is obtained using globally unambiguous decoding.

This method offers the possibility to continuous decode aircraft using only one message. It computes the latitude index (j) and the longitude index (m) based on this existing reference position and can be applied to both odd and even messages.

5.4.1 The reference position

The reference position should be close to the actual position, which must be within a 180 NM range. For example, this can be the position of the aircraft that has been decoded in the previous update.[1]

[1] Not knowing the speed of aircraft, it is common to consider a position decoded within 10 seconds as a valid reference position.

The range limitation is to ensure the consistency of the latitude and longitude zones between the reference position and the decoding position. The reference position is denoted as ($\mathrm{lat_{ref}}$, $\mathrm{lon_{ref}}$).

5.4.2 Calculation of latitude

Denote i as the value dependent on the type of the message:

$$i = \begin{cases} 0, & \text{even message} \\ 1, & \text{odd message} \end{cases} \tag{5.24}$$

The latitude zone size is different depending on the message type:

$$\mathrm{dLat} = \frac{360}{4n_z - i} \tag{5.25}$$

Then, the latitude zone index, j, is calculated as:

$$j = floor\left(\frac{\mathrm{lat_{ref}}}{\mathrm{dLat}}\right) + floor\left[\frac{mod(\mathrm{lat_{ref}}, \mathrm{dLat})}{\mathrm{dLat}} - \mathrm{lat_{cpr}} + \frac{1}{2}\right] \tag{5.26}$$

Knowing the latitude zone index, the latitude of the new position is:

$$\mathrm{lat} = \mathrm{dLat} \cdot (j + \mathrm{lat_{cpr}}) \tag{5.27}$$

5.4.3 Calculation of longitude

Next, we can calculate the increment of the longitude per zone based on the decoded latitude, which is dependent on both message type and latitude:

$$\mathrm{dLon} = \frac{360}{max\Big(\mathrm{NL(lat)} - i, 1\Big)} \tag{5.28}$$

Then, the longitude zone index, m, is calculated as:

$$m = floor\left(\frac{\mathrm{lon_{ref}}}{\mathrm{dLon}}\right) + floor\left(\frac{mod(\mathrm{lon_{ref}}, \mathrm{dLon})}{\mathrm{dLon}} - \mathrm{lon_{cpr}} + \frac{1}{2}\right) \tag{5.29}$$

Knowing the longitude zone index, the longitude of the new position is:

$$\mathrm{lon} = \mathrm{dLon} \cdot (m + \mathrm{lon_{cpr}}) \tag{5.30}$$

5.4.4 Decoding example

Next, we illustrate the decoding process of the messages from the previous example with a reference position. The message is:

```
Message:   8D40621D58C382D690C8AC2863A7
ME (data):         58C382D690C8AC
```

The reference position we will use is:

```
(52.258, 3.918)
```

We can easily convert the message data to binary format as:

```
+-------+-----+--------------+---+---+--------------------+--------------------+
| TC    |     | ALT          | T | F | CPR-LAT            | CPR-LON            |
|-------+-----+--------------+---+---+--------------------+--------------------|
| 01011 | 000 | 110000111000 | 0 | 0 | 10110101101001000 | 01100100010101100 |
+-------+-----+--------------+---+---+--------------------+--------------------+
|       |     |              |   |   | 93000              | 51372              |
+-------+-----+--------------+---+---+--------------------+--------------------+
```

Since this is an even message ($i = 0$), we can calculate dLat and latitude index (j) as follows:

$$\mathrm{dLat} = \frac{360}{4 \times 15 - 0} = 6 \tag{5.31}$$

$$j = floor\left(\frac{52.258}{6}\right) + floor\left[\frac{mod(52.258, 6)}{6} - \frac{93000}{2^{17}} + \frac{1}{2}\right] = 8 \tag{5.32}$$

$$\tag{5.33}$$

The latitude can be obtained as:

$$\mathrm{lat} = 6 \times \left(8 + \frac{93000}{2^{17}}\right) = 52.2572021484375 \tag{5.34}$$

Next, we can calculate dLon and longitude index (m) as:

$$\mathrm{dLon} = \frac{360}{max\Big(\mathrm{NL}(\mathrm{lat}_{\mathrm{even}}) - 0, 1\Big)} = \frac{360}{max(36, 1)} = 10 \tag{5.35}$$

$$m = floor\left(\frac{3.918}{10}\right) + floor\left(\frac{mod(3.918, 10)}{10} - \frac{51372}{2^{17}} + \frac{1}{2}\right) = 0 \tag{5.36}$$

$$\tag{5.37}$$

Finally, the aircraft longitude is:

$$\text{lon} = 10 \times \left(0 + \frac{51372}{2^{17}}\right) = 3.91937255859375 \tag{5.38}$$

Try it out

Using pyModeS, we can perform the previous locally unambiguous position calculation as:

```
import pyModeS as pms

msg = "8D40621D58C382D690C8AC2863A7"
lat_ref = 52.258
lon_ref = 3.918

pms.adsb.position_with_ref(msg, lat_ref, lon_ref)
```

Output:

```
(52.2572, 3.91937)
```

5.5 Altitude decoding

The altitude of the aircraft can be decoded with one position message, regardless of it being an even or odd type. However, depending on the type code of the messages, the altitudes are decoded differently. Table 3.3 from Chapter 3 shows two different types of altitudes. They are:

- **TC=9–18**: Airborne position, with barometric altitude encoded (in feet)
- **TC=20–22**: Airborne position, with GNSS Height encoded (in meters)

5.5.1 Barometric altitude

For barometric altitude (TC between 9 and 18), the 8th bit of the 12-bit altitude field is the *Q bit*. It indicates whether the altitude is encoded with an increment of 25 feet or 100 feet.

When $Q = 1$, the altitude is encoded with a 25 feet increment. Removing the Q-bit, the altitude shall be the multiple of 25 feet minus 1000 feet:

$$h = 25\ N - 1000 \quad \text{(ft)} \tag{5.39}$$

For example, based the example message in the previous section, the 12 bits in the altitude field can be read as follows:

```
1100001 1 1000
        ^
      Q-bit
```

Once the Q-bit is removed, the decimal representation of the remaining bits is 1560:

```
11000011000 -> 1560
```

The altitude of the aircraft becomes:

$$1560 \times 25 - 1000 = 38000 \quad \text{(ft)} \tag{5.40}$$

In the case where the altitude is higher than 50175 feet, a 100 feet increment is used. In this situation, the Q bit is set to `0`, and the rest of the bits are encoded using Gray code [5].

It is worth noting that it is possible for all altitude bits to be zeros,which indicates that the altitude information is not available.

5.5.2 GNSS height

When the position message has the Type Code from 20 to 22, the 12-bit altitude field is used for the encoding of the GNSS height. The GNSS height is derived from the global positioning satellites, and the decimal value of all 12 bits translates into the height of aircraft in meters.[2]

Try it out

Using pyModeS, we can perform the altitude decoding as:

```
import pyModeS as pms

msg = "8D40621D58C382D690C8AC2863A7"

pms.adsb.altitude(msg)
```

Output (altitude in feet):

```
38000
```

5.6 Verification of decoded positions

The CPR decoding algorithm does not guarantee the correct decoding of all positions. In rare cases, wrong positions can be calculated.

[2] This SI unit is commonly converted feet for consistency.

It is recommended to perform a so-called *reasonable test* to validate the decoded position. There are a couple of ways to perform such test to examine the plausibility of decoded positions:

1. The first approach is to use the receiver position. The decoded position should not exceed the coverage of the receiver. See Chapter 2, Figure 2.2.
2. A further test is to use more than one pair of different messages to produce multiple globally unambiguous positions. Then, the distance between these positions can be used to examine whether the decoded positions are reasonable.

If the decoded position fails the reasonable tests, it should be discarded.

6 | Surface position

When an aircraft is on the ground, a different type of message is used to broadcast its position information. Unlike the airborne position message, the surface position message also includes the speed of the aircraft. Since no altitude information needs to be transmitted, this provides some extra bits for more information, such as speed and track angle.

The surface position message has Type Code from 5 to 8, which also represents the level of uncertainties in the position. And the structure of the surface position message ME field is:

```
+-------+--------+------+--------+------+------+-------------+-------------+
| TC, 5 | MOV, 7 | S, 1 | TRK, 7 | T, 1 | F, 1 | LAT-CPR, 17 | LON-CPR, 17 |
+-------+--------+------+--------+------+------+-------------+-------------+
```

We can see that the fields are defined similarly to the ones from airborne messages, except for the surveillance status and altitude bits which are replaced by movement and ground track. The details of the fields are shown in Table 6.1

Table 6.1: Surface position message structure

FIELD		**MSG**	**ME**	**BITS**
Type Code	TC	33–37	1–5	5
Movement	MOV	38–44	6–12	7
Status for ground track 0: Invalid 1: Valid	S	45	13	1
Ground track	TRK	46–52	14–20	7
Time	T	53	21	1
CPR Format 0: even frame 1: odd frame	F	54	22	1
Encoded latitude	LAR-CPR	55–71	23–39	17
Encoded longitude	LON-CPR	72–88	40–56	17

6.1 Movement

The movement field encodes the aircraft ground speed. The ground speed of the aircraft is encoded non-linearly and with different quantizations. This is to ensure that a lower speed can be encoded with a improved precision than a higher speed. Different levels of quantizations are defined in Table 6.2.

Table 6.2: Surface position movement (ground speed) decoding

Encoded speed	Ground speed range	Quantization
0	Speed not available	
1	Stopped (v $<$ 0.125 kt)	
2–8	0.125 $\leq$ v $<$ 1 kt	0.125 kt steps
9–12	1 kt $\leq$ v $<$ 2 kt	0.25 kt steps
13–38	2 kt $\leq$ v $<$ 15 kt	0.5 kt steps
39–93	15 kt $\leq$ v $<$ 70 kt	1 kt steps
94–108	70 kt $\leq$ v $<$ 100 kt	2 kt steps
109–123	100 kt $\leq$ v $<$ 175 kt	5 kt steps
124	v $\geq$ 175 kt	
125–127	Reserved	

6.2 Ground track

When the status for the ground track field is set to `1`, the ground track is encoded with a precision of (and rounded to) 360/128 degrees. Zero degrees represents an aircraft ground track that is aligned with the true north. Based on the encoded value (n), the ground track (χ) is calculated as:

$$\chi = \frac{360\ n}{128} \quad \text{(degrees)} \tag{6.1}$$

When the status for the ground track field is set to `0`, the information contained in the ground track fields should be considered as invalid.

6.3 Position

Like coordinates from airborne messages, the surface latitude and longitude are also encoded in the Compact Position Reporting (CPR) format. CPR format bit indicates whether the message is an *even* or *odd* message. There are only a few slight differences in decoding the surface position positions.

First, the latitude zone size is four times smaller, which is defined as:

$$\begin{aligned} \mathrm{dLat}_{\mathrm{even}} &= \frac{90^\circ}{4N_Z} \\ \mathrm{dLat}_{\mathrm{odd}} &= \frac{90^\circ}{4N_Z - 1} \end{aligned} \tag{6.2}$$

Similarly, the longitude zone size is smaller too, which is:

$$\begin{aligned} \mathrm{dLon}_{\mathrm{even}} &= \frac{90^\circ}{n_{\mathrm{even}}} \\ \mathrm{dLon}_{\mathrm{odd}} &= \frac{90^\circ}{n_{\mathrm{odd}}} \end{aligned} \tag{6.3}$$

The algorithm used to encode the position is essentially the same as the one used for encoding airborne position. All the other equations from Chapter 5 can be used to decode surface position.

The main difference, when compared to the airborne position, is that the original number of bits used for encoding surface coordinates is 19 instead of 17. However, in the message, only the lower-order 17 bits are included and transmitted in the surface message. Thus, there can be multiple solutions for a pair of odd and even messages, even when globally unambiguous decoding is used. To identify the correct solution, a reference position needs to be given. This can be the position of the receiver or the location of the airfield.

When employing locally unambiguous decoding, the reference position needs to be within 45 nautical miles of the actual location. Commonly, the positions of the airport is used.

With the equations from section 5.3.1, the solution refers to the latitude located at the northern hemisphere. We need to compute a southern hemisphere solution by subtracting this value with 90 degrees. The final correct latitude is the one that is closest to the reference position.

Similarly, with the equations from section 5.3.2, the solution refers only the longitude in the 0 to 90 degrees quadrant. We need to compute the other three possible solutions by adding 90, 180, and 270 degrees. The final correct longitude is also the one that is closest to the reference position.

6.4 Decoding example

6.4.1 Globally unambiguous decoding

First, we decode the location using globally unambiguous decoding. The following example contains an even/odd pair of surface position messages:

```
8C4841753AAB238733C8CD4020B1
8C4841753A8A35323FAEBDAC702D    (most recent)
```

Based on the ADS-B message structure, we can identify the ICAO address and message ME field as follows:

```
+----+--------+----------------+--------+
|    | ICAO   |       ME       |  PI    |
+----+--------+----------------+--------+
| 8C | 484175 | 3AAB238733C8CD | 4020B1 |
| 8C | 484175 | 3A8A35323FAEBD | AC702D |
+----+--------+----------------+--------+
```

We can covert the message data into binary format:

```
+-------+---------+---+---------+---+---+------------------+------------------+
| TC    | MOV     | S | TRK     | T | F | CPR-LAT          | CPR-LON          |
+-------+---------+---+---------+---+---+------------------+------------------+
| 00111 | 0101010 | 1 | 0110010 | 0 | 0 | 11100001110011001 | 11100100011001101 |
| 00111 | 0101000 | 1 | 0100011 | 0 | 1 | 01001100100011111 | 11010111010111101 |
+-------+---------+---+---------+---+---+------------------+------------------+
```

Knowing the receiver location:

```
reference latitude: 51.990
reference longitude: 4.375
```

it is possible to extract the encoded CPR latitude and longitude binary and convert them to decimal format. Then, they are divided by the 2^{17}, representing the fractions of the positions within the latitude and longitude grids:

$$\begin{aligned} \text{lat}_{\text{cpr,even}} &= \frac{115609}{2^{17}} \\ \text{lon}_{\text{cpr,even}} &= \frac{116941}{2^{17}} \\ \text{lat}_{\text{cpr,odd}} &= \frac{39199}{2^{17}} \\ \text{lon}_{\text{cpr,odd}} &= \frac{110269}{2^{17}} \end{aligned} \tag{6.4}$$

We can calculate the latitude index, j, as:

$$j = floor\left(59 \times \frac{115609}{2^{17}} - 60 \times \frac{39199}{2^{17}} + \frac{1}{2}\right) = 34 \tag{6.5}$$

Then, we can decode the latitudes from both even and odd messages:

$$\begin{aligned} \text{lat}_{\text{even}} &= \frac{90}{4 \times 15}\left[mod(34, 60) + \frac{115609}{2^{17}}\right] = 52.323040008544920 \\ \text{lat}_{\text{odd}} &= \frac{90}{4 \times 15 - 1}\left[mod(34, 59) + \frac{39199}{2^{17}}\right] = 52.320607072215964 \end{aligned} \tag{6.6}$$

After validating the longitude zone of both messages are the same:

$$\begin{aligned} \mathrm{NL}(\mathrm{lat}_{\mathrm{even}}) &= \mathrm{NL}(52.323040008544920) = 36 \\ \mathrm{NL}(\mathrm{lat}_{\mathrm{odd}}) &= \mathrm{NL}(52.320607072215964) = 36 \end{aligned} \tag{6.7}$$

we can continue to calculate the global position. Since the even message is the most recent message, the northern hemisphere latitude solution is:

$$\mathrm{lat}_N = \mathrm{lat}_{\mathrm{odd}} = 52.320607072215964 \tag{6.8}$$

The southern hemisphere latitude solution is:

$$\begin{aligned} \mathrm{lat}_S &= 52.320607072215964 - 90 \\ &= -37.679392927784036 \end{aligned} \tag{6.9}$$

By comparing the result with the reference latitude (51.990), the final latitude value is the one located in the northern hemisphere.

$$\mathrm{lat} = \mathrm{lat}_N = 52.320607072215964 \tag{6.10}$$

The longitude solution in the first quadrant of 0 to 90 degrees is calculated based on the most recent (odd) message:

$$m = floor\left(\frac{116941}{2^{17}} \times (36-1) - \frac{110269}{2^{17}} \times 36 + \frac{1}{2}\right) = 1 \tag{6.11}$$

$$n = max\Big(36-1, 1\Big) = 35 \tag{6.12}$$

$$\mathrm{lon}_1 = \frac{90}{35}\Big[mod(1, 36-1) + \frac{110269}{2^{17}}\Big] = 4.734734671456474 \tag{6.13}$$

The three other possible solutions are:

$$\begin{aligned} \mathrm{lon}_2 &= \mathrm{lon}_1 + 90 = 94.734734671456474 \\ \mathrm{lon}_3 &= \mathrm{lon}_1 + 180 = 184.734734671456474 \\ \mathrm{lon}_4 &= \mathrm{lon}_1 + 270 = 274.734734671456474 \end{aligned} \tag{6.14}$$

By comparing with the reference longitude (4.375), the final longitude is the one located in the first quadrant. Combining the result of previously obtained latitude,

the final decoded coordinates (rounded up to the 6th decimal position) are:

```
latitude: 52.320607
longitude: 4.734735
```

Try it out

Using pyModeS, we can perform the decoding of this globally unambiguous surface position as:

```
import pyModeS as pms

msg0 = "8C4841753AAB238733C8CD4020B1"
msg1 = "8C4841753A8A35323FAEBDAC702D"
t0 = 1457996410
t1 = 1457996412
lat_ref = 51.990
lon_ref = 4.375

pms.adsb.position(msg0, msg1, t0, t1, lat_ref, lon_ref)
```

Output:

```
(52.32061, 4.73473)
```

Note that the reference coordinates are needed for decoding the surface position. In this case the reference is the coordinates of the airfield.

6.4.2 Locally unambiguous decoding

Secondly, we decode the location using locally unambiguous decoding. In this case, we want to use the position obtained previously as the reference.

The new message received is:

```
8C4841753A9A153237AEF0F275BE
```

The structure of the ADS-B message is:

```
+----+--------+----------------+--------+
|    | ICAO   |      Data      |  PI    |
+----+--------+----------------+--------+
| 8C | 484175 | 3A9A153237AEF0 | F275BE |
+----+--------+----------------+--------+
```

We can covert the message data into binary format:

```
+-------+---------+---+---------+---+---+------------------+------------------+
| TC    | MOV     | S | TRK     | T | F | CPR-LAT          | CPR-LON          |
+-------+---------+---+---------+---+---+------------------+------------------+
| 00111 | 0101001 | 1 | 0100001 | 0 | 1 | 01001100100011011 | 11010111011110000 |
+-------+---------+---+---------+---+---+------------------+------------------+
```

$$\mathrm{dLat} = \frac{90}{4 \times 15 - 1} = \frac{90}{59} \tag{6.15}$$

$$j = floor\left(\frac{52.320607}{90/59}\right) + floor\left[\frac{mod(52.320607, 90/59)}{90/59} - \frac{39195}{2^{17}} + \frac{1}{2}\right] = 34 \tag{6.16}$$

$$\mathrm{lat} = \mathrm{lat_{odd}} = \frac{90}{59} \times \left(34 + \frac{39195}{2^{17}}\right) = 52.32056051997815 \tag{6.17}$$

Next, the longitude can be calculated as follows:

$$\mathrm{dLon} = \frac{90}{max\Big(\mathrm{NL}(\mathrm{lat_{odd}}) - 1, 1\Big)} = \frac{90}{max(36 - 1, 1)} = \frac{90}{35} \tag{6.18}$$

$$m = floor\left(\frac{4.734735}{90/35}\right) + floor\left(\frac{mod(4.734735, 90/35)}{90/35} - \frac{110320}{2^{17}} + \frac{1}{2}\right) = 1 \tag{6.19}$$

$$\mathrm{lon} = \frac{90}{35} \times \left(1 + \frac{110320}{2^{17}}\right) = 4.735735212053571 \tag{6.20}$$

The final decoded coordinates (rounded up to 6 decimal positions) are:

```
latitude: 52.320561
longitude: 4.735735
```

Try it out

Using pyModeS, decode this locally unambiguous surface position as:

```
import pyModeS as pms
msg = "8C4841753A9A153237AEF0F275BE"
lat_ref, lon_ref = 51.990, 4.375
pms.adsb.position_with_ref(msg, lat_ref, lon_ref)
```

Output:

```
(52.32061, 4.73473)
```

6.4.3 Movement and track

As a continuation of the previous example, we can also calculate the speed and track angle from the movement and track fields.

The decimal value of movement `0101001` is 41. Based on Table 6.2, the speed is encoded using 1 kt steps. Thus, the speed of aircraft is:

$$15 + (42 - 39) \times 1 = 17kt \tag{6.21}$$

The decimal value of track angle `0100001` is `33`. The track angle can be calculated as:

$$\frac{360^\circ}{128} \times 33 = 92.8125^\circ \tag{6.22}$$

Try it out

Using pyModeS, decode aircraft speed and track angle as:

```
import pyModeS as pms
msg = "8C4841753A9A153237AEF0F275BE"
pms.adsb.velocity(msg)
```

Output (speed, track angle, vertical speed, tag):

```
17 92.8 0 GS
```

Note that the last two parameters returned by the previous function are vertical speed and a flag indicating whether the speed is ground speed (GS) or airspeed (IAS/TAS).

7 | Airborne velocity

Airborne velocities are all transmitted with Type Code 19 (`TC=19`). The general structure of the message is shown in Table 7.1. Four different subtypes are defined in bits 6–8 of the ME field. All sub-types share a similar overall message structure, with the different definitions from message bits 14 to 35 (or frame bits 46 to 67).

Table 7.1: Airborne velocity message structure

FIELD		MSG	ME	BITS
Type Code TC=19 [Binary: 10011]	TC	33–37	1–5	5
Sub-type	ST	38–40	6–8	3
Intent change flag	IC	41	9	1
IFR capability flag	IFR	42	10	1
Navigation uncertainty category for velocity ADS-B version 0 ADS-B version 1–2	 NUCr NUCv	43–45	11–13	3
Sub-type specific fields Refer to Table 7.2 and Table 7.3		46–67	14–35	22
Source bit for vertical rate 0: GNSS, 1: Barometer	VrSrc	38	36	1
Sign bit for vertical rate 0: Up, 1: Down	Svr	69	37	1
Vertical rate All zeros: no information LSB: 64 ft/min VR = 64 x (Decimal value - 1)	VR	70–78	38–46	9
Reserved		79–80	47–48	2
Sign bit for GNSS and Baro altitudes difference 0: GNSS alt above Baro alt 1: GNSS alt below Baro alt	SDif	81	49	1
Difference between GNSS and Baro altitudes All zeros: no information LSB: 25 ft	dAlt	82–88	50–56	7

Subtypes 1 and 2 are used to report ground speeds of aircraft. Subtypes 3 and 4 are used to report aircraft true airspeed or indicated airspeed. Reporting of airspeed in ADS-B only occurs when aircraft position cannot be determined based on the GNSS system. In the real world, subtype 3 messages are very rare.

Sub-type 2 and 4 are designed for supersonic aircraft. Their message structures are identical to subtypes 1 and 3, but with the speed resolution of 4 kt instead of 1 kt. However, since there are no operational supersonic airliners currently, there is no ADS-B airborne velocity message with sub-type 2 and 4 at this moment.[1]

[1] It is possible that with strong tail wind, the ground speed of an aircraft may appear to be “supersonic” during the flight.

These messages contain more information than just horizontal and vertical velocity. Two other significant types of information are the navigation uncertainty category for velocity, and the difference between the GNSS height and barometric altitude.

Compared to position messages, the decoding of velocity messages is more straightforward since there is no complicated encoding scheme. We will use two example messages to illustrate the decoding process in the rest of this chapter:

```
Message A:  8D4850209944099408381758284F (sub-type 1)
Message B:  8DA05F219B06B6AF189400CBC33F (sub-type 3)
```

Downlink format, ICAO address, ME field, and parity of both messages are as follows:

```
+-----+----+--------+----------------+--------+
+     | DF | ICAO   | ME             |  PI    |
+-----+----+--------+----------------+--------+
+ A   | 8D | 485020 | 99440994083817 | 5B284F |
+-----+----+--------+----------------+--------+
+ B   | 8D | A05F21 | 9B06B6AF189400 | CBC33F |
+-----+----+--------+----------------+--------+
```

Next, we can convert two ME fields into binary and decompose the structure further as follows:

```
Message A (sub-type 1):
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|TC   |ST |IC|IFR|NUC|           |VrSrc|Svr|VR       |RESV|SDif|DAlt   |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|10011|001|0 |1  |000|10000001001|0    |1  |000001110|00  |0   |0010111|
|     |   |  |   |   |10010100000|     |   |         |    |    |       |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|19   |1  |0 |1  |0  |           |0    |1  |14       |0   |0   |23     |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
```

```
Message B (sub-type 3):
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|TC   |ST |IC|IFR|NUC|           |VrSrc|Svr|VR       |RESV|SDif|DAlt   |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|10011|011|0 |0  |000|11010110110|1    |1  |000100101|00  |0   |0000000|
|     |   |  |   |   |10101111000|     |   |         |    |    |       |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
|5    |3  |0 |0  |0  |           |1    |1  |37       |0   |0   |0      |
+-----+---+--+---+---+-----------+-----+---+---------+----+----+-------+
```

Here, the second rows and third rows of the messages are binary and decimal values, respectively.

7.1 Vertical rate

The method for decoding vertical rate and its related parameters are the same for different sub-types.

First of all, the vertical rate source bit `VrSrc` (ME bit 36) indicates the source of the altitude measurements. GNSS altitude is encoded as `0`, while `1` encodes the barometric altitude.

The direction of aircraft vertical movement can be read from `Svr` bit (ME bit 37), with `0` and `1` referring to climb and descent, respectively. The encoded vertical rate value `VR` can be computed using message ME bits 38 to 46. If the 9-bit block contains all zeros, the vertical rate information is not available.

The final vertical speed (VS) is calculated as:

$$\mathrm{VS} = (2S_{vr} - 1) \cdot 64 \cdot (V_R - 1) \tag{7.1}$$

where the final vertical speed has the unit of ft/min.

Based on the example `Message A`, the vertical speed is calculated as:

$$\mathrm{VS}_A = (2 \times 0 - 1) \times 64 \times (37 - 1) = -832 \text{ ft/min} \tag{7.2}$$

which indicates that the aircraft A is descending with a vertical speed of 832 ft/min.

Based on the example `Message B`, the vertical speed is calculated as:

$$\mathrm{VS}_B = (2 \times 0 - 1) \times 64 \times (14 - 1) = -2304 \text{ ft/min} \tag{7.3}$$

which indicates that the aircraft B is descending with a vertical speed of 2304 ft/min.

7.2 GNSS and barometric altitudes difference

The last eight bits of the ME field contain the difference between the barometric and GNSS altitudes. Of these eight bits, the first bit indicates the sign for the value of the remaining bits. When it is `1`, the value is negative. This means the GNSS altitude is below the barometric altitude.

The value is encoded with the increment of 25 feet, and the equation to calculate the altitude difference is:

$$\Delta h = s(n - 1) \times 25 \tag{7.4}$$

where s is the sign and n is the decimal value of the encoded altitude difference.

Based on the example `Message A`, the difference between GNSS and barometric altitude is:

$$\Delta h = 1(23 - 1) \times 25 = 550 \text{ ft} \tag{7.5}$$

When all the bits are zeros (as shown in `Message B`) or ones, this information is not available.

7.3 Sub-type 1 and 2: Ground speed decoding

The velocity information fields of subtype 1 and 2 messages consist of the East-West velocity component and the North-South velocity component, as well as the signs for these two values. The structure of ME bits 14 to 35 is shown in Table 7.2.

Table 7.2: Velocity fields of sub-type 1 and 2 (ME bit 14 to 35)

FIELD		MSG	ME	BITS
Direction for E-W velocity component 0: from West to East 1: from East to West	Dew	46	14	1
East-West velocity component All zeros: no information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1) Unit: Knots	Vew	47–56	15–24	10
Direction for N-S velocity component 0: from South to North 1: from North to South	Dns	57	25	1
North-South velocity component All zeros: No information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1) Unit: Knots	Vns	58–67	26–35	10

Here, we can see that ME bits 14 and 25 are sign bits indicating the directions of each component. Each 10-bit block following these sign bits encodes a velocity component. It is important to point out that when any of the 10-bit blocks contains only zeros, no velocity information is available. For supersonic velocity (sub-type 2), the final velocity components are multiplied by a factor of 4.

To calculate the velocity, four values are needed, which are `Sew`, `Vew`, `Sns`, and `Vns`. It is common to consider speed vector's West-East and South-North component as positive values, which are denoted as V_x and V_y. The formulas for sub-type 1 (subsonic) and sub-type 2 (supersonic) differ by a factor of 4.

For subsonic speed (sub-type 1), the speed components can be computed as:

$$V_x = (-2S_{ew} + 1)(V_{ew} - 1) \tag{7.6}$$
$$V_y = (-2S_{ns} + 1)(V_{ns} - 1) \tag{7.7}$$

For supersonic speed (sub-type 2), the speed components can be computed as:

$$V_x = 4 \cdot (-2S_{ew} + 1)(V_{ew} - 1) \tag{7.8}$$
$$V_y = 4 \cdot (-2S_{ns} + 1)(V_{ns} - 1) \tag{7.9}$$

The final ground speed (V) of aircraft is:

$$V = \sqrt{V_x^2 + V_y^2} \tag{7.10}$$

With these two speed components, the ground track angle (λ) of the aircraft can also be calculated conveniently as:

$$\lambda = arctan2\left(V_x, V_y\right) \cdot \frac{360}{2\pi} \tag{7.11}$$
$$\lambda = mod(\lambda, 360) \tag{7.12}$$

where Equation 7.12 ensures the track angle range of [0, 360) degree, with North at 0 degrees, which is the case for most aviation applications and studies.

Based on the example message A given earlier:

```
Message A, ME bit 14-35
+-----+------------+-----+------------+-------+
| Sew | Vew        | Sns | Vns        | VrSrc |
+-----+------------+-----+------------+-------+
| 1   | 0000001001 | 1   | 0010100000 | 0     |
+-----+------------+-----+------------+-------+
| 1   | 9          | 1   | 160        | 0     |
+-----+------------+-----+------------+-------+
```

the final aircraft speed and track angle are calculated as:

$$V_x = (-2 \times 1 + 1) \times (9 - 1) = -8 \tag{7.13}$$

$$V_y = (-2 \times 1 + 1) \times (160 - 1) = -159 \tag{7.14}$$

$$V = \sqrt{(-9)^2 + (-160)^2} = 159.20 \tag{7.15}$$

$$\lambda = arctan2(-9, -160) \times \frac{360}{2\pi} = 182.88 \tag{7.16}$$

where the aircraft speed is 159.20 kt and the track angle is 182.88 degrees.

7.4 Sub-type 3 and 4: Airspeed decoding

ADS-B velocity messages with sub-type 3 or 4 are broadcast when the ground speed of the aircraft is not known. This could happen when, for example, aircraft positions cannot be obtained due to the low availability of GNSS satellites. Furthermore, instead of West/East and South/North components, speed and heading are encoded directly. Table 7.3 lists the details of message fields.

Table 7.3: Velocity fields of sub-type 3 and 4 (ME bit 14 to 35)

FIELD		MSG	ME	BITS
Status bit for magnetic heading 0: not available 1: available	SH	46	14	1
Magnetic heading LSB: 360/1024 degrees Heading = Decimal value $\times 360/1024$	HDG	47–56	15–24	10
Airspeed type 0: Indicated airspeed (IAS) 1: True airspeed (TAS)	T	57	25	1
Airspeed All zeros: no information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1) Unit: knots	AS	58–67	26–35	10

The magnetic heading (ϕ) of the aircraft is calculated as:

$$\phi = \text{HDG} \cdot \frac{360}{1024} \quad \text{degrees} \tag{7.17}$$

The speed calculations also differ between sub-type 3 and 4:

$$V_{as} = \begin{cases} \text{AS} - 1 & \text{sub-type 3} \\ 4 \cdot (\text{AS} - 1) & \text{sub-type 4} \end{cases} \tag{7.18}$$

Note that the airspeed type bit `T` represents whether the value is the indicated airspeed (IAS, `T=0`) or true airspeed (TAS, `T=1`).

Now, we can apply the decoding process to the previous example `Message B`:

```
Message B, ME bit 14-35
+----+------------+---+------------+
| SH | HDG        | T | AS         |
+----+------------+---+------------+
| 1  | 1010110110 | 1 | 0101111000 |
+----+------------+---+------------+
| 1  | 694        | 1 | 376        |
+----+------------+---+------------+
```

With heading status `SH=1`, we know that the magnetic heading information is available. Thus, the heading and airspeed can be derived as:

$$\phi = 694 \times \frac{360}{1024} = 243.98 \tag{7.19}$$

$$V_{as} = 376 - 1 = 375 \tag{7.20}$$

With airspeed type `T=1`, we can also determine that the speed refers to the true airspeed of the aircraft.

Try it out

Using pyModeS, we can perform the decoding the airborne velocities messages as follows:

```
import pyModeS as pms

msgA = "8D485020994409940838175B284F"
msgB = "8DA05F219B06B6AF189400CBC33F"

vA = pms.adsb.velocity(msgA)
vB = pms.adsb.velocity(msgB)
```

We have the following decoded values.

```
vA: (159, 182.88, -832, 'GS')
vB: (375, 243.98, -2304, 'TAS')
```

Note that the parameters returned by the `velocity()` function are speed, track angle (or heading), vertical speed, and speed type.

8 | Aircraft operation status

The aircraft operational status message is designed to provide various information on an aircraft. It is transmitted with Type Code 31 (`TC=31`). The structures of this message type differ significantly over different ADS-B versions. The message has been defined in all ADS-B versions. But in practice, it is not implemented in ADS-B version 0. From version 1 onward, the operational status includes more information, such as ADS-B version, accuracy, and integrity indicators.

8.1 Version 0

In version 0, the structure of the message is shown in Table 8.1. Two main blocks contain the status information, which includes operational capabilities (16 bits) and operational status (16 bits). Each block contains four parameters, and each parameter is encoded with 4 bits.

Table 8.1: Aircraft operational status (Version 0)

FIELD		MSG	ME	BITS
Type code = 31 (Binary: 11111)	TC	33–37	1–5	5
Sub-type code = 0 (Binary: 000)	ST	38–40	6–8	3
Enroute operational capabilities	CC4	41–44	9–12	4
Terminal area operational capabilities	CC3	45–48	13–16	4
Approach/landing operational capabilities	CC2	49–52	17–21	4
Surface operational capabilities	CC1	53–56	22–24	4
Enroute operational status	OM4	57–60	25–28	4
Terminal area operational status	OM3	61–64	29–32	4
Approach/landing operational status	OM2	65–68	33–36	4
Surface operational status	OM1	69–72	36–39	4
Reserved		73–88	41–56	16

However, even though all the parameter fields are defined, all `CC` and OM bits are reserved in the third and fourth blocks according to [13], which eventually made the definitions unusable for version 0 transponders. As version 0 transponders do not transmit any operation status messages, the absence of TC=31 messages can help identify transponder version 0, which is discussed in Chapter 3.

8.2 Version 1

From version 1 onward, the operational status report is implemented and broadcast by the transponder. Table 8.2 lists the structure of operational message in version 1.

Bits 73 to 86, which are unused in the previous version, are now defined with new parameters. These parameters include ADS-B version number and various indicators

Table 8.2: Aircraft operational status (Version 1)

FIELD		MSG	ME	BITS
Type code = 31 (Binary: 11111)	TC	33–37	1–5	5
Sub-type code 0: airborne, 1: surface, 2–7: reserved	ST	38–40	6–8	3
Capacity class codes ST=0: Airborne capacity class codes (16 bits) ST=1: Surface capacity codes (12 bits) + Length/width code (4 bits)	CC	41–56	9–24	16
Operational mode codes	OM	57–72	25–40	16
ADS-B version number 0: Comply with DOC 9871, Appendix A 1: Comply with DOC 9871, Appendix B 2–7: Reserved	Ver	73–75	41–43	3
NIC supplement	NICs	76	44	1
Navigational accuracy category - position	NACp	77–80	45–48	4
ST=0: Barometric altitude quality **ST=1: Reserved**	BAQ	81–82	49–50	2
Surveillance integrity level	SIL	83–84	51–52	2
ST=0: Barometric altitude integrity **ST=1: Track angle or heading**		85	53	1
Horizontal reference direction	HRD	86	54	1
Reserved		87–88	55–56	2

related to the accuracy of the state measurements broadcast by the aircraft in other types of ADS-B messages.

It is worth noting that the definition of sub-types for some fields allow us to differentiate between airborne status messages and surface status messages.

8.3 Version 2

Compared to the previous update, the changes in the operational status report from version 1 to 2 are minimal. Several fields are renamed and a SIL supplement bit is added to bit-87. The structure is shown in Table 8.3.

Table 8.3: Aircraft operational status (Version 2)

FIELD		**MSG**	**ME**	**BITS**
Type code = 31 (Binary: 11111)	TC	33–37	1–5	5
Sub-type code 0: Airborne status message 1: Surface status message	ST	38–40	6–8	3
Capacity class codes ST=0: Airborne capacity class codes (16 bits) ST=1: Surface capacity codes (12 bits) + Length/width code (4 bits)	CC	41–56	9–24	16
Operational mode codes **ST=0: Airborne operational mode codes** **ST=1: Surface operational mode codes**	OM	57–72	25–40	16
ADS-B version number 0: Comply with DOC 9871, Appendix A 1: Comply with DOC 9871, Appendix B 2: Comply with DOC 9871, Appendix C 3–7: Reserved	Ver	73–75	41–43	3
NIC supplement - A	NICa	76	44	
Navigational accuracy category - position	NACp	77–80	45–48	4
ST=0: Geometric vertical accuracy **ST=1: Reserved**	GVA	81–82	49–50	2
Source integrity level	SIL	83–84	51–52	2
ST=0: Barometric altitude integrity **ST=1: Track angle or heading**		85	53	1
Horizontal reference direction	HRD	86	54	1
SIL supplement	SILs	87	55	1
Reserved		88	56	1

9 Uncertainties in ADS-B

There are many parameters in ADS-B that define the quality of data in the position and velocity reports. These parameters serve as accuracy and confidence indicators for aircraft position and speed information transmitted in ADS-B messages.

With each evolution of the ADS-B versions, these parameters have been renamed and redefined. These updates complicate the understanding and analysis of uncertainties in ADS-B. This chapter is designed to give a better overview of the uncertainty indicators in different ADS-B versions, and it also gives a clear mapping between the the indicators and actual uncertainty values.

9.1 Terminology

In general, there are three types of data quality indicators:

- **Uncertainty indicators**: These indicators are introduced in the ADS-B version 0. Parameter values indicate that at least 95% of measurements are within the allowed uncertainty bounds.
- **Accuracy indicators**: These indicators are first introduced in version 1 and are intended to be a drop-in replacement for uncertainty indicators in version 0. Parameter values also indicate that at least 95% of measurements are within the allowed accuracy bounds.
- **Integrity indicators**: These indicators are also first introduced in version 1, replacing the uncertainty indicators from version 0.

Commonly, two parameters are related to the position and velocity separately in each group of indicators. Table 9.1 shows the six major uncertainty indicators related ADS-B versions and their value ranges.

Table 9.1: Data quality indicators

Indicator	Acronym	Version	Values
Navigation uncertainty category–position	NUCp	0	0–9
Navigation uncertainty category–rate (velocity)	NUCr	0	0–4
Navigation accuracy category–position	NACp	1, 2	0–11
Navigation accuracy category–velocity	NACv	1, 2	0–4
Navigation integrity category	NIC	1, 2	0–11
Surveillance integrity level	SIL	1, 2	0–3

Each quality indicator also relates to a set of parameters that express the exact amount of uncertainties, errors, or probabilities. Table 9.2 lists these parameters.

These parameters are identified by different bits from different messages. Due to the evolution of ADS-B versions, the definitions can also differ. In order to correctly

Table 9.2: Parameters related with uncertainty, accuracy, and integrity

	Parameter	
NUCp (V0)	Horizontal protection limit	HPL
	95% containment radius on horizontal position error	Rc/μ
	95% containment radius on vertical position error	Rc/v
NUCr (V0)	Horizontal velocity error (95%)	HVE
	Vertical velocity error (95%)	VVE
NACp (V1,2)	Estimated position uncertainty (95% horizontal accuracy - p). a.k.a. horizontal figure of merit (HFOM) in GNSS	EPU
	Vertical estimated position uncertainty (95% vertical accuracy - p). a.k.a. vertical figure of merit (VFOM) in GNSS	VEPU
NACv (V1,2)	Horizontal figure of merit (95% horizontal accuracy - v)	HFOMr
	Vertical figure of merit (95% horizontal accuracy - v)	VFOMr
NIC (V1,2)	Horizontal containment radius limit	RC
	Vertical protection limit	VPL
SIL (V1,2)	Probability of exceeding horizontal containment radius	P-RC
	Probability of exceeding vertical integrity containment region	P-VPL

obtain these parameters, the ADS-B transponder version must be identified. This can be based on the logic explained in Chapter 3.

In the rest of this chapter, these parameters are explained in detail according to different ADS-B versions. The changes in the same parameters between different versions are also indicated.

9.2 Version 0

In ADS-B version 0, only uncertainties are defined. Two sets of parameters are related to the position and velocity (or rate) separately.

9.2.1 Navigation uncertainty category - position (NUCp)

NUCp is directly related to ADS-B Type Code with one-to-one mapping. The mapping between Type Code and NUCp in surface position messages and the two types of airborne position messages is shown in Table 9.3.

In general, a higher NUCp number (lower TC number) represents higher confidence in the position measurement. When dealing with position uncertainty, the horizontal protection limit (HPL), the containment radius on horizontal position error (denoted as Rc/μ), and the containment radius on vertical position error (denoted as Rc/v) are used to quantify the uncertainties. All values are shown in Table 9.4.

It is worth noting that, in the case of airborne position with GNSS height (TC=20–22), HPL, and Rc/μ are defined with slight differences compared to other types

Table 9.3: Mapping between TC and NUCp

	Surface position				
TC	0	5	6	7	8
NUCp	0	9	8	7	6

	Airborne position (Barometric altitude)									
TC	9	10	11	12	13	14	15	16	17	18
NUCp	9	8	7	6	5	4	3	2	1	0

	(GNSS altitude)		
TC	20	21	22
NUCp	9	8	0

Table 9.4: NUCp parameters and their values

TC	**NUCp**	**HPL**	**Rc/μ**	**Rc/v**
0	0	N/A	N/A	N/A
5	9	<7.5 m	<3 m	N/A
6	8	<25 m	<10 m	N/A
7	7	<0.1 NM (185 m)	<0.05 NM (93 m)	N/A
8	6	>0.1 NM (185 m)	>0.05 NM (93 m)	N/A
9	9	<7.5 m	<3 m	N/A
10	8	<25 m	<10 m	N/A
11	7	<0.1 NM (185 m)	<0.05 NM (93 m)	N/A
12	6	<0.2 NM (370 m)	<0.1 NM (185 m)	N/A
13	5	<0.5 NM (926 m)	<0.25 NM (463 m)	N/A
14	4	<1 NM (1852 m)	<0.5 NM (926 m)	N/A
15	3	<2 NM (3704 m)	<1 NM (1852 m)	N/A
16	2	<10 NM (18520 m)	<5 NM (9260 m)	N/A
17	1	<20 NM (37040 m)	<10 NM (18520 m)	N/A
18	0	>20 NM (37040 m)	>10 NM (18520 m)	N/A
20	9	<7.5 m	<3 m	<4 m
21	8	<25 m	<10 m	<15 m
22	0	>25 m	>10 m	>15 m

of position messages. In addition, the containment radius for the vertical position (Rc/v) is also defined in this message. This is possible because the altitude is obtained from GNSS sources.

9.2.2 Navigation uncertainty category - rate (NUCr)

The Navigation Uncertainty Category - rate (NUCr) is used to indicate the uncertainty of the horizontal and vertical speeds. The bits representing NUCr can be found in the airborne velocity message (TC=19). The NUCr is located at message

bits 43–45 (or ME bits 11–13), which defines the 95% of the error in horizontal and vertical speed as shown in Table 9.5.

Table 9.5: NUCr parameters and their values

NUCp	HVE (95%)	VVE (95%)
0	N/A	N/A
1	<10 m/s	<15.2 m/s (50 fps)
2	<3 m/s	<4.5 m/s (15 fps)
3	<1 m/s	<1.5 m/s (5 fps)
4	<0.3 m/s	<0.46 m/s (1.5 fps)

9.3 Version 1

In ADS-B version 1, the uncertainty category is removed, replaced by the accuracy category and the integrity category. Both NUCp and NUCr from version 0 no longer exist in version 1. New indicators introduced in version 1 are NACp, NACv, NIC, and SIL.

9.3.1 Navigation integrity category (NIC)

NIC is designed to replace NUCp from version 0, but with more levels included. Like NUCp in version 0, the Navigation Integrity Category (NIC) is related to the Type Code. However, Type Code and NIC no longer have a one-to-one mapping relationship. With more levels defined, a supplemental bit is required to distinguish two levels represented by some of the same Type Codes.

The NIC Supplement bit (NICs) is introduced in the operation status messages (TC=31) and located at the message bit 76 (or ME bit 44). The relationship between TC, NIC, and Rc are listed in Table 9.6.

9.3.2 Navigation accuracy category – position (NACp)

NACp is introduced in ADS-B version 1 as a complementary indicator of NIC. The NACp can be obtained from the operational status message, message frame bits 77–80 (or ME bits 45–48).

With NACp, the 95% horizontal and vertical accuracy bounds can be determined. They are the estimated position uncertainty (EPU) and the vertical estimated position uncertainty (VEPU), which are also known as the horizontal figure of merit (HFOM) and the vertical figure of merit (VFOM) in GNSS.

EPU and Rc in NIC have the following relationship:

Table 9.6: NIC parameters and their values (version 1)

TC	NICs	NIC	RC	VPL
0	N/A	N/A	N/A	N/A
5	0	11	<7.5 m	N/A
6	0	10	<25 m	N/A
7	1	9	<75 m	N/A
	0	8	<0.1 NM (185 m)	N/A
8	0	0	>0.1 NM or Unknown	N/A
9	0	11	<7.5 m	<11 m
10	0	10	<25 m	<37.5 m
11	1	9	<75 m	<112 m
	0	8	<0.1 NM (185 m)	N/A
12	0	7	<0.2 NM (370 m)	N/A
13	0	6	<0.5 NM (926 m)	N/A
	1		<0.6 NM (1111 m)	N/A
14	0	5	<1.0 NM (1852 m)	N/A
15	0	4	<2 NM (3704 m)	N/A
16	1	3	<4 NM (7408 m)	N/A
	0	2	<8 NM (14.8 km)	N/A
17	0	1	<20 NM (37.0 km)	N/A
18	0	0	>20 NM or Unknown	N/A
20	0	11	<7.5 m	<11 m
21	0	10	<25 m	<37.5 m
22	0	0	>25 m	>112 m

$$\begin{aligned} \text{EPU} &\approx \text{Rc}/2.5 \qquad \text{for NACp} \geq 9 \\ \text{EPU} &\approx \text{Rc}/2.0 \qquad \text{for NACp} < 9 \end{aligned} \tag{9.1}$$

NACp and its related parameter values are defined in Table 9.7.

9.3.3 Navigation accuracy category – velocity (NACv)

NACv is introduced in version 1 to replace the NUCv from version 0. The bits are located at the same location and have the same definitions of values. These bits are contained in airborne velocity message (TC=19) message bits 43–45 (or ME bits 11–13). It defines the 95% of the errors in horizontal and vertical speeds. The detailed definitions of the Horizontal Figure of Merit for rate (HFOMr) and Vertical Figure of Merit for rate (HFOMr) are shown in Table 9.8.

Table 9.7: NACp parameters and their values

NACp	EPU (or HFOM)	VEPU (or VFOM)
11	<3 m	<4 m
10	<10 m	<15 m
9	<30 m	<45 m
8	<0.05 NM (93 m)	N/A
7	<0.1 NM (185 m)	N/A
6	<0.3 NM (556 m)	N/A
5	<0.5 NM (926 m)	N/A
4	<1.0 NM (1852 m)	N/A
3	<2 NM (3704 m)	N/A
2	<4 NM (7408 m)	N/A
1	<10 NM (18520 m)	N/A
0	>10 NM or Unknown	N/A

Table 9.8: NACv parameters and their values

NACv	HFOMr	VFOMr
0	N/A	N/A
1	<10 m/s	<15.2 m/s (50 fps)
2	<3 m/s	<4.5 m/s (15 fps)
3	<1 m/s	<1.5 m/s (5 fps)
4	<0.3 m/s	<0.46 m/s (1.5 fps)

9.3.4 Surveillance integrity level (SIL)

SIL is introduced in version 1 and used to indicate the probability of measurements exceeding the containment radius. The SIL value can also be found in the operational status message (TC=31), message bits 83–84 (or ME bits 51–52)

Each SIL value corresponds to two probabilities that describe the horizontal (P-RC) and vertical (P-VPL) components respectively. The definitions are as follows:

Table 9.9: SIL parameters and their values

SIL	P-RC	P-VPL
0	unknown	unknown
1	$< 1 \times 10^{-3}$	$< 1 \times 10^{-3}$
2	$< 1 \times 10^{-5}$	$< 1 \times 10^{-5}$
3	$< 1 \times 10^{-7}$	$< 2 \times 10^{-7}$

The unit for P-RCu and P-VPL can be per flight hour or per sample, except when SIL=3, the unit for P-VPL becomes per 150 seconds per sample.

9.4 Version 2

There are fewer changes between version 1 and version 2 when compared to the earlier update. The major changes are the further refined NIC levels and minor updates of SIL parameters.

9.4.1 Navigation integrity category (NIC)

In version 2, NIC levels can be obtained with three additional supplement bits named NIC supplement bit A (NICa), NIC supplement bit B (NICb), and NIC supplement bit C (NICc). These bits can be found as follows:

- NICa is located in the operational status message (TC=31) (message bit 76 or ME bit 44), which is the same as in ADS-B version 1.
- NICb is located in the airborne position message (TC=9–18), message bit 40 or ME bit 8), where the Single Antenna Flag was located in previous ADS-B versions.
- NICc is also located in the operational status message (TC=31) (message bit 52 or ME bit 20).

With these supplemental bits and the Type Code, the NIC value in version 2 can be calculated. NIC values and their related Rc values are shown in Table 9.10.

9.4.2 Surveillance integrity level (SIL)

In version 2, an additional SIL supplement bit (SILs) is introduced to better distinguish whether the value has the unit of per flight hour or per sample. The SILs bit can also be found in the operational status message, message bit 87 (or ME bit 55). The definitions are:

- SILs=0: probability is *per hour* based
- SILs=1: probability is *per sample* based

The values related to SIL remain the same as shown in Table 9.9.

9.4.3 NACp and NACv

NACp and NACv in version 2 remain the same as in version 1. Related parameters and definitions can be found in the previous Table 9.7 and Table 9.8, respectively.

Table 9.10: NIC parameters and their values (version 2)

TC	NICa	NICb	NICc	NIC	Rc
5	0	N/A	0	11	<7.5 m
6	0	N/A	0	10	<25 m
7	1	N/A	0	9	<75 m
	0	N/A	0	8	<0.1 NM (185 m)
8	1	N/A	1	7	<0.2 NM (370 m)
	1	N/A	0	6	<0.3 NM (556 m)
	0	N/A	1		<0.6 NM (1111 m)
	0	N/A	0	0	>0.6 NM or unknown
9	0	0	N/A	11	<7.5 m
10	0	0	N/A	10	<25 m
11	1	1	N/A	9	<75 m
	0	0	N/A	8	<0.1 NM (185 m)
12	0	0	N/A	7	<0.2 NM (370 m)
13	0	1	N/A	6	<0.3 NM (556 m)
	0	0	N/A		<0.5 NM (926 m)
	1	1	N/A		<0.6 NM (1111 m)
14	0	0	N/A	5	<1.0 NM (1852 m)
15	0	0	N/A	4	<2 NM (3704 m)
16	1	1	N/A	3	<4 NM (7408 m)
	0	0	N/A	2	<8 NM (14.8 km)
17	0	0	N/A	1	<20 NM (37.0 km)
18	0	0	N/A	0	>20 NM or unknown
20	N/A	N/A	N/A	11	<7.5 m
21	N/A	N/A	N/A	10	<25 m
22	N/A	N/A	N/A	0	>25 m

10 | Error control in ADS-B

In almost all digital telecommunications, error control is essential. An adequate error control coding scheme in telecommunication allows the receiver to validate the correctness of the information that has been transmitted. Sometimes, it also provides the receiver with the ability to correct errors. Depending on the characteristics of the communication channel and the types of messages, different error control coding schemes can be adopted.

For error detection, three types of error control codes are broadly used, which are 1) parity, 2) checksum, and 3) cyclic redundancy check (CRC)[7]. Though there are similarities among these concepts, they should not be confused:

- Parity is the simplest form of error detection. Commonly, the message is divided into segments of equal length. One parity bit is added to each segment and sent along with the message. The parity bit ensures either an even or odd number of `1` bits are contained in the segment. It guarantees the detection of at least one-bit errors. A two-dimensional parity check is also common, where additional parity bits are added to the same n-th bit in the segments.
- Checksum bits are longer and computed differently. The message is also first divided into segments of equal length. The checksum is generated by adding the segments using the *ones' complements arithmetic* and sent along with the message. On the receiving side, the same arithmetic can be applied, where the complement of the sum should be zero.
- CRC is an algorithm based on binary polynomial arithmetic. A pre-defined divisor (also known as 'generator') is used to compute the remainder using the binary polynomial division. The CRC remainder is appended to the message. The receiving side can run the same process on the entire message to check whether the remainder is zero. CRC is a much stronger method for error detection. It also offers the ability to correct errors [16].

> **Note**
>
> The CRC *remainder* term is confusingly defined in Mode S literature and standards. Many documents refer it as *parity* or *checksum*. To be consistent with Mode S standards, such as [13, 21], parity, checksum, and CRC remainder are considered to be synonyms in this book. They all refer to the CRC remainder.

10.1 CRC error control

Mode S communications (including ADS-B) use CRC error control coding. In all types of Mode S downlink messages, the last 24 bits are reserved for the CRC remainder. In ADS-B, the CRC remainder is directly appended as the final 24 bits of

the messages. However, in other types of Mode S messages, those bits can be overlaid with other information (such as the ICAO address) using the `XOR` (*exclusive disjunction*) operator before being appended to the message.

A basic CRC workflow is shown in Figure 10.1. The diagram on the left-hand side shows how the CRC remainder is generated on the sender side. The diagram on the right-hand side shows how the same arithmetic can be used on the receiver side to validate the message.

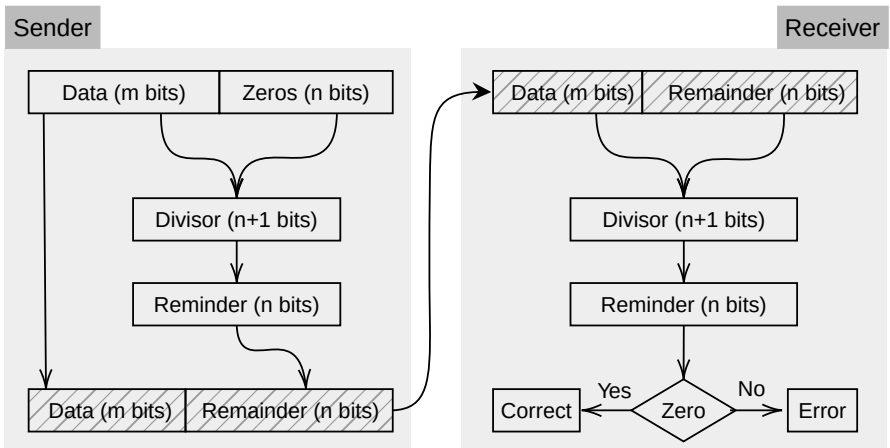


Figure 10.1: The CRC error control processes

The binary polynomial falls in the finite field arithmetic [2]. The divisions of binary polynomials can essentially be considered as the `XOR` bitwise operations, which can be easily implemented and calculated by computers. Figure 10.2 illustrates a CRC remainder calculation example.

10.2 ADS-B parity

The polynomial form of the divisor (generator), $G(x)$, used for ADS-B (as well as other Mode S messages) is [6]:

$$\begin{aligned} G(x) =& x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} \\ &+ x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^3 + 1 \end{aligned} \tag{10.1}$$

This generator code was found by [14], which is known for its efficiency to correct burst errors. It can represented in binary formats as `1111111111111010000001001` (or `1FFF409` in hexadecimal format). A 24-bit CRC remainder is generated using

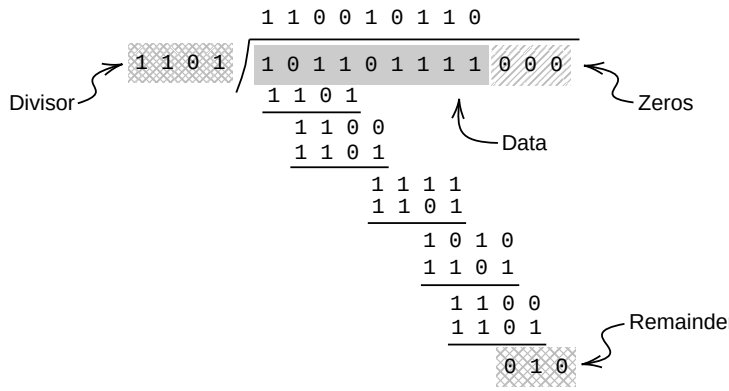


Figure 10.2: A CRC example with a 4-bit divisor and 3-bit remainder

this generator for each ADS-B message.

Knowing the logic of CRC, the computation of the remainder and the verification of the error is fairly simple. Let x^i represent each bit of the message and $M(x)$ represent the polynomial corresponding to the ADS-B message, the CRC remainder (parity) $P(x)$ can thus be calculated as:

$$\begin{aligned} M(x) &= \sum_{i=0}^{87} a_i x^i, \quad a_i \in (0,1) \\ P(x) &= M(x) \,\%\, G(x) \end{aligned} \tag{10.2}$$

In the following pseudocode, the algorithm for computing the remainder of an ADS-B message is shown:

```
generator = 1111111111111010000001001

data_hex = 8D406B902015A678D4D220[000000]  # 11 + 3 zero bytes

data = 10001101010000000011010 11100100000010000000001  # 88 bits
       01011010011001111000011 01010011010010001000000
      [000000000000000000000000]  # append 24 zero bits

FOR i FROM 0 TO (112-24):
  IF data[i] IS 1:
    data[i:i+24] = data[i:i+24] XOR generator

remainder = data[-24:]

# final result: 101010100100101111011010, or AA4BDA in hexadecimal
```

To check whether an error occurs in the message, replace the `data_hex` in the previous example with the received message and run the same CRC process. The message is correct if all final 24 remainder bits are zeros.

Try it out

Using pyModeS, we can check the correctness of a ADS-B message as:

```
import pyModeS as pms

msgA = "8D406B902015A678D4D220AA4BDA"
msgB = "8D4CA251204994B1C36E60A5343D"

remainderA = pms.crc(msgA)  # should be 0
remainderB = pms.crc(msgB)  # should be 16
```

When the remainder is zero, the message is not corrupted. Otherwise, the error has occurred during the transmission.

Part III

Decoding Mode S

11 | Basics of Mode S services

In Chapter 1, an overview of different Mode S services is given. Most of the services except the extended squitter are interrogations based, which means information is only transmitted upon request. The request, also known as '*uplink*', is transmitted using 1030 MHz radio frequency. The reply ('*downlink*') signals are all transmitted using the 1090 MHz radio frequency. Hence, all Mode S downlink messages can be intercepted using the same setup as ADS-B.

In the following chapters of this book, we are going to explain the interrogation based Mode S services in four groups, specifically:

1. All-call reply (DF 11)
2. ACAS short and long replies (DF 0/16)
3. Altitude and identity replies (DF 4/5)
4. Comm-B, with altitude and identity replies (DF 20/21)
   (a) Mode S elementary surveillance (ELS)
   (b) Mode S enhanced surveillance (EHS)
   (c) Meteorological information

In this chapter, we discuss some of the common aspects regarding the decoding of Mode S messages. First of all, the structure of Mode S services is reviewed. Then, Mode S parity and ICAO address recovery are discussed. Finally, some common terminologies related to Comm-B messages are explained.

11.1 Mode S message structures

Based on Figure 1.3 from the introduction chapter, we see that there are two types of Mode S messages in terms of message length. Table 1.1 indicates that among the current 11 different downlink formats, four are short messages consisting of 56 bits. The other seven are long messages with 112 bits. All formats share the same structure of a header consisting of 5 bits of format code and 24 bits parity at the end, as shown in the Figure 11.1.

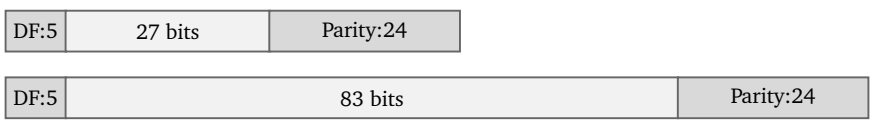


Figure 11.1: The structures of Mode S short and long messages

11.2 Parity

Mode S uses three types of parity. The first one is simply the CRC reminder, which is used for ADS-B in the extended squitter (see Chapter 10). The other two types of parities are Address Parity (AP) and Data Parity (DP). These different parities can be calculated as follows:

1) *Parity*: Let x^i represent each bit of the message and $M(x)$ represent the polynomial corresponding Mode S message, the parity $P(x)$ can thus be calculated as the CRC remainder:

$$\begin{aligned} M(x) &= \sum_{i=0}^{87 \text{ or } 32} a_i x^i, \quad a_i \in (0,1) \\ P(x) &= M(x) \,\%\, G(x) \end{aligned} \tag{11.1}$$

For an extended squitter message, $P(x)$ is directly transmitted as the parity in the last 24 bits of the message. Errors can be detected by performing the same computation process at the receiving side when received parity differs from this newly computed remainder.

2) *Address Parity*: However, for other types of Mode S messages, Address Parity is generated by overlaying the normal CRC remainder with the 24-bit transponder ICAO address. Thus, the final parity included in the message is not $P(x)$, but a new parity $P_A(x)$:

$$P_A(x) = P(x) + A(x) \tag{11.2}$$

where $A(x)$ is the polynomial representing the ICAO address of the transponder.

3) *Data Parity*: For some of the Downlink Format 20 and 21 Mode S messages, upon request of the SSR interrogation, another overlay with the Comm-B Data Selector (BDS) number is performed [6]. In this case, the Data Parity, is included in the last 24 bits of the Mode S message and transmitted.

When DP is in use, the parity is overlaid with the modified ICAO address, Modified AA (MA). The MA is calculated as the polynomial addition of ICAO address and BDS code (with 16 bits of zeros appended after), for example:

```
ICAO:          DD33AA      1101 1101 0011 0011 1010 1010

                                        XOR

BDS 4,4        440000      0100 0100 0000 0000 0000 0000
-------------------------------------------------------
Modified AA    9933AA      1001 1001 0011 0011 1010 1010
```

Denoting the modified address as $P_{\mathrm{MA}}(x)$ and the Data Parity as $P_D(x)$, the calculation of $P_D(x)$ is:

$$\begin{aligned} P_D(x) &= P(x) + P_{\mathrm{MA}}(x) \\ &= P(x) + A(x) + D(x) \end{aligned} \tag{11.3}$$

where $D(x)$ is the polynomial representing the BDS code of the Mode S Comm-B message.

11.3 ICAO address recovery

Since the interrogations are not known, information such as the transponder address is not known to third party receivers. The lack of such information makes error detection difficult.

For AP, the message parity field is produced by overlaying the direct parity with the transponder address. Hence, in order to recover the ICAO address, we can simply overlay the received parity (AP) with the parity calculated again from the payload data:

$$A'(x) = P_A(x) + P(x) \tag{11.4}$$

where $P_A(x)$ is the last 24-bit (assuming Address Parity) for the received message.

In order to demonstrate how to recover an ICAO address, we use the following example message:

```
Message:      A0001838CA380031440000F24177
--------------------------------------------
Payload:      A0001838CA380031440000
--------------------------------------------
Parity (AP):                        F24177
```

Figure 11.2 illustrates the ICAO address recovering process. We first use the Mode S CRC algorithm to compute the remainder (parity) from only the payload data `A0001838CA380031440000`. The remainder is found to be `CE2CA7`. Then, by performing the polynomial addition (XOR for bitwise operation) with the actual parity included in the message, the ICAO address can be computed as `3C6DD0`.

For the cases of Data Parity, it is still possible to recover the transponder address using a similar process. However, we need to overlay again the previous result with the BDS code, assuming that the BDS code can be identified from the data:

$$A'(x) = P_D(x) + P(x) + D(x) \tag{11.5}$$

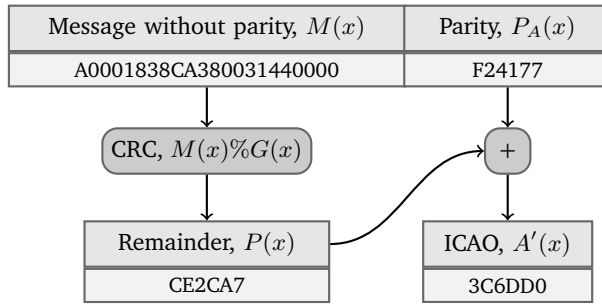


Figure 11.2: The ICAO address recovery logic

where $P_D(x)$ is the last 24-bit (assuming Data Parity) for the received message.

It is worth noting that in either case, the resulting $A'(x)$ is the same as the actual transponder address ($A(x)$) only if no error has occurred during the transmission. If the message is corrupted, the obtained address will be different from the actual one.

By combining with ICAO addresses obtained in ADS-B and other information independently decoded from Mode S messages, this process can also be used for detecting Mode S errors as a third party observer. This method is described in [26].

Try it out

Using pyModeS, we can obtain the ICAO address as:

```
import pyModeS as pms

msg = "A0001838CA380031440000F24177"
pms.icao(msg)
```

Output:

```
3C6DD0
```

11.4 Two's complement coding

Two's complement coding is used for representing negative numbers in some Mode S messages, for example, heading and vertical rates in BDS 5,0 and BDS 6,0 messages. Every parameter in Mode S using two's complement coding include a 1-bit sign and n-bit value.

1) If the sign bit is `0`, the result is simply the decimal representation of the value bits.

For example, we have the following representation of a signed parameter:

```
 sign    value
------------------
  0    111010011
```

The result is the decimal representation of the last nine value bits, which is `467`.

2) If the sign bit is `1`, we first calculate the decimal representation of the value bits (x) and then calculate the negative value as:

$$x - 2^n \tag{11.6}$$

For example, if we changed the signed bit in the previous example to `1`, as follows:

```
 sign    value
------------------
  1    111010011
```

since the sign bit is `1` and there are `9` value bits, the final value is calculated as:

$$467 - 2^9 = -45 \tag{11.7}$$

12 | All-call reply

Chapter 1 shows that Mode S all-call replies can be generated by Mode S transponder to answer Mode S-only all-call interrogations and Mode A/C/S all-call interrogations.

Downlink format 11 is used for the all-call reply, and the length of the messages is 56 bits. The structure of an all-call reply message is quite simple. It only contains four fields, which are shown in Table 12.1.

Table 12.1: All-call reply

FIELD		MSG	BITS
Downlink format	DF	1–5	5
Capability	CA	6–8	3
Address announced	AA	9–32	24
Parity/interrogator identifier	PI	33–56	24

The fields can be decoded as follows:

- CA: The definition of transponder capability is the same as in ADS-B messages (Table 3.2 in Chapter 3).
- AA: The address refers to the 24-bit transponder address.
- PI: The decoding of PI is similar to the decoding of ADS-B parity (Chapter 10).

For a message with downlink format 11, PI is overlaid with the interrogator identifier.[1] Hence, assuming there is no error in the data, the CRC will produce the interrogator identifier code for all-call replies. The interrogator identifier code can be zero. In this case the message is generated due to a spontaneous acquisition squitter (see Chapter 14).

A decoding example is shown as follows:

```
MSG HEX:  5D484FDEA248F5
MSG BIN:  01011   101    01001000010011111011110  10100010010010000111101 01
          [DF=11] [CA=5] [AA=484FDE]              [CRC=22]
```

With the capability value (CA=5), we can see the aircraft has a Level 2+ transponder, with the ability to set CA to 7 and that it is airborne. The CRC remainder indicates the interrogator identifier code is 22.

[1] In fact, PI in ADS-B is also overlaid with the interrogator identifier. However, since ADS-B is not interrogation-based, the identifier is set to 0. Thus, PI is always the same as the CRC remainder in ADS-B.

Try it out

Using pyModeS, we can obtain the ICAO address as:

```
import pyModeS as pms

msg = "5D484FDEA248F5"
pms.icao(msg)
```

Output:

```
484FDE
```

13 | Surveillance replies

Surveillance replies consist of short messages (56 bits) that respond to the selective interrogations of the secondary surveillance radar based on the 24-bit transponder addresses of the aircraft. Two types of information are transmitted: altitude (DF=4) and identity (DF=5).

13.1 Message structure

The structure of altitude and identity surveillance replies are similar, as shown in Tables 13.1 and 13.2. The main difference is that bits 20 to 32 are used to encode either the altitude or the squawk code.

Table 13.1: Surveillance altitude reply (DF=4)

FIELD		**MSG**	**BITS**
Downlink format	DF	1–5	5
Flight status	FS	6–8	3
Downlink request	DR	9–13	5
Utility message	UM	14–19	6
Altitude code	AC	20–32	13
Address parity	AP	33–56	24

Table 13.2: Surveillance identity reply (DF=5)

FIELD		**MSG**	**BITS**
Downlink format	DF	1–5	5
Flight status	FS	6–8	3
Downlink request	DR	9–13	5
Utility message	UM	14–19	6
Identity code	ID	20–32	13
Address parity	AP	33–56	24

The definitions of the common fields are:

- *Flight status (FS)*: 3 bits, shows status of alert, special position pulse (SPI, in Mode A only) and aircraft status (airborne or on-ground). The field is interpreted as:

  `000` : no alert, no SPI, aircraft is airborne
  `001` : no alert, no SPI, aircraft is on-ground
  `010` : alert, no SPI, aircraft is airborne
  `011` : alert, no SPI, aircraft is on-ground
  `100` : alert, SPI, aircraft is airborne or on-ground

`101` : no alert, SPI, aircraft is airborne or on-ground
`110` : reserved
`111` : not assigned

- *Downlink request (DR)*: 5 bits, contains the type of request. In surveillance replies, only values 0, 1, 4, and 5 are used. The field can be decoded as:

  `00000` : no downlink request
  `00001` : request to send Comm-B message
  `00100` : Comm-B broadcast message 1 available
  `00101` : Comm-B broadcast message 2 available

- *Utility message (UM)*: 6 bits, contains transponder communication status information.
  - IIS: The first 4 bits of UM indicate the interrogator identifier code.
  - IDS: The last 2 bits indicate the type of reservation made by the interrogator.

    `00` : no information
    `01` : IIS contains Comm-B interrogator identifier code
    `10` : IIS contains Comm-C interrogator identifier code
    `11` : IIS contains Comm-D interrogator identifier code

13.2 Altitude code

The 13-bit altitude code can be encoded in 25 feet increments, 100 feet increments, or in metric unit of meters. The 7th bit (MSG bit 28) is defined as the M bit. When the M bit is set to `0`, the 9th bit (MSG bit 30) is defined as the Q bit.

The decoding rules are as follows:

- When all bits are set to `0`, altitude information is not available or invalid.
- When M=1, removing the M bit, the remaining 12 bits represent the altitude in meters.
- When M=0 and Q=1, removing the M and Q bits, the remaining 11 bits encode the altitude with 25 feet increments. Denote N as the decimal value of the remaining 11 bits. The altitude is calculated as $25 \times N - 1000$ feet.
- When M=0 and Q=0, the remaining bits are defined as follows:

```
                                  M       Q
+----+----+----+----+----+----+---+----+---+----+----+----+----+
| C1 | A1 | C2 | A2 | C4 | A4 | 0 | B1 | 0 | B2 | D2 | B4 | D4 |
+----+----+----+----+----+----+---+----+---+----+----+----+----+
```

  This structure corresponds to the Mode C altitude reply. This structure is used to encode altitudes above 50187.5 feet.

Try it out

Using pyModeS, we can calculate the altitude code as:

```
import pyModeS as pms

msg = "2000171806A983"
altitude = pms.altcode(msg)  # 36000 (ft)
```

13.3 Identity code

The 13-bit identity code encodes the 4 octal digit squawk code (from `0000` to `7777`). The structure of this field is shown as follows:

```
+----+----+----+----+----+----+---+----+----+----+----+----+----+
| C1 | A1 | C2 | A2 | C4 | A4 | X | B1 | D1 | B2 | D2 | B4 | D4 |
+----+----+----+----+----+----+---+----+----+----+----+----+----+
```

The binary representation of the octal digit is:

```
A4 A2 A1 | B4 B2 B1 | C4 C2 C1 | D4 D2 D1
```

Next, we will use an example to explain the decoding of the identity code. The example message is:

```
MSG HEX: 2A00516D492B80
MSG BIN: 00101 01000000000010 1000101101101 01001001001010111000000
         [DF=5]               [identity code]
```

The binary identity code can be interpreted as follows:

```
C1 A1 C2 A2 C4 A4  X  B1 D1 B2 D2 B4 D4
1  0  0  0  1  0   1   1  0  1  1  0  1
```

Rearranging the bits, we have three groups of binaries:

```
A4 A2 A1 | B4 B2 B1 | C4 C2 C1 | D4 D2 D1
0  0  0  | 0  1  1  | 1  0  1  | 1  1  0
```

Finally, the four octal digit squawk code can be decoded from the binary groups as:

```
0 3 5 6
```

Try it out

Using pyModeS, we can calculate the squawk code as:

```
import pyModeS as pms

msg = "2A00516D492B80"
squawk = pms.idcode(msg)  # 0356
```

14 | Airborne collision avoidance system

14.1 Background

Airborne Collision Avoidance System (ACAS) is a system to reduce the risk of mid-air collisions and near mid-air collisions between aircraft. There are three types of ACAS systems according to [10], which are:

- ACAS I: Gives traffic advisories (TA), without recommending manoeuvres.
- ACAS II: Gives traffic advisories (TA) and resolution advisories (RA) only in vertical directions.
- ACAS III: Gives traffic advisories (TA) and resolution advisories (RA) in both horizontal and vertical directions. ACAS III is not currently implemented

This chapter will focus on ACAS II. The ACAS II is a system that utilizes the aircraft transponder, which interrogates the Mode C and Mode S transponders of nearby aircraft. When threats are detected, corresponding alerts are given to the pilots.

Currently, the most common implementation of ACAS II is *Traffic alert and Collision Avoidance System* (TCAS) II version 7.1, which was initiated by EUROCONTROL. It has been mandatory for aircraft in Europe since 2015. ACAS II works independently of the navigation system, FMS, and ATC. No input from these systems is considered for producing the alerts.

> **Note**
>
> In the future, a new system developed by FAA, called 'ACAS X', is expected to replace the current ACAS II. It makes use of dynamic programming to generate resolution advisories and offers four performance capability variants for different air traffic scenarios. [3]

In ACAS II, TA and RA are triggered when certain thresholds to the closest point of approach (CPA) are passed. The thresholds depend on the altitude, speed, and heading of the aircraft. The examples of the TA and RA regions are illustrated in Figure 14.1.

14.2 ACAS with Mode C transponders

The ACAS system uses Mode C only all-call interrogation to detect aircraft that are only equipped with Mode A/C transponders. In this case, the ACAS system initiates a sequence of interrogations with increasing power.

The interrogation pulse is slightly different from a Mode A/C-only all-call interrogation. A special S_1 pulse (also known as 'whisper-shout') is designed to reduce

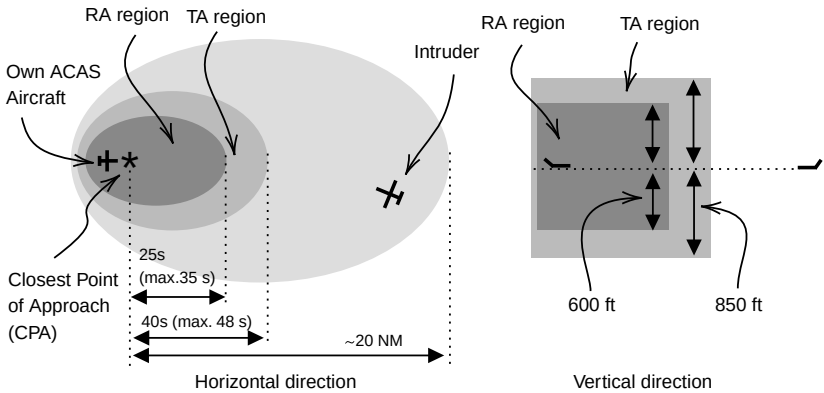


Figure 14.1: Example of an ACAS protection volume in horizontal and vertical directions (between FL50 and FL100)

interference. The S_1 pulse is inserted 2 μs ($\pm 0.15 \mu s$) before the P_1 pulse.

The reply should be sent with the following rules:

- With both S_1 and P_1 above the minimum triggering level (MTL), no reply will be generated.
- With both S_1 and P_1 at MTL, the transponder will respond to 10% of the interrogations.
- With P_1 at MTL and S_1 at MTL - 3 dB, the transponder will respond to 70% of the interrogations.
- With P_1 at MTL and S_1 at MTL - 6 dB, the transponder will respond to 90% of the interrogations.

14.3 ACAS with Mode S transponders

For Mode S transponders, the ACAS system employs a three-phase process, which includes the phases of *detection*, *surveillance*, and *coordination*.

For the detection phase, ACAS passively listens to Mode S only all-call replies (DF=11). These all-call replies are usually generated as a result of ground SSR interrogations, or created by spontaneous acquisition squitter (broadcast of DF=11 message with interrogator code `0`). In this process, aircraft with Mode S transponders in the vicinity are discovered. ACAS may also listen to extended squitter messages (Downlink Format 17, ADS-B) to detect other aircraft.

Once another aircraft is determined to be within the ACAS surveillance range, and within 10,000 ft of the own aircraft[1], ACAS will initiate a short air-air interrogation

[1] This can be acquired through existing DF=0 and DF=4 replies, or through active interrogations

(UF=0) to acquire the range. The interrogation rate is defined as:

- Once every five cycles: when the target aircraft remains in the surveillance range.
- Once every cycle: when the target aircraft is within 3 NM or with time to closest approach less than 60 s.

The surveillance interrogation is stopped when all the following conditions are met:

- A reply (DF=0) is received
- Both aircraft are below 18,000 ft.
- Target aircraft is more than 3 NM and 60 s away from the closest point of approach.

Tables 14.1 and 14.2 lists the fields for ACAS surveillance interrogation and reply messages.

Table 14.1: ACAS surveillance interrogation (uplink), UF=0

FIELD		**MSG**	**BITS**
Uplink Format	UF	1–5	5
Reserved		6–8	3
Reply length	RL	9	1
Reserved		10–13	4
Acquisition	AQ	14	1
Data selector	DS	15–22	8
Reserved		23–32	10
Address parity	AP	33–56	24

Table 14.2: ACAS surveillance reply (downlink), DF=0

FIELD		**MSG**	**BITS**
Downlink Format	DF	1–5	5
Vertical status	VS	6	1
Cross-link capability	CC	7	1
Reserved		8	1
Sensitivity level	SL	9–11	3
Reserved		12–13	2
Reply information	RI	14–17	4
Reserved		18–19	2
Altitude code	AC	20–32	13
Address parity	AP	33–56	24

Once the target aircraft is within the RA region (a threat), ACAS initiates the coordination interrogation (UF=16). In this step, resolution information are transmitted and received through coordination replies (DF=16). Information that is included in both coordination messages is shown in Tables 14.3 and 14.4.

Table 14.3: ACAS coordination interrogation (UF=16)

FIELD		MSG	BITS
Uplink Format	UF	1–5	5
Reserved		6–8	3
Reply length	RL	9	1
Reserved		10–13	4
Acquisition	AQ	14	1
Reserved		15–32	18
Message, U	MU	33–88	56
Address parity	AP	33–56	24

Table 14.4: ACAS coordination reply (DF=16)

FIELD		MSG	BITS
Downlink Format	DF	1–5	5
Vertical status	VS	6	1
Reserved		7–8	2
Sensitivity level	SL	9–11	3
Reserved		12–13	2
Reply information	RI	14–17	4
Reserved		18–19	2
Altitude code	AC	20–32	13
Message, V	MV	33–88	56
Address parity	AP	89–112	24

Specific fields in the above mentioned messages are defined as follows:

- *Reply length (RL)*: 1 bit, it defines the required reply format: `0` requires a reply with DF=0, while `1` requires reply with DF=16.
- *Acquisition (AQ)*: 1 bit, it contains code that controls the content of RI field in the reply.
- *Data selector (DS)*: 8 bits, it indicates the BDS code of the MV content in reply with DF=16.
- *Vertical status (VS)*: 1 bit, it indicates whether the aircraft is airborne (`0`) or on the ground (`1`).
- *Cross-link capability (CC)*: 1 bit, it refers to the capability of reply DF=16 upon request of UF=0. When this 1-bit field is set to `1`, the cross-link is supported. Otherwise, the field is set to `0`.
- *Sensitivity level (SL)*: 3 bits, it represents the sensitivity level of the ACAS system, except that `0` indicates the ACAS is inoperative.
- *Reply information (RI)*: 4 bits, it indicates the type of reply to interrogating aircraft. For ACAS message, valid values are 0 and from 2 to 4. Other values are not part of the ACAS:

`0000` : No operating ACAS
`0010` : ACAS with resolution capability inhibited
`0011` : ACAS with vertical-only resolution capability
`0111` : ACAS with vertical and horizontal resolution capability

- *Altitude Code (AC)*: 13 bits, it encodes the altitude of the aircraft. It can be decoded according to section 13.2.

14.4 ACAS coordination interrogation

Message U-definition (MU) is transmitted in ACAS coordination interrogation. MU is used to transit resolution, ACAS broadcast, and RA broadcast.

14.4.1 UDS=3,0

When ACAS resolution information is transmitted in the UF=16 message, the first 8 bits of MU, U-definition subfields (UDS), are set to `0011 0000` (UDS=3,0). The corresponding fields are indicated in Table 14.5.

Table 14.5: UF=16, MU for ACAS resolution messages, UDS=3,0

FIELD		**MSG**	**MU**	**BITS**
U-definition subfield 1 [0011]	UDS1	33–36	1–4	4
U-definition subfield 2 [0000]	UDS2	37–40	5–8	4
Reserved		41	9	1
Multiple threat bit	MTB	42	10	1
Cancel vertical RAC	CVC	43–44	11–12	2
Vertical RAC	VRC	45–46	13–14	2
Cancel Horizontal RAC	CHC	47–49	15–17	3
Horizontal RAC	HRC	50–52	18–20	3
Reserved		53–55	21–23	3
Horizontal sense bits	HSB	56–60	24–28	5
Vertical sense bits	VSB	61–64	29–32	4
Aircraft address	MID	65–88	33–56	24

These fields can be interpreted as follows:

- *Multiple threat bit (MTB)*: 1 bit, indicates whether multiple threats are present.
- *Vertical RAC* [2] *(VRC)*: 2 bits, contains vertical resolution advisory complement information:

  `00` : No vertical RAC information
  `01` : Do not pass below
  `10` : Do not pass above
  `11` : Not assigned
- *Cancel vertical RAC (CVC)*: 2 bits, cancels previously sent VRC:

[2] RAC: Resolution Advisory Complement

`00` : No cancellation information
`01` : Cancel "Do not pass below"
`10` : Cancel "Do not pass above"
`11` : Not assigned

- *Horizontal RAC (HRC)*: 3 bits, contains horizontal resolution advisory complementary information:

  `000` : No information
  `001` : Other ACAS sense is turn left; do not turn left
  `010` : Other ACAS sense is turn left; do not turn right
  `101` : Other ACAS sense is turn right; do not turn left
  `110` : Other ACAS sense is turn right; do not turn right
  `other` : Not assigned

- *Cancel horizontal RAC (CHC)*: 3 bits, cancels previously sent HRC:

  `000` : No cancellation information
  `001` : Cancel "Do not turn left"
  `010` : Cancel "Do not turn right"
  `other` : Not assigned

- *Horizontal sense bits (HSB)*: 5 bits, uses Hamming code with an extra parity bit to detect errors (up to 3 bits) in CHC and HRC fields.
- *Vertical sense bits (VSB)*: 4 bits, uses Hamming code with an extra parity bit to detect errors (up to 3 bits) in CVC and VRC fields.
- *Aircraft address (MID)*: 24 bits, contains the 24-bits aircraft transponder address of the interrogating ACAS aircraft.

14.4.2 UDS=3,1

When UF=16 is used for RA broadcast, UDS is set to `0011 0001` (UDS=3,1). The corresponding fields are described in Table 14.6.

Table 14.6: UF=16, MU for RA broadcast, UDS=3,1

FIELD		MSG	MU	BITS
U-definition subfield 1 [0011]	UDS1	33–36	1–4	4
U-definition subfield 2 [0001]	UDS2	37–40	5–8	4
Active RAs	ARA	41–54	9–22	14
RAC's record	RAC	55–58	23–26	4
RA terminated indicator	RAT	59	27	1
Multiple threat encounter	MTE	60	28	1
Reserved		61–62	29–30	2
Mode A identity code	AID	63–75	31–43	13
Mode C altitude code	CAC	76–88	44–56	13

These fields can be interpreted as follows:

- *Active RA (ARA)*: 14 bits, indicates the resolution advisory characteristics. It has to be interpreted together with the MTB field.

  - When ARA first bit (MSG bit 41) is `1` and MTE is either `0` or `1`:
    Bit 42: RA is corrective (`1`) or preventive (`0`)
    Bit 43: RA is downward sense (`1`) or upward sense (`0`)
    Bit 44: RA is increased rate (`1`) or not (`0`)
    Bit 45: RA is a sense reversal (`1`) or not (`0`)
    Bit 46: RA is altitude crossing (`1`) or not (`0`)
    Bit 47: RA is positive (`1`) or vertical speed limit (`0`)
    Bit 48–54: Reserved for ACAS III
  - When ARA first bit (MSG bit 41) is `0` and MTE is `1`:
    Bit 42: RA requires a correction in the upward sense (`1`) or not (`0`)
    Bit 43: RA requires a positive climb (`1`) or not (`0`)
    Bit 44: RA requires a correction in the downward sense (`1`) or not (`0`)
    Bit 45: RA requires a positive descent (`1`) or not (`0`)
    Bit 46: RA requires a crossing (`1`) or not (`0`)
    Bit 47: RA is a sense reversal (`1`) or not (`0`)
    Bit 48–54: Reserved for ACAS III
  - When ARA first bit (MSG bit 41) is `0` and MTE is `0`, no vertical RA is generated.

- *RAC's record (RAC)*: 4 bits, contains current active RACs that are received from other ACAS aircraft (if any). Each of the four bits in this field indicates the following RAC when set to `1`. When a bit is set to `0`, the corresponding RAC is inactive.
  Bit 55: Do not pass below
  Bit 56: Do not pass above
  Bit 57: Do not pass left
  Bit 58: Do not pass right
- *RA terminated indicator (RAT)*: 1 bit, indicates whether ACAS is currently generating RA in the ARA field or RA in the ARA field has been terminated.[3]
- *Multiple threat encounter (MTE)*: 1 bit, indicates whether multiple threats are currently being processed by the ACAS resolution. When MTE is set to `0`, either one threat is being processed (ARA bit 41 sets to `1`) or no threat is being processed (ARA bit 41 sets to `0`). When MTE is set to `1`, multiple threats are being processed.
- *Mode A identity code (AID)*: 13 bits, contains the Mode A identity code (sqwake code) of the reporting aircraft. It can be decoded according to section 13.3.
- *Mode C altitude code (CAC)*: 13 bits, contains the Mode C altitude code reporting aircraft. It can be decoded according to section 13.2.

14.4.3 UDS=3,2

When UF=16 is used for ACAS broadcast, UDS is set to `0011 0010` (UDS=3,2). The corresponding fields are indicated in Table 14.7.

[3] The Mode S transponder is still required to report RA 18 seconds after it is terminated by ACAS. Hence, the RAT field is used.

Table 14.7: UF=16, MU for ACAS broadcast, UDS=3,1

FIELD		MSG	MU	BITS
U-definition subfield 1 [0011]	UDS1	33–36	1–4	4
U-definition subfield 2 [0010]	UDS2	37–40	5–8	4
Reserved		41–64	9–32	24
Aircraft address	MID	65–8	33–56	24

The only information that is broadcast in the MU of this message is the 24-bit transponder address of the interrogating aircraft. Its purpose is to inform other aircraft about ACAS capability of the broadcasting aircraft.

14.5 ACAS coordination reply

Message V-definition (MV) is transmitted in ACAS coordination reply message.

14.5.1 VDS=3,0

Similar to MU in the UF=16 message, MV in the DF=16 message contains a few common fields. The corresponding fields are indicated in Table 14.8.

Table 14.8: DF=16, MV for coordinated reply, VDS=3,0

FIELD		MSG	MV	BITS
V-definition subfield 1 [0011]	VDS1	33–36	1–4	4
V-definition subfield 2 [0000]	VDS2	37–40	5–8	4
Active RAs	ARA	41–54	9–22	14
RAC's record	RAC	55–58	23–26	4
RA terminated indicator	RAT	59	27	1
Multiple threat encounter	MTE	60	28	1
Reserved		61–88	29–56	28

We can see that the structure of the MV fields is similar to MU fields of RA broadcast (UDS=3,1) from Table 14.6. The interpretations of these fields are also the same as in section 14.4.2.

14.5.2 Other VDS

When the first eight bits are not `0011 0000`, the MV field contain the Ground-initiated Comm-B information that was requested in DS field of uplink (UF=0) in Table 14.1. Comm-B will be explained in the next chapter.

15 | Comm-B

Comm-B messages count for a large portion of the Mode S selective interrogation responses. The message can have a downlink format of either 20 or 21, depending on whether the aircraft identity code or altitude code is included in the message.

Comm-B protocol supports many different types of messages (up to 255). Several important surveillance services utilize some of the Comm-B message types (see Figure 1.6 from Chapter 1). In this book, we are mainly interested in three types of services, which are Mode S Elementary Surveillance (ELS), Mode S Enhanced Surveillance (EHS), and Meteorological information. By decoding these messages, we can discover some additional information of an aircraft.

15.1 Structure

Comm-B messages have similar structures as surveillance replies (DF=4/5). The structures are shown in the Tables 15.1 and 15.2.

Table 15.1: Comm-B, altitude reply (DF=20)

FIELD		**MSG**	**BITS**
Downlink format	DF	1–5	5
Flight status	FS	6–8	3
Downlink request	DR	9–13	5
Utility message	UM	14–19	6
Altitude code	AC	20–32	13
Message, Comm-B	MB	33–88	56
Parity		89–112	24

Table 15.2: Comm-B, identity reply (DF=21)

FIELD		**MSG**	**BITS**
Downlink format	DF	1–5	5
Flight status	FS	6–8	3
Downlink request	DR	9–13	5
Utility message	UM	14–19	6
Identity code	ID	20–32	13
Message, Comm-B	MB	33–88	56
Parity		89–112	24

The definitions of these common fields are the same as surveillance replies in Chapter 13. In addition, depending on the request in the uplink, either address parity or data parity (see Chapter 11) can be included in the downlink message.

15.2 BDS

Comm-B Data Selector (BDS) is an 8-bit code that determines which information to be included in the MB fields. It is often shown as a 2-digit hexadecimal, for example, `4,0` or `0,A`. We can make a comparison between the BDS code and Type Code used in ADS-B. They both help to identify which structure shall be used to decode the message, except that the BDS code is only included in the uplink (Comm-A). For Comm-B messages, BDS codes are not always included.

Without knowing the BDS code of the downlink message, the information contained in the MB field cannot be decoded. Fortunately, there are methods available that can be used to infer the BDS code for most of the messages and decode them. In later Chapter 19, the inference process will be explained.

The following is the list of BDS codes for messages that will be discussed in detail in the different chapters.

- BDS 1,0 - Data link capability report
- BDS 1,7 - Common usage GICB capability report
- BDS 2,0 - Aircraft identification
- BDS 3,0 - ACAS active resolution advisory
- BDS 4,0 - Selected vertical intention
- BDS 5,0 - Track and turn report
- BDS 6,0 - Heading and speed report
- BDS 4,4 - Meteorological routine air report
- BDS 4,5 - Meteorological hazard report

Here, the first four BDS codes (`1,0` , `1,7` , `2,0` , `3,0`) belong to the ELS service, the next three ones (`4,0` , `5,0` , `6,0`) belong to the EHS services, and the last two codes (`4,4` , `4,5`) report meteorological information. All ELS, EHS and meteorological services are discussed in the following chapters.

It is also worth noting that even ADS-B messages belong to Mode S extended squitter, they are still assigned with BDS codes. The list of BDS codes for these ADS-B messages are:

- BDS 0,5 - Extended squitter airborne position
- BDS 0,6 - Extended squitter surface position
- BDS 0,7 - Extended squitter status
- BDS 0,8 - Extended squitter identification and category
- BDS 0,9 - Extended squitter airborne velocity information

16 | Mode S elementary surveillance

Mode S Elementary Surveillance (ELS) provides a set of basic functionalities that are supported by the Mode S transponders. These basic functionalities include the reporting of aircraft identity, altitude, transponder capability, and flight status. The set of BDS codes included in ELS are `1,0`, `1,7`, `2,0`, and `3,0`.

16.1 Data link capability report (BDS 1,0)

This message is designed to report the data link capability of the installed Mode S transponder. Table 16.1 shows all fields in this message.

Table 16.1: Data link capability report (BDS 1,0), MB field

FIELD	MSG	MB	BITS
BDS Code [0001 0000]	33–40	1–8	8
Configuration flag	41	9	1
Reserved [00000]	42–46	10–14	6
Overlay Command Capability (OCC)	47	15	1
Reserved for ACAS	48	16	1
Mode S subnetwork version number	49–55	17–23	7
Transponder enhanced protocol indicator	56	24	1
Mode S specific services capability	57	25	1
Uplink ELM average throughput capacity	58–60	26–28	3
Downlink ELM throughput	61–64	29–32	4
Aircraft identification capability	65	33	1
Squitter capability subfield (SCS)	66	34	1
Surveillance identifier code (SIC)	67	35	1
Common usage GICB capability report	68	36	1
Reserved for ACAS	69–72	37–40	4
Data terminal equipment (DTE) status	73–80	41–56	16

In the data link capability report, the first eight bits indicate the BDS number, which is `1,0`, or `0001 0000` in binary format.

The definitions of the other fields are explained as follows:

1) ACAS related bits can be decoded as:

```
MSG     MB      Coding
-----------------------------------------------------------------
48      16      0: ACAS failed or on standby
                1: ACAS operating
-----------------------------------------------------------------
69      37      0: Hybrid surveillance not operational
                1: Hybrid surveillance fitted and operational
-----------------------------------------------------------------
70      38      0: ACAS generating TAs only
                1: ACAS generating TAs and RAs
-----------------------------------------------------------------
72|71   40|39   0|0: RTCA/DO-185 (pre-ACAS)
                0|1: RTCA/DO-185A
                1|0: RTCA/DO-185B or EUROCAE ED 143
                1|1: Reserved for future versions
```

2) Valid values for *Mode S subnetwork version number* are from 0 to 5, corresponds to the compliance to a different version of ICAO documentations. Numbers 6 to 127 are currently unassigned. Numbers 0 to 5 can be interpreted as:

```
0:  Subnetwork not available
1:  ICAO Doc 9688 (1996)
2:  ICAO Doc 9688 (1998)
3:  ICAO Annex 10, Vol III, Amdt 77
4:  ICAO Doc 9871 (Ed 1), RTCA DO-181D, EUROCAE ED-73C
5:  ICAO Doc 9871 (Ed 2), RTCA DO-181E, EUROCAE ED-73E
>5: Reserved for future use
```

3) *Overlay Command Capability* indicates whether the transponder supports BDS overlay (Data Parity).

4) When *Transponder enhanced protocol indicator* bit is set to `1`, the transponder is a Level 5 transponder. Value `0` indicates a Level 2 to 4 transponder.

5) When *Mode S specific services capability* bit is set to `1`, at least one Mode S specific service is supported.[1].

6) *Aircraft identification capability* indicates availability of identification (Callsign).

7) When *Squitter capability subfield* bit is set to `1`, both BDS 0,5 and `0,6` registers have been updated in the past 9 to 11 seconds.

8) *Surveillance identifier code* determines whether the transponder has the surveillance identification code capability.

9) *Common usage GICB capability report* is set to `1` every time the GICB capacity report (BDS 1,7) is changed. Register `1,7` is sampled every minute to check for changes.

[1] The additional specific services are BDS codes other than `0,2`, `0,3`, `1,0`, `1,7` - `1,C`, `2,0`, and `3,0`

16.2 Common usage GICB capability report (BDS 1,7)

This message is designed to report Common Usage Ground-initiated Comm-B (GICB) capabilities. The fields in this report are shown in Table 16.2. In this table, each bit indicates whether the corresponding service is available from the transponder.

A bit is set to `1` when the corresponding register has a valid input that has been updated at the required rate. This means that the same aircraft would respond with different GICB reports due to the availability of the relevant data.

Table 16.2: Common usage GICB capability report (BDS 1,0), MB field

FIELD	MSG	MB	BITS
0,5 Extended squitter airborne position	33	1	1
0,6 Extended squitter surface position	34	2	1
0,7 Extended squitter status	35	3	1
0,8 Extended squitter identification and category	36	4	1
0,9 Extended squitter airborne velocity information	37	5	1
0,A Extended squitter event-driven information	38	6	1
2,0 Aircraft identification	39	7	1
2,1 Aircraft registration number	40	8	1
4,0 Selected vertical intention	41	9	1
4,1 Next waypoint identifier	42	10	1
4,2 Next waypoint position	43	11	1
4,3 Next waypoint information	44	12	1
4,4 Meteorological routine report	45	13	1
4,5 Meteorological hazard report	46	14	1
4.8 VHF channel report	47	15	1
5,0 Track and turn report	48	16	1
5,1 Position coarse	49	17	1
5,2 Position fine	50	18	1
5,3 Air-referenced state vector	51	19	1
5,4 Waypoint 1	52	20	1
5,5 Waypoint 2	53	21	1
5,6 Waypoint 3	54	22	1
5,F Quasi-static parameter monitoring	55	23	1
6,0 Heading and speed report	56	24	1
Reserved for aircraft capability	57	25	1
Reserved for aircraft capability	58	26	1
E,1 Reserved for Mode S BITE (Built In Test Equipment)	59	27	1
E,2 Reserved for Mode S BITE (Built In Test Equipment)	60	28	1
F,1 Military applications	61	29	1
Reserved	62–80	30–56	17

We can use the following message as an example:

```
MSG:      A0000638FA81C10000000081A92F
MB:               FA81C100000000
MB BIN:   11111010100000011100000100000000000000000000000000000000
```

From the MB field, we can identify that following bits are set to `1`:

```
Bit: 1, 2, 3, 4, 5, 7, 9, 16, 17, 18, 24
```

Hence, BDS codes supported by the transponder are:

```
BDS05 BDS06 BDS07 BDS08 BDS09     < ADSB
BDS20                             < ELS
BDS40 BDS50 BDS51 and BDS60       < EHS
```

Try it out

Using pyModeS, we can decode the GICB information as:

```
import pyModeS as pms

msg = "A0000638FA81C10000000081A92F"
capabilities = pms.commb.cap17(msg)
```

A list of supporting BDS code will be returned:

```
BDS05, BDS06, BDS07, BDS08, BDS09, BDS20,
BDS40, BDS50, BDS51, BDS52, BDS60
```

16.3 Aircraft identification (BDS 2,0)

Similar to an ADS-B aircraft identification message, the callsign of an aircraft can be decoded from BDS 2,0 messages. The structure of the MB field is defined in Table 16.3.

Table 16.3: Aircraft identification (BDS 2,0), MB field

FIELD	MSG	MB	BITS
BDS Code [0010 0000]	33–40	1–8	8
Character 1	41–46	9–14	6
Character 2	47–52	15–20	6
Character 3	53–58	21–26	6
Character 4	59–64	27–32	6
Character 5	65–70	33–38	6
Character 6	71–76	39–44	6
Character 7	77–82	45–50	6
Character 8	83–88	51–56	6

The first eight bits indicates the BDS code `0010 0000` (`2,0` in hexadecimal). Each of the callsign characters is represented by six bits. We first need to convert the binary numbers to decimals. Each decimal value corresponds to the index of the letter in the following character map:

```
#ABCDEFGHIJKLMNOPQRSTUVWXYZ##### ################0123456789######
```

It is worth noting that this character mapping is the same as the one used for ADS-B identification, which also corresponds to a part of the ASCII code map.

The following is an example on how to decode the callsign from a sample BDS 2,0 message:

```
MSG:  A000083E202CC371C31DE0AA1CCF
MB:           202CC371C31DE0
--------------------------------------------------------------------------------
MB BIN:  0010 0000 001011 001100 001101 110001 110000 110001 110111 100000
HEX:        2    0
DEC:                11     12     13     49     48     49     55     32
CHR:                K      L      M      1      0      1      7     [SPACE]
--------------------------------------------------------------------------------
ID:   KLM1017
```

Try it out

Using pyModeS, we can decode the callsign in this message as:

```
import pyModeS as pms

msg = "A000083E202CC371C31DE0AA1CCF"
callsign = pms.commb.cs20(msg)
```

Callsign `KLM1017_` will be returned by the previous function. The space character is replace by `_` in pyModeS.

16.4 ACAS active resolution advisory (BDS 3,0)

The BDS 3,0 message is used to report resolution advisories (RA) generated by ACAS equipment. The structure of the MB field is defined in Table 16.4.[2]

Table 16.4: ACAS active resolution advisory (BDS 3,0), MB field

FIELD	MSG	MB	BITS
BDS Code [0011 0000]	33–40	1–8	8
Active resolution advisories	41–54	9–22	14
Resolution advisory complements record	55–58	23–26	4
RA terminated	59	27	1
Multiple threat encounter	60	28	1
Threat type indicator	61–62	29–30	2
Threat identity data	63–86	31–56	26

1) The first 8 bits show the BDS code of `3,0` or `0011 0000` in binary format.

2) The 14-bit active resolution advisories (ARA) indicate the characteristics of the RA generated by the ACAS associated with the transponder. To decode the information, we need to decode MB bit 9 (MSG bit 41) and MB bit 28 (MSG bit 60). Table 16.5 shows how to interpret these two bits.

Table 16.5: BDS 3,0 MB bits 9 and 28

MB:9	MB:28	
0	0	No RA has been generated
0	1	Multiple threats, RA is intended to provide vertical separation below some threats and above some other threats
1	0	Only one threat
0	1	Multiple threats, RA is intended to provide vertical separation in the same direction

MB bits 16–22 are reserved for ACAS III. When `MB:9=1`, the meanings of MB bits from 10 to 15 are shown in Table 16.6.

When `MB:9=0` and `MB:28=1`, the meanings of MB bits from 10 to 15 are shown in Table 16.7.

[2] The decoding is also subject to the version of aircraft's TCAS version. For example, TCAS Version 7.0 does not comply with this interpretations.

Table 16.6: BDS 3,0 MB bits 10 to 15 (MB:9=1)

MB	Value	
10	0	RA is preventive
	1	RA is corrective
11	0	Upward sense RA has been generated
	1	Downward sense RA has been generated
12	0	RA is not increased rate
	1	RA is increased rate
13	0	RA is not a sense reversal
	1	RA is a sense reversal
14	0	RA is not altitude crossing
	1	RA is altitude crossing
15	0	RA is vertical speed limit
	1	RA is positive

Table 16.7: BDS 3,0 MB bits 10 to 15 (MB:9=0 MB:28=1)

MB	Value	
10	0	RA does not require a correction in the upward sense
	1	RA requires a correction in the upward sense
11	0	RA does not require a positive climb
	1	RA requires a positive climb
12	0	RA does not require a correction in the downward sense
	1	RA requires a correction in the downward sense
13	0	RA does not require a positive descend
	1	RA requires a positive descend
14	0	RA does not require a crossing
	1	RA requires a crossing
15	0	RA is not a sense reversal
	1	RA is a sense reversal

2) The resolution advisory complements (RAC) record consists of four bits, and each bit has the following meaning:

```
MB:23=1   Do not pass below
MB:24=1   Do not pass above
MB:25=1   Do not turn left
MB:26=1   Do not turn right
```

3) An RA terminated bit indicates whether previously generated RA has been terminated.

4) The threat type indicator contains two bits with the following meaning:

```
00    No identity data in threat identity data
01    Threat identity data contains a Mode~S transponder address
10    Threat identity data contains altitude, range, and bearing
11    Not assigned
```

When the threat type indicator is `01`, MB bits 31–54 contain the 24-bit Mode S transponder address and the last two bits are set to zero.

When the threat type indicator is `10`, MB bits 31–56 is divided into three different segments.

- MB bits 31–43 contain the 13-bit threat altitude code. It is the same structure as the altitude code from section 13.2.
- MB bits 44–50 contain the most recent threat range from ACAS. When it is 0, no range estimation is available. When it is 127, the range is greater than 12.55 nautical miles. For other values, the range is calculated as $(n-1)/10 \pm 0.05$ nautical miles.
- MB bits 51–56 contain the most recent estimated bearing of the threat aircraft, relative to their own heading. When it is 0, no estimation is available. Values larger than 60 are not assigned. For other values (from 1 to 60), the bearing range is calculated as $[6(n-1), 6n]$ in degrees.

17 | Mode S enhanced surveillance

Mode S Enhanced Surveillance (EHS) provides a set of advanced functionalities for the Mode S transponders. Three different types of reports are included in EHS. They are designed to report vertical intent, turning performance, and airspeeds. This chapter explaines structures and decoding processes of these reports.

17.1 Selected vertical intention (BDS 4,0)

The selected vertical intention message is designed for air traffic control to obtain an aircraft's current vertical intentions. For example, an aircraft controller can use this information to check whether an aircraft is complying with an altitude command. Table 17.1 shows the structure of the message.

Table 17.1: Selected intention (BDS 4,0), MB field

FIELD	MSG	MB	BITS
Status (for MCP/FCU selected altitude)	33	1	1
MCP/FCU selected altitude Range: [0, 65520] ft LSB: 16 ft	34–45	2–13	12
Status (for FMS selected altitude)	46	14	1
FMS selected altitude Range: [0, 65520] ft LSB: 16 ft	47–58	15–26	12
Status (for barometric press setting)	59	27	1
Barometric pressure setting Note: actual value minus 800 mb Range: [0, 410] mb LSB: 0.1 mb	60–71	28–39	12
Reserved (all zeros)	72–79	40–47	8
Status of MCP/FCU mode	80	48	1
VNAV mode	81	49	1
Alt hold mode	82	50	1
Approach mode	83	51	1
Reserved (all zeros)	84–85	52–53	2
Status of target altitude source	86	54	1
Target altitude source	87–88	55–56	2

Two different types of selected altitude fields are included in the message. Values in the *MCP/FCU selected altitude* field are from the mode control panel or flight control unit. These are often inputs from the pilot. Values in *FMS selected altitude* field are derived from the flight management system controlling the vertical profile, which

often does not come from manual input. In practice, FMS selected altitude values are often empty or unusable. When it is presented, the value is mostly the same as MCP/FCU selected altitude.

The *barometric pressure setting* value is the actual pressure in millibars (mb) subtracted by a constant of 800. If the actual value is below 800 mb (or above 1209 mb), the corresponding status bit (MB:27) is set to `0`.

The last two bits in the MB field show the source of the target altitude. They have the following meaning:

```
00: Unknown source
01: Aircraft altitude
10: FCU/MCP selected altitude
11: FMS selected altitude
```

Figure 17.1 illustrates an example of how to decode a BDS 4,0 message.

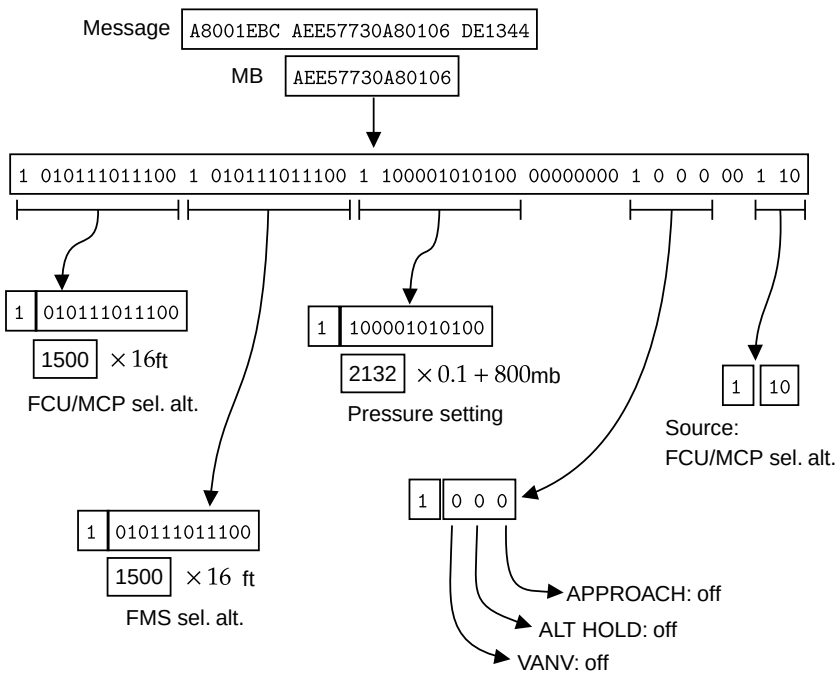


Figure 17.1: BDS 4,0 decoding example

Try it out

Using pyModeS, we can decode information of BDS 4,0 messages as:

```
import pyModeS as pms

msg = "A8001EBCAEE57730A80106DE1344"

pms.commb.selalt40fms(msg)
# 24000, FMS selected altitude (ft)

pms.commb.selalt40mcp(msg)
# 24000, MCP selected altitude (ft)

pms.commb.p40baro(msg)
# 1013.2, pressure (mb)
```

17.2 Track and turn report (BDS 5,0)

The track and turn report is designed to provide parameters to describe aircraft turns. In this type of message, roll angle, track angle, and track rate are provided. It also includes the ground speed and true airspeed (TAS) of the aircraft.

Table 17.2 shows the structure of the message.

Table 17.2: Track and turn report (BDS 5,0), MB field

FIELD	MSG	MB	BITS
Status (for roll angle)	33	1	1
Sign	34	2	1
Roll angle Range: [-90, +90] degrees LSB: 45/256 degrees	35–43	3–11	9
Status (for track angle)	44	12	1
Sign	45	13	1
True track angle Range: [-180, 180] degrees LSB: 90/512 degrees	46–55	14–23	10
Status (for ground speed)	56	24	1
Ground speed Range: [0, 2046] kt LSB: 2 kt	57–66	25–34	10
Status (for track angle rate)	67	35	1
Sign	68	36	1
Track angle rate Range: [-16, 16] degrees/second LSB: 8/256 degrees/second	69–77	37–45	9
Status (for true airspeed)	78	46	1
True airspeed Range: [0, 2046] kt LSB: 2 kt	79–88	47–56	10

In this message, three signed values are shown, which are roll angle, track angle, and track angle rate. Two's complement coding (see section 11.4) should be used to calculate these values.

The following Figure 17.2 shows an example of decoding a BDS 5,0 message.

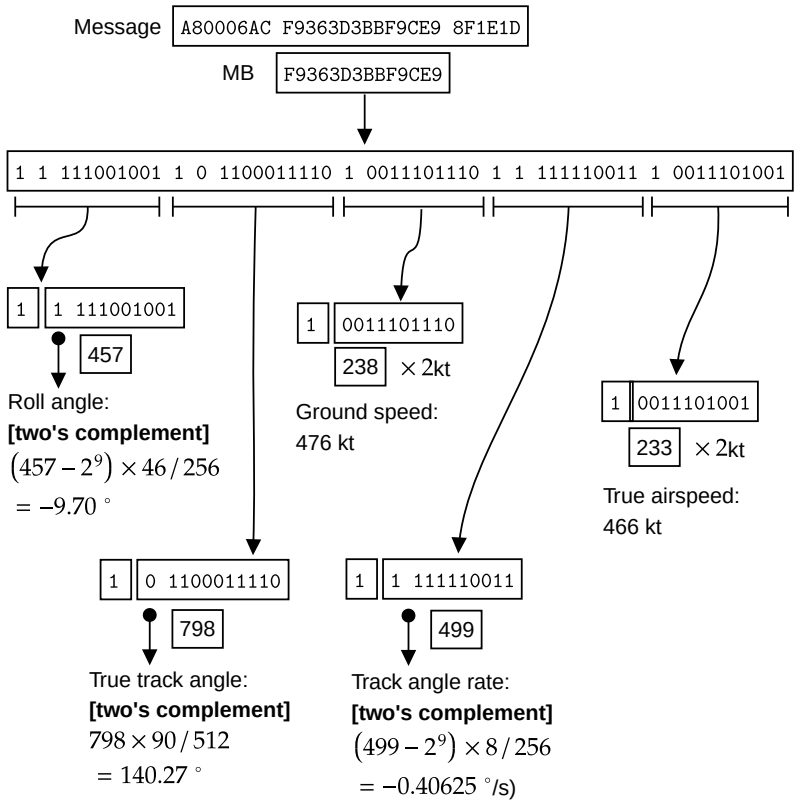


Figure 17.2: BDS 5,0 decoding example

Try it out

Using pyModeS, we can decode information of BDS 5,0 messages as:

```
import pyModeS as pms

msg = "A80006ACF9363D3BBF9CE98F1E1D"

pms.commb.roll50(msg)   # -9.7, roll angle (deg)
pms.commb.trk50(msg)    # 140.273, track angle (deg)
pms.commb.rtrk50(msg)   # -0.406, track angle rate (deg/s)
pms.commb.gs50(msg)     # 476, ground speed (kt)
pms.commb.tas50(msg)    # 466, TAS (kt)
```

17.3 Heading and speed report (BDS 6,0)

The heading and speed report is designed to downlink various airspeed and vertical rate to air traffic controllers. In this message, indicated airspeed (IAS), Mach number, barometric altitude rate, inertial vertical velocity, and the magnetic heading of the aircraft are provided. Table 17.3 shows the structure of the message.

Table 17.3: Heading and speed report (BDS 6,0), MB field

FIELD	MSG	MB	BITS
Status (for magnetic heading)	33	1	1
Sign	34	2	1
Magnetic heading Range: [-180, +180] degrees LSB: 90/512 degrees	35–44	3–12	10
Status (for indicated airspeed)	45	13	1
Indicated airspeed Range: [0, 1023] kt LSB: 1 kt	46–55	14–23	10
Status (for Mach number)	56	24	1
Mach number Range: [0, 4.092] LSB: 0.004	57–66	25–34	10
Status (for barometric altitude rate)	67	35	1
Sign	68	36	1
Barometric altitude rate Range: [-16384, +16352] ft/min LSB: 32 ft/min	69–77	37–45	9
Status (for inertial vertical velocity)	78	46	1
Sign	79	47	1
Inertial vertical velocity Range: [-16384, +16352] ft/min LSB: 32 ft/min	80–88	48–56	9

In this message, there are a few signed values, such as heading and two vertical rates. Two's complement coding (see section 11.4) should be used to calculate these values.

The magnetic heading is the aircraft's heading with respect to the magnetic North, which can be different from the true north (for example, used for the track angle from ADS-B and BDS 5,0). Often, an aircraft obtains the magnetic heading by adding its true North heading with the magnetic declination from a world magnetic model, such as [4]. It is worth noting that the true North heading is not necessarily the same as the track angle due to the influence of wind.

In the heading and speed report, two different kinds of vertical rates are reported. Barometric altitude rates are only derived from barometer measurements. Since

the source data from air data system is not filtered, significant noise is contained in these values. In contrast, inertial vertical velocities are values provided by navigational equipment from different sources including the flight management computer. According to [11], data sources with different levels of priorities are defined for these two values, which are listed in Table 17.4.

Table 17.4: Data sources for two vertical rates in heading and speed report

Parameter	**Input Data Source Priorities**
Barometric altitude rate	1. Air Data System 2. Inertial Reference System/Flight Management System
Inertial vertical velocity	1. Flight Management Computer / GNSS integrated 2. Flight Management Computer (General) 3. Inertial Reference System/Flight Management System

Figure 17.3 illustrates an example on how to decode a BDS 6,0 message.

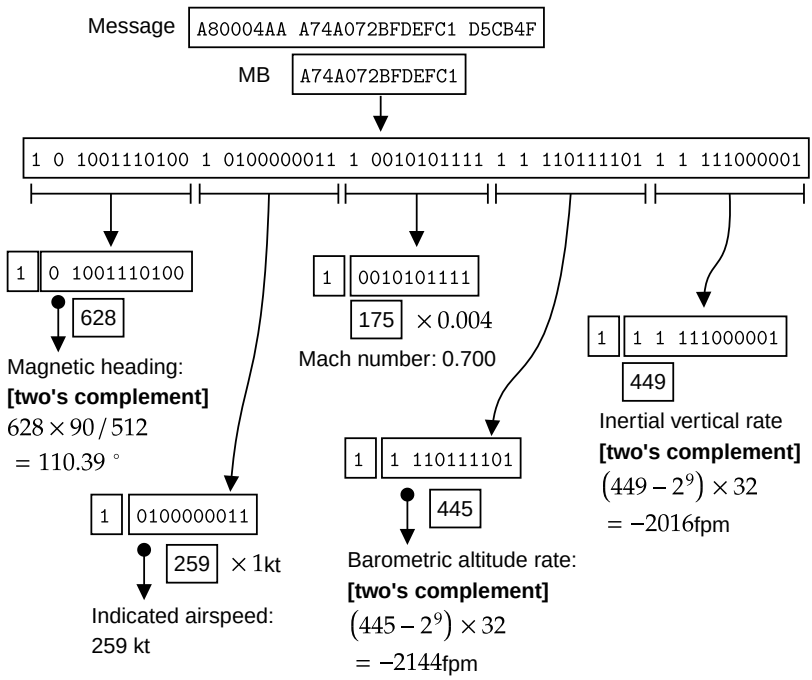


Figure 17.3: BDS 6,0 decoding example

Try it out

Using pyModeS, we can decode information of BDS 6,0 messages as:

```
import pyModeS as pms

msg = "A80004AAA74A072BFDEFC1D5CB4F"

pms.commb.hdg60(msg)      # 110.391, heading (deg)
pms.commb.ias60(msg)      # 259, ISA (kt)
pms.commb.mach60(msg)     # 0.7, Mach (-)
pms.commb.vr60baro(msg)   # -2144, baro vertical rate (ft/min)
pms.commb.vr60ins(msg)    # -2016, INS vertical rate (ft/min)
```

18 | Mode S meteorological services

In the current Mode S design, two message formats are used for aircraft to communicate meteorological conditions. These messages are meteorological routine air report (MRAR) and meteorological hazard report (MHR). In this chapter, we focus on explaining the information contained in these two types of messages.

18.1 Meteorological routine air report (BDS 4,4)

In MRAR messages, information on wind, air temperature, pressure, and humidity is transmitted. The structure of the message is shown in Table 18.1.

Table 18.1: Meteorological routine air report (BDS 4,4), MB field

FIELD	MSG	MB	BITS
Figure of merit / source	33–36	1–4	4
Status (for wind)	37	5	1
Wind speed Range: [0, 511] knots LSB: 1 knots	38–46	6–14	9
Wind direction Range: [0, 360] degrees LSB: 180/256 degrees	47–55	15–23	9
Sign (for temperature)	56	24	1
Static air temperature Range: [-128, +128] °C LSB: 0.25 °C	57–66	25–34	10
Status (for pressure)	67	35	1
Average static pressure Range: [0, 2048] hPa LSB: 1 hPa	68–78	36–46	11
Status (for turbulence)	79	47	1
Turbulence	80–81	48–49	2
Status (for humidity)	82	50	1
Humidity Range: [0%, 100%] LSB: 100/64 %	83–88	51–56	6

The first field of the message defined the figure of merit (FOM)/source of the information. The values indicate the following:

- 0: Invalid
- 1: Inertial system (INS)
- 2: Global Navigation Satellite System (GNSS)

- 3: Distance measuring equipment-based navigation (DME/DME)
- 4: Very High Frequency omnidirectional range/distance measuring equipment based navigation (VOR/DME)
- 5–15: Reserved

For static air temperature, the encoded value can be negative. Hence, two's complement coding (see section 11.4) is used for encoding the value. The actual maximum range of temperature is from -80 °C to +60 °C.

It is also worth pointing out a discrepancy in the design. The temperature is encoded using 10 bits, where the least significant bit value should have been 0.125°. However, according to the official document [13], the LSB value is 0.25°. ICAO is planning to update this to 0.125° in the future. However, most current implementations still use 0.25°.

Figure 18.1 shows the decoding of an MRAR example message.

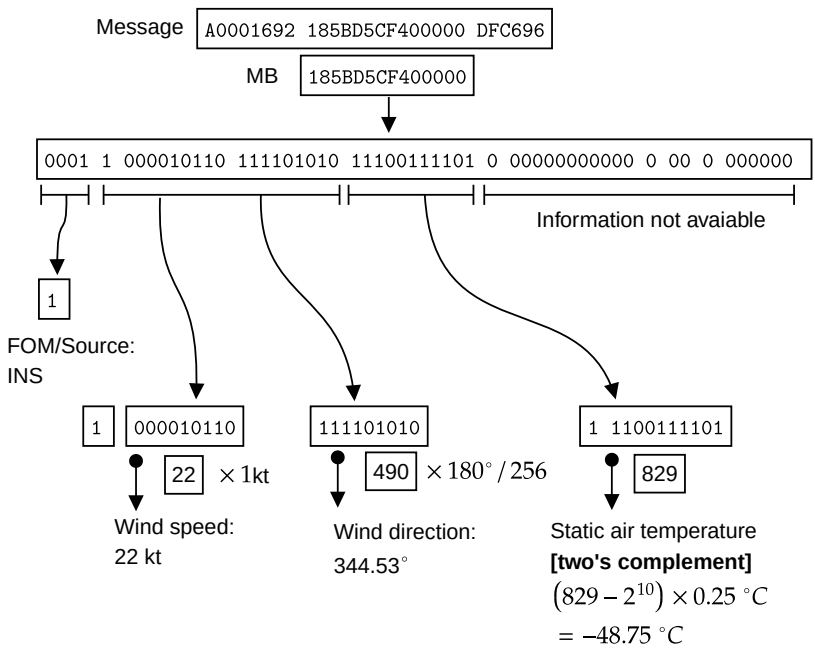


Figure 18.1: Meteorological routine air report (BDS 4,4) decoding example

Try it out

Using pyModeS, we can decode information of BDS 4,4 messages as:

```
import pyModeS as pms

msg = "A0001692185BD5CF400000DFC696"

pms.commb.wind44(msg)   # (22, 344.5)
pms.commb.temp44(msg)   # (-48.75, -24.375)
pms.commb.p44(msg)      # None
pms.commb.hum44(msg)    # None
```

18.2 Meteorological hazard report (BDS 4,5)

In MHR messages, different hazard condition levels are reported, such as turbulence, wind shear, microburst, icing, and wake vortex. It also includes temperature, pressure, and radio height. Table 18.2 shows the structure of the MHR message.

It is worth noting that during real flights, MHR messages are much rarer than MARA messages. Based on the tests I conducted, whenever MRAR messages are detected, most of the them only contain temperature information.

Table 18.2: Meteorological harzard report (BDS 4,5), MB field

FIELD	MSG	MB	BITS
Status (for turbulence)	33	1	1
Turbulence	34–35	2–3	2
Status (for wind shear)	36	4	1
Wind shear	37–38	5–6	2
Status (for microburst)	39	7	1
Microburst	40–41	8–9	2
Status (for icing)	42	10	1
Icing	43–44	11–12	2
Status (for wake vortex)	45	13	1
Wake vortex	46–47	14–15	2
Status (for temperature)	48	16	1
Sign (for temperature)	49	17	1
Static air temperature Range: [-128, +128] °C LSB: 0.25 °C	50–58	18–26	9
Status (for pressure)	59	27	1
Average static pressure Range: [0, 2048] hPa LSB: 1 hPa	60–70	28–38	11
Status (for height)	71	39	1
Radio height Range: [0, 65 528] ft LSB: 16 ft	72–83	40–51	12
Reserved	84–88	52–56	5

The levels for turbulence, wind shear, microburst, icing, and wake vortex are encoded as follows:

- `00` : NIL
- `01` : LIGHT
- `10` : MODERATE
- `11` : SEVERE

As is the case for MRAR messages, the actual range of the temperature is from -80

°C to +60 °C, and it is encoded using the two's complement coding.

Figure 18.2 shows the decoding of an example message. Note that in this example, only the air temperature information is included. None of the hazard conditions is reported.

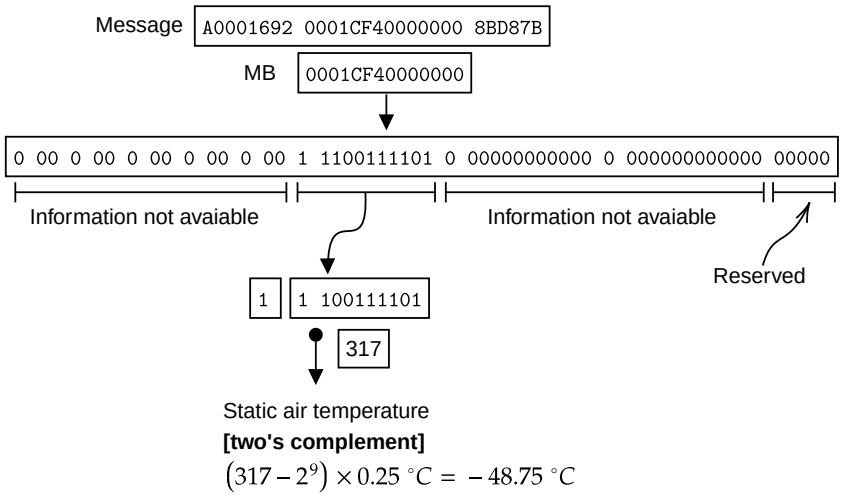


Figure 18.2: Meteorological harzard report (BDS 4,5) decoding example

Note

The availability of BDS 4,4 messages is quite low. Very few aircraft's transponders have these capabilities enabled. BDS 4,5 messages are even rarer. When such a message is transmitted, it is very common that the information in many fields is not available.

19 | Inferencing of BDS codes

In Chapter 15, we discussed the basic structure and protocols of Mode S Comm-B messages. Each message type is identified by an 8-bit Comm-B Data Selector (BDS) code, which is only transmitted in the uplink message but not included in the downlink message. As third parties observing the replies to Mode S surveillance interrogations, we must first determine the BDS code before decoding the content of any Comm-B messages.

19.1 BDS codes identification logics

Each Mode S message has a predefined structure and variables. For almost all common Comm-B message types, there are rules for certain bits. For example, these bits are:

- Reserved bits: These are bits in the different message types that are reserved for future use. They all have to be zeros. If any of the bits is not zero, the possibility of a certain BDS code should be ruled out.
- Status bits: Some fields in Comm-B messages have their corresponding status bits. When a status bit is set to zero, all bits in the field must be zero. If the field contains non-zero bits, the possibility of a certain BDS code can be ruled out.
- Value rules: Different fields in the messages also have different physical ranges. For example, the Mach number in BDS 6,0 should not be higher than 1, and the temperature value in BDS 4,4 should be between -80°C and 60°C. These constants can then be used to exclude certain BDS codes.

Tables 19.1, 19.2, and 19.3 show the rules for identifying different types of ELS, EHS, and meteorological messages.

Table 19.1: Heuristic identification logic for ELS

BDS	MB bits	Parameter	Rules
1,0	1–8	BDS code	Equal to `0001 0000`
	10–14	Reserved	All zeros
1,7	7	BDS 2,0 enabled	Equal to `1`
	29–56	Reserved	All zeros
2,0	1–8	BDS code	Equal to `0010 0000`
	9–56	Callsign	Only contains 0–9, A–Z, or space
3,0	1–8	BDS code	Equal to `0011 0000`
	29–30	Threat type	Not equal to `11`
	16–22	ACAS	less than 48

Table 19.2: Heuristic identification logic for EHS

BDS	MB bits	Parameter	Rules
4,0	1 : 2–13	MCP/FCU selected altitude	Status consistent
	14 : 15–26	FMS selected altitude	Status consistent
	27 : 28–39	Barometric pressure	Status consistent
	40 - 47	Reserved	All zeros
	52 - 53	Reserved	All zeros
5,0	1 : 2–11	Roll angle	Status consistent Between -50 and 50 degrees
	12 : 13–23	True track angle	Status consistent
	24 : 25–34	Ground speed	Status consistent Between 0 and 600 kt
	35 : 36–45	Track angle rate	Status consistent
	45 : 46–56	True airspeed	Status consistent Between 0 and 500 kt
6,0	1 : 2–12	Magnetic heading	Status consistent
	13 : 14–23	Indicated airspeed	Status consistent Between 0 and 500 kt
	24 : 25–34	Mach number	Status consistent Between 0 and 1
	35 : 36–45	Barometric vertical rate	Status consistent Between -6000 and 6000 fpm
	46 : 47–56	Inertial vertical rate	Status consistent Between -6000 and 6000 fpm

Table 19.3: Heuristic identification logic for MRAR and MHR

BDS	MB bits	Parameter	Rules
4,4	1–4	FOM	Less than 5
	5 : 6–23	Wind speed / direction	Status consistent speed less than 250 kt
	24–34	Static air temperature	Between -80 and 60°C
4,5	1 : 2–3	Turbulence	Status consistent
	4 : 5–6	Wind shear	Status consistent
	7 : 8–9	Microburst	Status consistent
	10 : 11–12	Icing	Status consistent
	13 : 14–15	Wake vortex	Status consistent
	16 : 17–26	Static air temperature	Status consistent Between -80 and 60°C
	27 : 28–28	Static pressure	Status consistent
	39 : 40–51	Radio height	Status consistent
	52–56	Reserved	All zeros

19.2 Identification of BSD 5,0 and 6,0

Of all the previously mentioned message types, BDS 5,0 and BDS 6,0 are the two message types that share the most similar structures. From Table 19.2, we can see the differences are in bit 12/13 and bit 45/46 of BDS 5,0 and BDS 6,0. This similarity can cause a number of messages to be identified as both BDS 5,0 and 6,0 messages.

In order to distinguish between these two messages, we need to utilize the information contained in the messages. BDS 5,0 and BDS 6,0 both contain aircraft speed information. We can design the additional check using the following logic:

- Assuming the message is BDS 5,0, compare the difference between the ground speed and true airspeed. The difference should not be too large. Empirically, this threshold can be set at approximately 200 kt to include possible wind speed.
- Assuming the message is BDS 6,0, convert the Mach number to calibrated airspeed based on altitude code under ISA condition [27]. The difference between calibrated airspeed and indicated airspeed should not be too large.

If the previous logic does not eliminate one of the two possibilities, we need to make use of the information collected in ADS-B (if available) to verify the information. ADS-B data from the same aircraft can be used as a reference, to check whether the speed from the assumed message type agrees with the ground speed from ADS-B. The details of this process is described in [26]. Wind information can also be taken into consideration to make the identification more accurate.

19.3 Decoding examples

The basic BDS code identification logic is relatively simple. It checks all criteria from all message types and decides whether all but one BDS code can be eliminated. In this example section, we are only going to show how a message with both possibilities of BDS 5,0 and BDS 6,0 can be identified.

Figure 19.1 shows the identification of a DF=20 message.

We see that BDS 6,0 is identified since the difference between the ground speed and airspeed is too high in the BDS 5,0 assumption.

Figure 19.2 shows the identification of a DF=21 message. In DF=21 messages, the altitude code is not included. Thus, we have to rely on other information, such as speed and track angle from ADS-B to validate the BDS code. The assumption about it being BDS 5,0 corroborates the speed information from ADS-B.

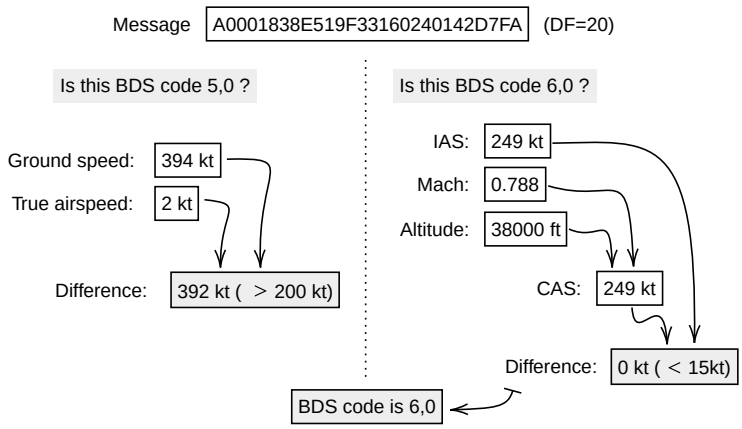


Figure 19.1: Identification of BDS code, DF=20

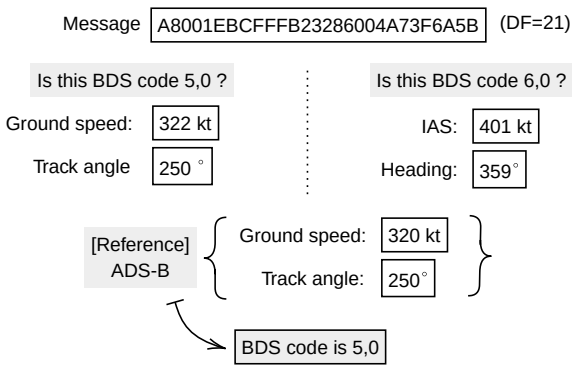


Figure 19.2: Identification of BDS code, DF=21

Try it out

Using pyModeS, we can infer the BDS code of a message as:

```
import pyModeS as pms

msg1 = "A0001838E519F33160240142D7FA"
bds1 = pms.bds.infer(msg1)

msg2 = "A8001EBCFFFB23286004A73F6A5B"
bds2 = pms.bds.infer(msg2)
```

The first BDS code should be BDS60. The second message has the possibility of being either BDS50 or BDS60, we can further infer the type based on ADS-B information:

```
bds2 = pms.bds.is50or60(msg2, 320, 250, 14000)
```

The final BDS code should be BDS50.

Part IV

Conclusions

20 | Summary and beyond

20.1 Summary

I started the book by introducing the most fundamental knowledge on modern radar technologies, establishing the technical background of Mode S and ADS-B signals, and describing the hardware and software for receiving these signals. By the end of Part 1, I demonstrated how to set up a basic system, including the antenna, software-defined radio, and software tools. One thing worth noting is that I only illustrated the process of setting up two open-source tools under a Linux operation system, but they can certainly be used under other different operating systems.

The main focus of the book (Parts 2 and 3) has been to explain the detailed structures of all ADS-B and common Mode S downlink messages, as well as to illustrate the decoding process with step-by-step examples when necessary. I tried to cover as many fields as possible in these messages. However, there may be certain fields that lack detailed explanation. The ICAO Annex Volume IV and ICAO Doc 9871 are always two excellent sources to find out more information on Mode S-related topics.

In Part 2, I first introduced the basics of ADS-B, such as message structure, types, and different ADS-B versions. Then, I explained the structures and decoding processes for all ADS-B messages, including identification and category, airborne position, surface position, airborne velocity, and aircraft operation status. I also talked about parameters like uncertainties, accuracies, and integrities of ADS-B positions and velocities. Finally, in Part 2, I described the error control mechanism of ADS-B messages, which can be used to detect corrupt messages and with potential to correct them.

In Part 3, I focused on the decoding of other types of Mode S messages. After explaining the basics of Mode S, I discussed all-call replies, surveillance replies, ACAS messages, and Comm-B messages. These are messages transmitted in different Mode S downlink formats, with Comm-B downlink containing the most information. In theory, it supports up to 256 message types (not all are used currently). Within Comm-B communication, some subsets of messages are commonly used by air traffic controllers. Then, I continued with the decoding of nine message types in three groups of Mode S services, which are Mode S elementary surveillance, Mode S enhanced surveillance, and Mode S meteorological services. As the third party receiver, the difficulty of decoding Comm-B messages is that the types cannot be easily identified like ADS-B messages. So, in the last chapter of this part, I discussed the important aspect of how to infer Comm-B types from downlink messages.

20.2 Crowd-sourced networks

In Chapter 2, I talked about the coverage of ADS-B receivers. Even with the perfect visibility, one receiver can only cover a maximum area of around 400 km radius, due to the curvature of the Earth. The coverage is further reduced when the antenna is not free of obstacles like buildings and mountains.

To cover full flight trajectories for a larger area, a group of receivers needs to be deployed. Furthermore, to have coverage over the entire continent (or multiple continents), an even larger network of receivers is needed. Easy access to low-cost software-defined radios make crowd-sourced ADS-B networks a reality nowadays.

A crowd-sourced network usually consists of a large group of enthusiasts who are voluntarily feeding their receiver data to a central server (or cluster of servers) on the internet owned by an entity. The owner of the service can be a commercial company or a non-profit organization. These companies often provide free premium user accounts to contributors in return for their data. Some companies also provide free receivers for users to host in areas with less coverage. In such cases, they often make agreements with receiver hosts to maintain and keep the receiver online as much as possible. Table 20.1 lists several ADS-B networks.

Table 20.1: A short list of ADS-B receiver networks

Name	Type	Founded	Number of receivers
FlightAware	Commercial	2005	30,000+*
FlightRadar24	Commercial	2006	25,000+*
RadarBox24	Commercial	2007	15,000+
OpenSky network	Non-profit/research	2013	3,500+*
ADS-B Exchange	Non-profit	2016	5,000+*

* Number of receivers obtained from each company's website, as of January 2021.

Another advantage of having a network of receivers is that it can be used to locate aircraft or validate their locations using multilateration [15], and this capability can be applied to any Mode A/C/S signals transmitted by aircraft. This technique has been used by many receiver networks. It is specifically useful to locate aircraft that are not equipped with ADS-B compliant transponders.

OpenSky network

Among these crowd-sourced networks, the OpenSky network [1] is one of the most well-known research networks in the air transportation research community. It is a crowd-sourced network that includes volunteers and researchers around the world. Currently, it has good coverage over Europe and a large part of North America. Figure 20.1 shows a recent current global coverage of the OpenSky network.

The OpenSky network provides users free and unlimited access to its data for non-

[1] https://opensky-network.org

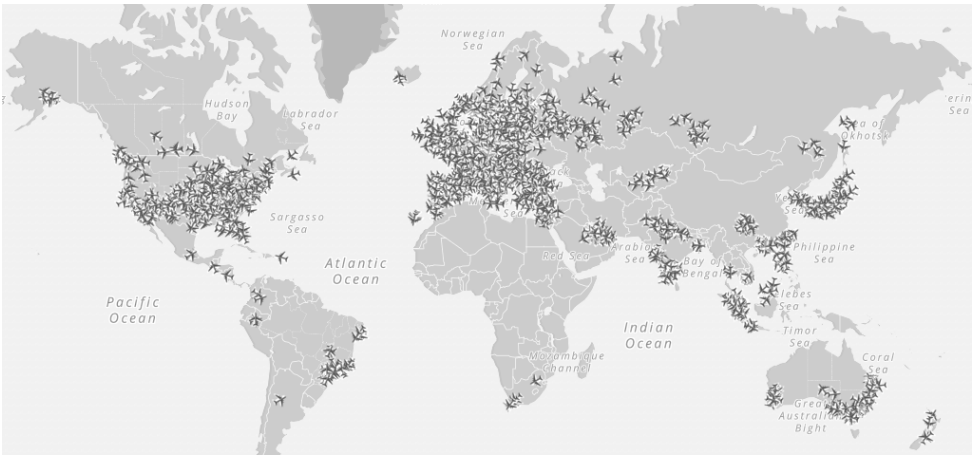


Figure 20.1: Coverage of OpenSky network, as of November 2020

commercial purposes [23]. It is possible to access its live ADS-B traffic data through a web API. Access to historical data is also possible through its Impala shell. However, the user must request this permission first. I also wrote a Python library,[2] which facilitates the access and decoding of raw Mode S messages from the OpenSky historical database using its Impala interface [25]. Another tool, *traffic*, can also be used to access the OpenSky historical database [19].

If you are setting up a receiver or have an existing receiver, I strongly encourage you to consider sharing your Mode S feed with the OpenSky network to increase coverage and to support open flight data for researchers. Guides on how to feed data to OpenSky can be found on its website.[3]

20.3 Additional data

For many air traffic-related studies, researchers frequently require information that is not transmitted in ADS-B and Mode S. One of the first things researchers often need is an aircraft database that can map the transponder address from ADS-B or Mode S to aircraft registration and aircraft type.

A few countries, including the United States, make this information publicly available for download. However, a large part of this information is not officially available or behind paywalls. There are many different sources of aircraft data that are maintained by hobbyists and volunteers on the internet. One of the most comprehensive databases for aircraft is also maintained by OpenSky network. A curated list of data sources can be found on the *Open Aviation Data* website.[4].

[2] https://github.com/junzis/pyopensky
[3] https://opensky-network.org/contribute/improve-coverage
[4] https://atmdata.github.io/sources/

On this website, a few other common data sources are listed too, including, for example, aircraft performance database, engine emission data, and weather data. All these data sources are available for public use.

20.4 Congestion

In the first chapter of this book, I discussed all types of signals that are transmitted over the 1090 MHz frequency. They include Mode A, Mode C, and Mode S. Due to the mix of different types of transponders from a wide range of aircraft including both commercial and general aviation, all these modes of surveillance are still in use nowadays. In addition to the increasing number of operational aircraft, the frequency congestion can be quite a serious issue during the daytime in busy airspace.

In a recent study [24], with the focus on Dutch airspace, a very high rate of corrupted ADS-B signals during the daytime was found. Figure 20.2 illustrates the severity of the frequency congestion during one day of radio frequency testing.

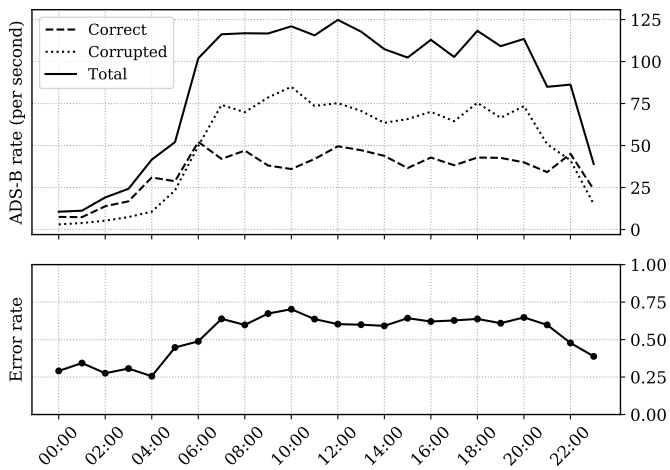


Figure 20.2: Corruption detected in ADS-B messages during a 24-hour period on 29 January, 2020 for Dutch airspace.

Even though this test is conducted in limit area, high congestion of the 1090 MHz radio frequency applies to other busy airspaces. The situation is especially serious in Europe, where the air traffic density is quite high during daytime operations. The mix of all different modes of surveillance (A/C/S) combined with the continuously increasing air traffic makes the channel busier.

20.5 The Future of Mode S and ADS-B

Reducing radio frequency congestion is one important area for future research. Some potential solutions to help us mitigate this problem in the future include, for

example:

- Reduce and phase out Mode A and Mode C surveillance;
- Balance Mode S interrogations among different surveillance radars;
- Migrate interrogation based surveillance to broadcast (like ADS-B);
- Introduce more error resistance modulation and channel coding methods;
- Make use of alternative frequencies.

Another issue is the global coverage of ADS-B. Currently, it is still difficult to fully track aircraft over the oceans, due to the lack of ground stations. One trend is to make use of satellites for receiving and relaying ADS-B messages transmitted by planes to ground stations [18]. Companies like Aireon and Spire have launched satellites that can receive ADS-B signals and make the data available commercially.

It is still challenging for satellites to handle signals with a low signal to noise ratio given their high orbits, as well as the high signal garbling rates caused by their large coverage. Furthermore, a large quantity of oceanic ADS-B data is located behind paywalls, which makes it hard for researchers to take advantage of space-based ADS-B data.

To this end, how to help researchers access higher-quality open surveillance data over a larger geographic scope, is a question that remains for me, you, and the entire research community to answer.

References

[1] Blythe, W., Anderson, H., and King, N. *Ads-b implementation and operations guidance document*. International Civil Aviation Organization, 2011.

[2] Carlitz, L. "The arithmetic of polynomials in a Galois field". In: *American Journal of Mathematics* 54.1 (1932), pp. 39–50.

[3] Chomik, G. "The future of collision avoidance–ACAS X". In: *International Journal of Engineering Trends and Technology* 39.5 (2016), pp. 284–287.

[4] Chulliat, A., Macmillan, S., Alken, P., Beggan, C., Nair, M., Hamilton, B., Woods, A., Ridley, V., Maus, S., and Thomson, A. "The US/UK world magnetic model for 2015-2020". In: (2015).

[5] Doran, R. W. "The Gray Code". In: *Journal of Universal Computer Science* 13.11 (2007), pp. 1573–1597.

[6] Gertz, J. L. *Fundamentals of mode s parity coding*. Tech. rep. Massachusetts Institute of Technology, Lincoln Laboratory, 1984.

[7] Grami, A. *Introduction to Digital Communications*. Academic Press, 2015. Chap. 10: Error-Control Coding. DOI: 10.1016/B978-0-12-407682-2.00010-7.

[8] Grappel, R. D., Harris, G. S., Kozar, M. J., and Wiken, R. T. "Elementary Surveillance (ELS) and Enhanced Surveillance (EHS) Validation via Mode S Secondary Radar Surveillance". In: *Project Report ATC-337, Lincoln Lab., MIT* (2008).

[9] Gray, D. E. and Barrett, C. W. *Privacy Impact Assessment (PIA) - Federal Aviation Administration (FAA) Privacy ICAO Address System*. U.S. Department of Transportation, 2019.

[10] ICAO. *Annex 10 to the Convention on International Civil Aviation, Aeronautical Telecommunications*. International Civil Aviation Organization, 2002.

[11] ICAO. *Manual on Mode S Specific Services, 2nd Edition*. Tech. rep. Doc 9688, AN/952. 2004.

[12] ICAO. *Secondary Surveillance Radar Mode S Advisory Circular*. International Civil Aviation Organization, 1983.

[13] ICAO. *Technical Provisions for Mode S Services and Extended Squitter*. International Civil Aviation Organization, 2008.

[14] Kasami, T. and Matoba, S. "Some efficient shortened cyclic codes for burst-error correction (Corresp.)" In: *IEEE Transactions on Information Theory* 10.3 (1964), pp. 252–253. DOI: 10.1109/TIT.1964.1053665.

[15] Kaune, R., Steffes, C., Rau, S., Konle, W., and Pagel, J. "Wide area multilateration using ADS-B transponder signals". In: *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE. 2012, pp. 727–734.

[16] Mandel, T. and Mache, J. "Investigating CRC Polynomials that Correct Burst Errors." In: *ICWN*. 2009, pp. 632–637.

[17] Mazda, F. *Telecommunications engineer's reference book*. Butterworth-Heinemann, 2014.

[18] Noschese, P., Porfili, S., and Di Girolamo, S. "Ads-b via iridium next satellites". In: *Digital Communications-Enhanced Surveillance of Aircraft and Vehicles (TIWDC/ESAV), 2011 Tyrrhenian International Workshop on*. IEEE. 2011, pp. 213–218.

[19] Olive, X. "traffic, a toolbox for processing and analysing air traffic data". In: *Journal of Open Source Software* 4.39 (2019), p. 1518. DOI: 10.21105/joss.01518.

[20] Orlando, V. A. “The mode S beacon radar system”. In: *The Lincoln Laboratory Journal* 2.3 (1989), pp. 345–362.

[21] RTCA. “Minimum operational performance standards for 1090MHz extended squitter automatic dependent surveillance-broadcast (ADS-B) and traffic information services-broadcast (TIS-B)[J]”. In: *RTCA DO-260B* 1.1 (2011), pp. 1365–1372.

[22] RTCA. “Minimum Operational Performance Standards for Universal Access Transceiver (UAT) automatic dependent surveillance–broadcast”. In: *RTCA DO-282A* (2002).

[23] Schäfer, M., Strohmeier, M., Lenders, V., Martinovic, I., and Wilhelm, M. “Bringing up OpenSky: A large-scale ADS-B sensor network for research”. In: *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IEEE. 2014, pp. 83–94.

[24] Sun, J. and Hoekstra, J. “Analyzing Aircraft Surveillance Signal Quality at the 1090 Megahertz Radio Frequency”. In: *Proceedings of the 9th International Conference for Research in Air Transportation*. 2020.

[25] Sun, J. and Hoekstra, J. “Integrating pyModeS and OpenSky Historical Database”. In: *Proceedings of the 7th OpenSky Workshop 2019*. Vol. 67. 2019. DOI: 10.29007/mmsb.

[26] Sun, J., Vû, H., Ellerbroek, J., and Hoekstra, J. M. “pymodes: Decoding Mode S Surveillance Data for Open Air Transportation Research”. In: *IEEE Transactions on Intelligent Transportation Systems* (2019). DOI: 10.1109/TITS.2019.2914770.

[27] Young, T. M. *Performance of the Jet Transport Airplane: Analysis Methods, Flight Operations, and Regulations*. John Wiley & Sons, 2017.

Acknowledgements

I could not have completed this book without the support of my wife, Marie. She proofread every word in this book and gave valuable input on its content. I want to thank my parents for their encouragement of my curiosity and scientific endeavors since childhood. My sons, William and Vincent, make my life and research more joyful, especially when I see they are curious and enthusiastic to see me testing antennas, receivers, and hardware.

Navigating through the mountain of information in different ICAO documents is complicated. I am grateful to have met Huy Vû and supervised his master thesis project. He was able to find almost any needed information regarding Mode S, which was a great help for this book and the pyModeS library. He is now a data analyst at *LVNL*, the Dutch air traffic control, and we still work together on interesting research topics.

In the summer of 2017, I received an email from a researcher from *ONERA*, the French aerospace research institute, who wanted to meet me for a coffee to discuss Mode S. That coffee discussion with Xavier Olive has turned into a fruitful collaboration. I want to say thanks to Xavier for his input on Mode S, ADS-B, and all other related and unrelated discussions.

Before the notion of this book ever existed, I was a new PhD student at TU Delft's CNS/ATM research group. I would like to thank Jacco Hoekstra and Joost Ellerbroek, who were my promotors and provided me with great support for all my PhD research topics. I am also extremely happy to have been able to continue working with them as a colleague since 2019. I would also like to extend my thanks to the Dean of our faculty, Henri Werij, for his consistent support and for offering me the opportunity to continue my research at the faculty.

My research often relies on Mode S and ADS-B data from regions beyond the coverage of our antenna in Delft. This is only possible with the help of crowd-sourced networks. I want to thank Martin Strohmeier from *OpenSky network*, Sean Atkinson from *FlightRadar24*, and James Stanford from *ADS-B Exchange* for sharing their data with me for different research projects over the past years.

The manual was an open-access book from the beginning. I heartily appreciate all the GitHub contributors who made suggestions and edits for this book and pyModeS library.[5] Since I decided to publish this book with TU Delft OPEN Publishing, our publishing officer, Frédérique Belliard, has been extremely helpful in making this book a reality. Finally, I would also like to thank Petr Jonas, Xavier Olive, and Enrico Spinielli for their comprehensive peer reviews that led to the successful completion of this book.

[5] The most up-to-date list of contributors can be found at:
https://github.com/junzis/the-1090mhz-riddle/graphs/contributors
https://github.com/junzis/pyModeS/graphs/contributors

About the Author

Junzi Sun is an assistant professor at TU Delft, currently working in the CNS/ATM group of the Aerospace Engineering Faculty. He is passionate about open-source, open science, and making air transportation more sustainable.

He was born in China and completed his bachelor's degree in telecommunication at Beijing University of Posts and Telecommunications. Then, he moved to Europe and obtained his master's degree in aerospace engineering at Polytechnic University of Catalonia. After that, he worked in Spain and France for a few years before continuing his doctoral research in the Netherlands, where he obtained his Ph.D. degree and currently works and lives.