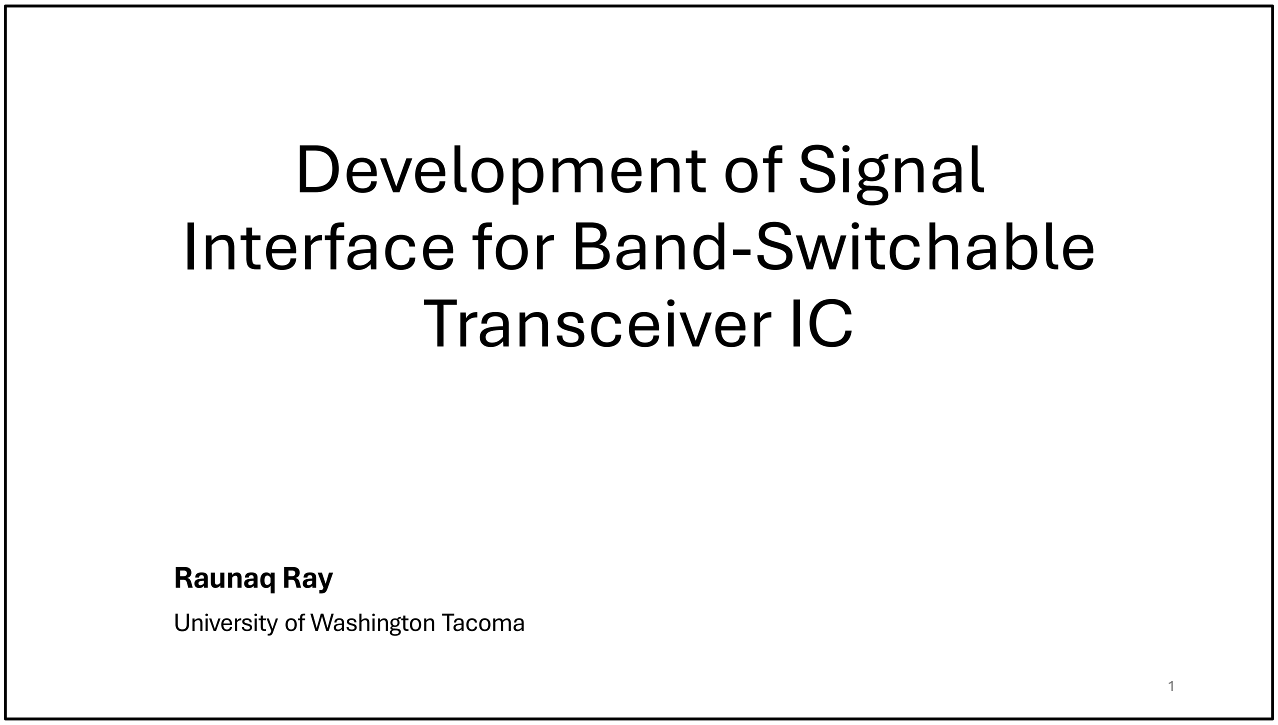


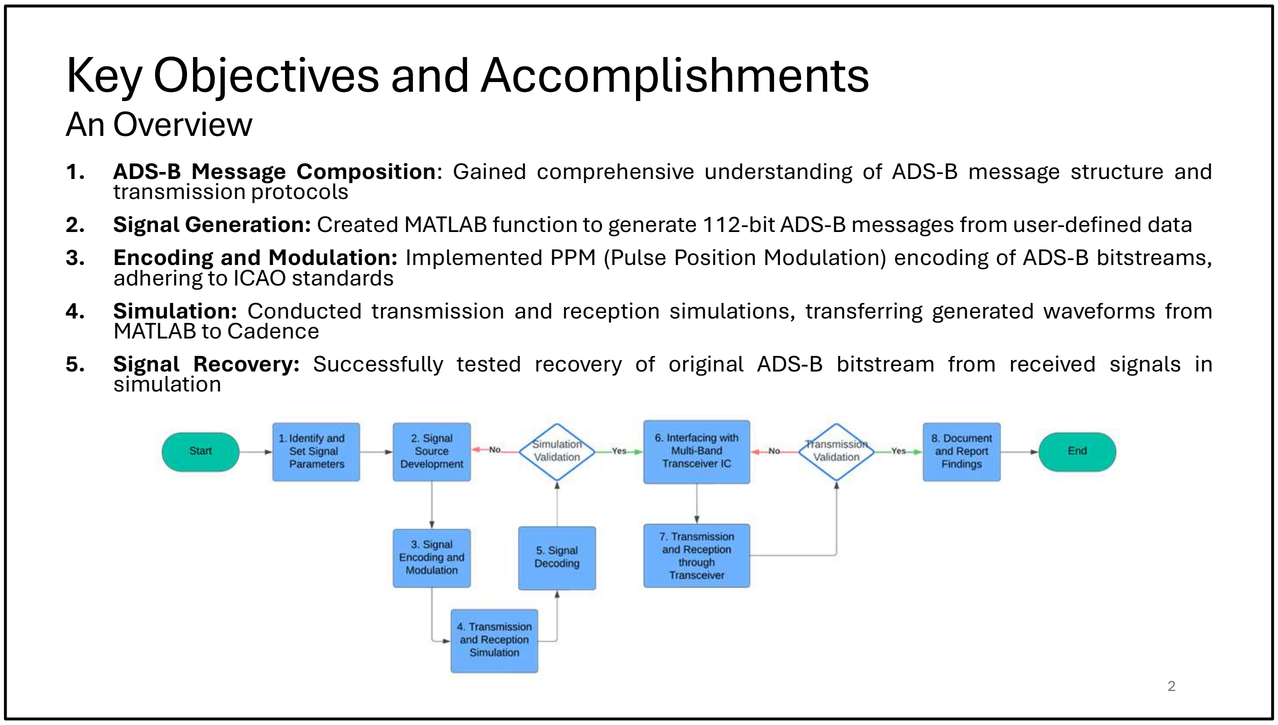
Development of Signal Interface for Band-Switchable Transceiver IC

Raunaq Ray

University of Washington Tacoma

1

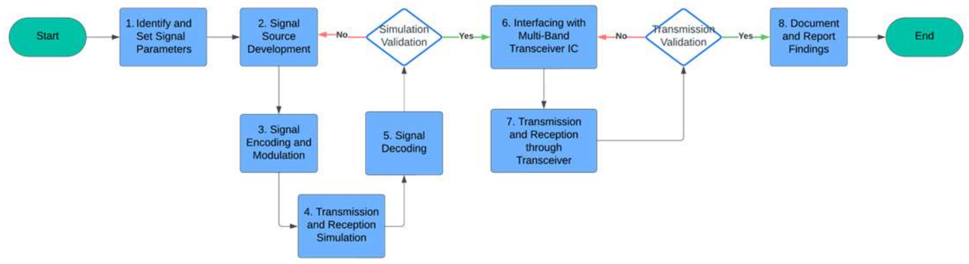
- Hello everyone, I'm here to present my summer project, which focuses on developing a signal interface for the band-switchable transceiver IC currently under development by Babak and Professor Dawn.
- Our primary objective has been to generate sample ADS-B messages from user-defined data and utilize these to evaluate the transceiver IC's performance.



Key Objectives and Accomplishments

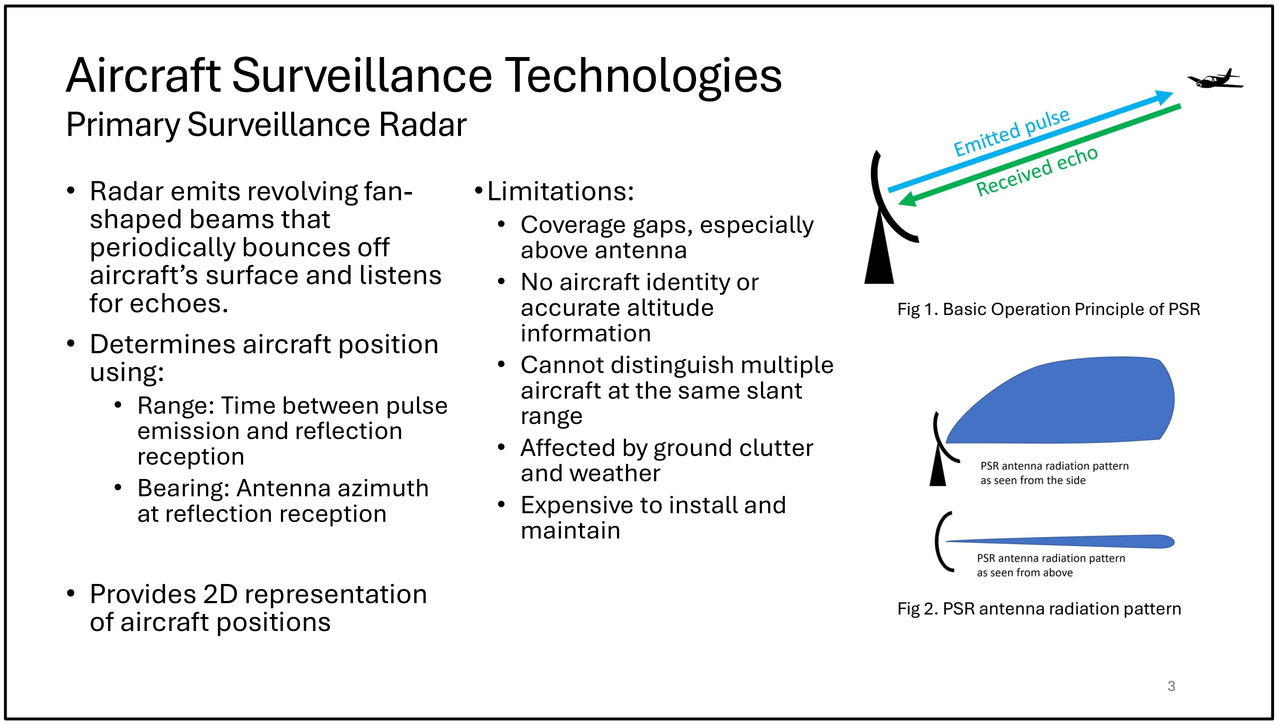
An Overview

1. **ADS-B Message Composition**: Gained comprehensive understanding of ADS-B message structure and transmission protocols
2. **Signal Generation:** Created MATLAB function to generate 112-bit ADS-B messages from user-defined data
3. **Encoding and Modulation:** Implemented PPM (Pulse Position Modulation) encoding of ADS-B bitstreams, adhering to ICAO standards
4. **Simulation:** Conducted transmission and reception simulations, transferring generated waveforms from MATLAB to Cadence
5. **Signal Recovery:** Successfully tested recovery of original ADS-B bitstream from received signals in simulation


Start
1. Identify and Set Signal Parameters
2. Signal Source Development
No
Simulation Validation
Yes
3. Signal Encoding and Modulation
4. Transmission and Reception Simulation
5. Signal Decoding
6. Interfacing with Multi-Band Transceiver IC
7. Transmission and Reception through Transceiver
No
Transmission Validation
Yes
8. Document and Report Findings
End

2

- This project has provided me with a comprehensive understanding of the ADS-B message structure, which will help to expand the project to similar protocols like RemoteID.
- The MATLAB functions I developed allow for flexibility and customization, enabling users to simulate various scenarios.
- As shown in the flowchart, I've successfully implemented stages 1 through 5 this summer. Our next steps involve interfacing these generated signals with the transceiver IC to test its performance under various conditions.



Aircraft Surveillance Technologies

Primary Surveillance Radar

- Radar emits revolving fan-shaped beams that periodically bounces off aircraft's surface and listens for echoes.
- Determines aircraft position using:
  - Range: Time between pulse emission and reflection reception
  - Bearing: Antenna azimuth at reflection reception
- Provides 2D representation of aircraft positions

- Limitations:
  - Coverage gaps, especially above antenna
  - No aircraft identity or accurate altitude information
  - Cannot distinguish multiple aircraft at the same slant range
  - Affected by ground clutter and weather
  - Expensive to install and maintain

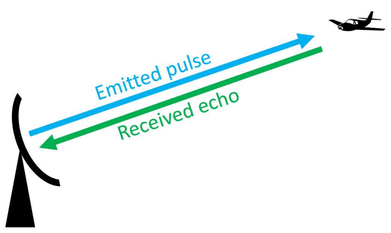
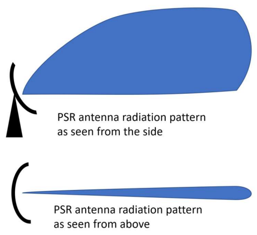
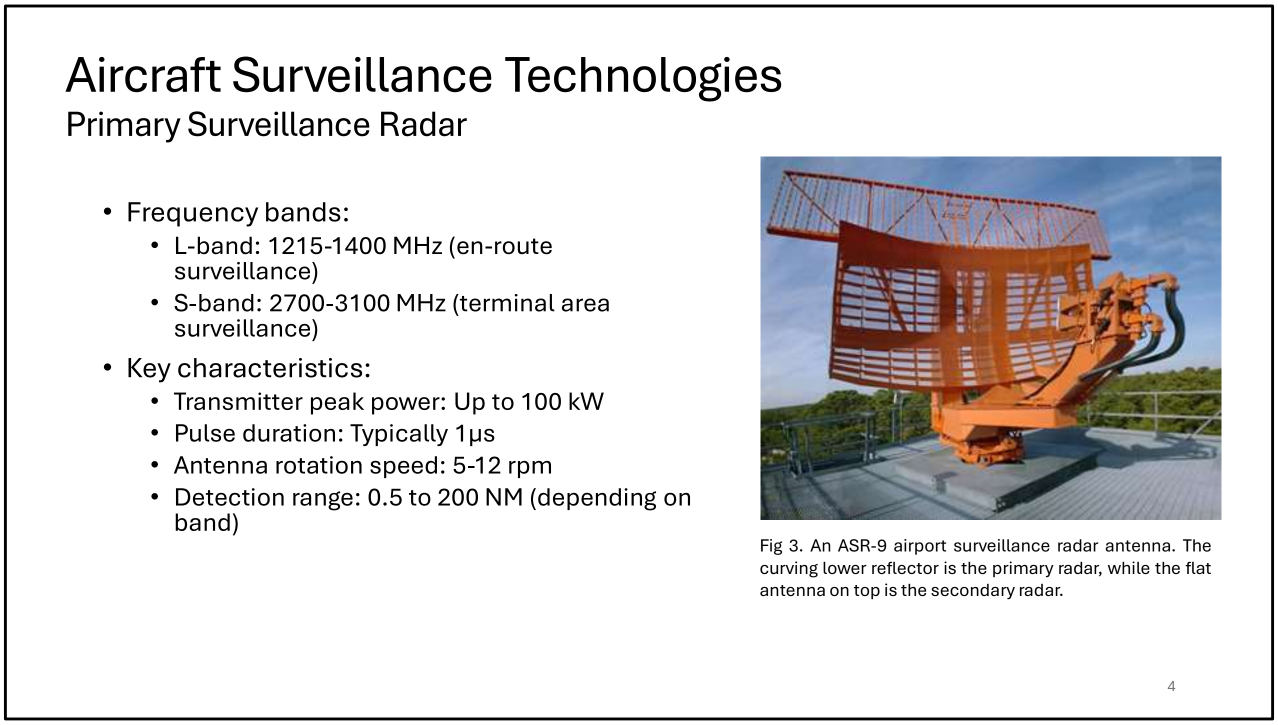


Fig 1. Basic Operation Principle of PSR


PSR antenna radiation pattern as seen from the side
PSR antenna radiation pattern as seen from above
Fig 2. PSR antenna radiation pattern

3

- I will quickly talk about the Surveillance Technologies I studied to understand the development of ADS-B and similar broadcast messages.
- Primary Surveillance Radar is quite fundamental in air traffic control. It operates by sending out rotating beams that help detect aircraft by listening for their echoes.
- It determines an aircraft's position through Range and Bearing. This approach allows for effective tracking in a 2D space.
- It has notable limitations, such as coverage gaps above the antenna and the struggles to distinguish between multiple aircraft at the same slant range.
- Despite its limitations, PSR is invaluable for detecting non-cooperative targets, such as birds.



Aircraft Surveillance Technologies

Primary Surveillance Radar

- Frequency bands:
  - L-band: 1215-1400 MHz (en-route surveillance)
  - S-band: 2700-3100 MHz (terminal area surveillance)
- Key characteristics:
  - Transmitter peak power: Up to 100 kW
  - Pulse duration: Typically 1μs
  - Antenna rotation speed: 5-12 rpm
  - Detection range: 0.5 to 200 NM (depending on band)

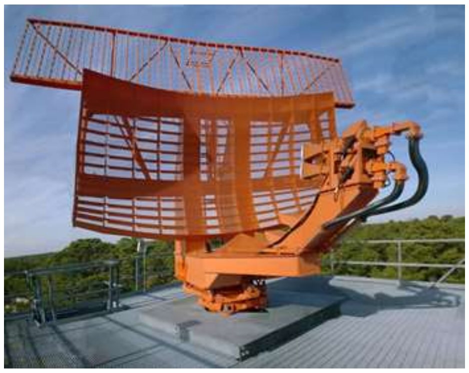
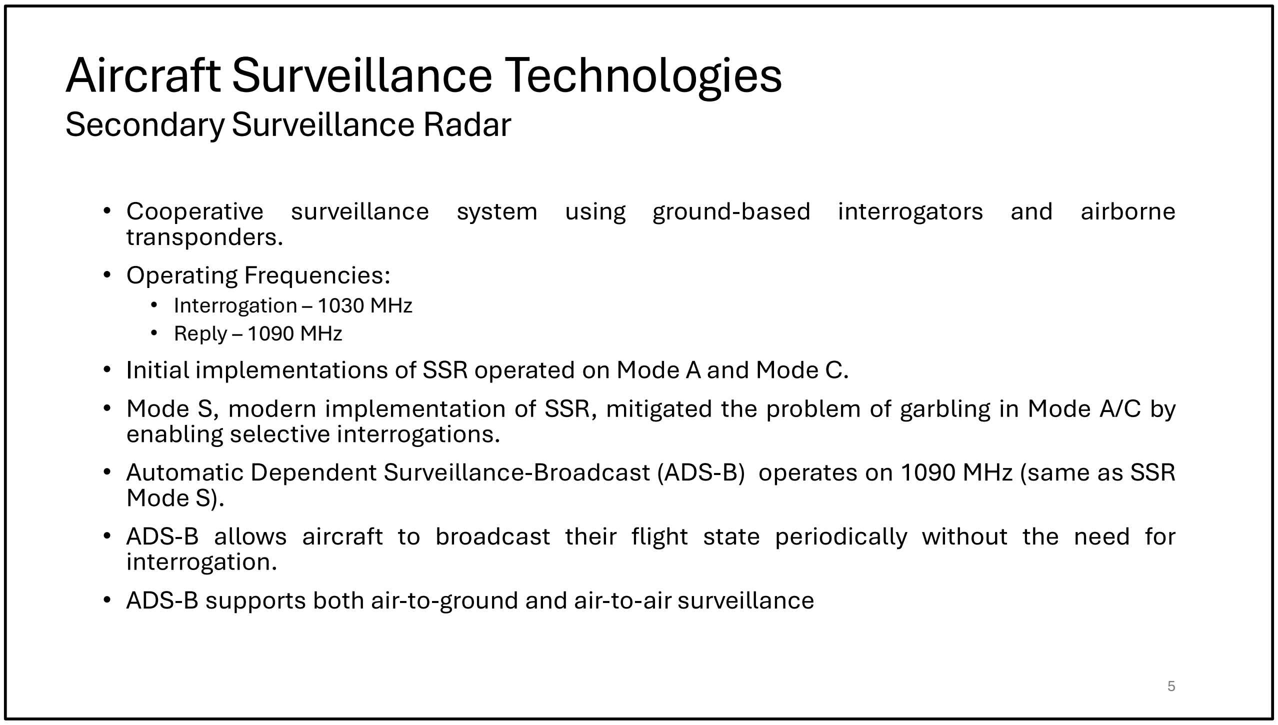


Fig 3. An ASR-9 airport surveillance radar antenna. The curving lower reflector is the primary radar, while the flat antenna on top is the secondary radar.

- These are some technical details of the Primary Surveillance Radar.
- Although these details are not directly relevant to my project, it helped me understand the overall picture of Aircraft Surveillance.



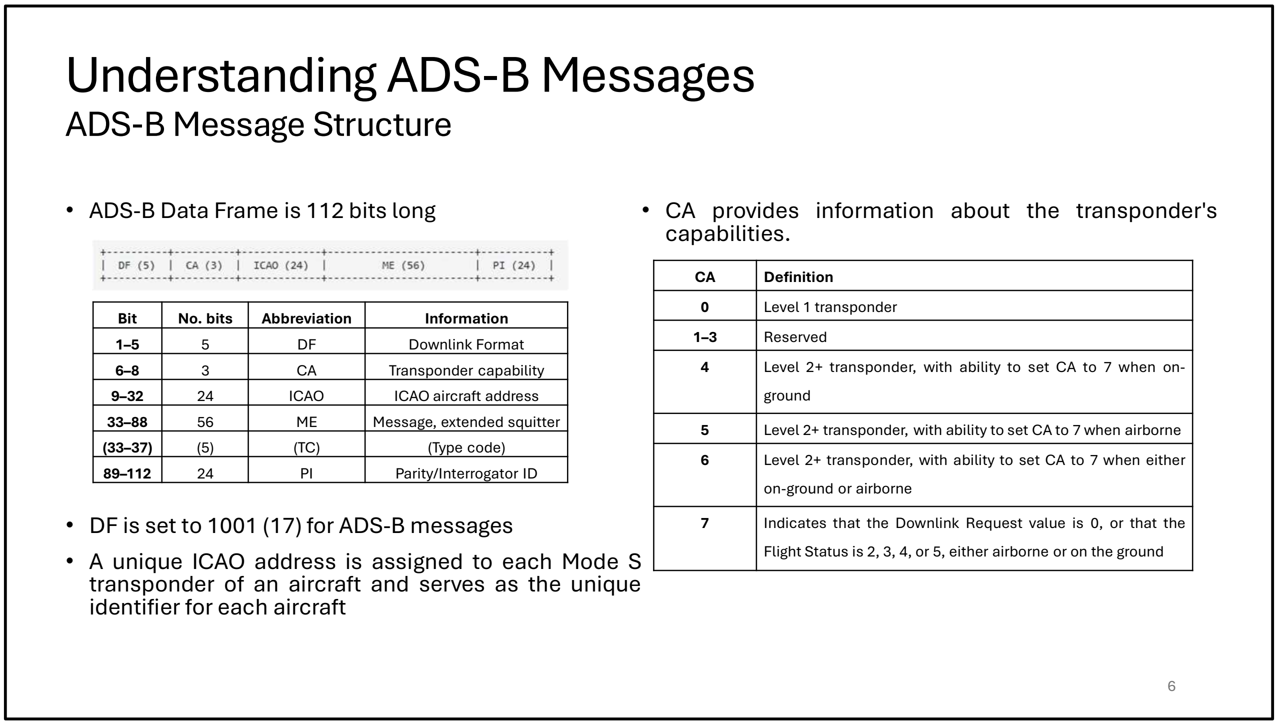
Aircraft Surveillance Technologies

Secondary Surveillance Radar

- Cooperative surveillance system using ground-based interrogators and airborne transponders.
- Operating Frequencies:
  - Interrogation – 1030 MHz
  - Reply – 1090 MHz
- Initial implementations of SSR operated on Mode A and Mode C.
- Mode S, modern implementation of SSR, mitigated the problem of garbling in Mode A/C by enabling selective interrogations.
- Automatic Dependent Surveillance-Broadcast (ADS-B) operates on 1090 MHz (same as SSR Mode S).
- ADS-B allows aircraft to broadcast their flight state periodically without the need for interrogation.
- ADS-B supports both air-to-ground and air-to-air surveillance

5

- Secondary Surveillance Radars rely on active cooperation between ground-based interrogators and airborne transponders.
- The initial implementations of SSR utilized Mode A and Mode C, which provided basic identification and altitude information. However, these modes had limitations, particularly in crowded airspace.
- Mode S mitigated issues like signal garbling that were common in Modes A and C by allowing for selective interrogations. This means ground stations can query specific aircraft without overwhelming the system with unnecessary data.
- ADS-B, operates on the same 1090 MHz frequency as Mode S, but takes a different approach by enabling aircraft to broadcast their flight state periodically without needing an interrogation from the ground.



Understanding ADS-B Messages

ADS-B Message Structure

- ADS-B Data Frame is 112 bits long

```
+----------+---------+-----------+-----------------------+----------+
| DF (5)   | CA (3)  | ICAO (24) |        ME (56)        | PI (24)  |
+----------+---------+-----------+-----------------------+----------+
```

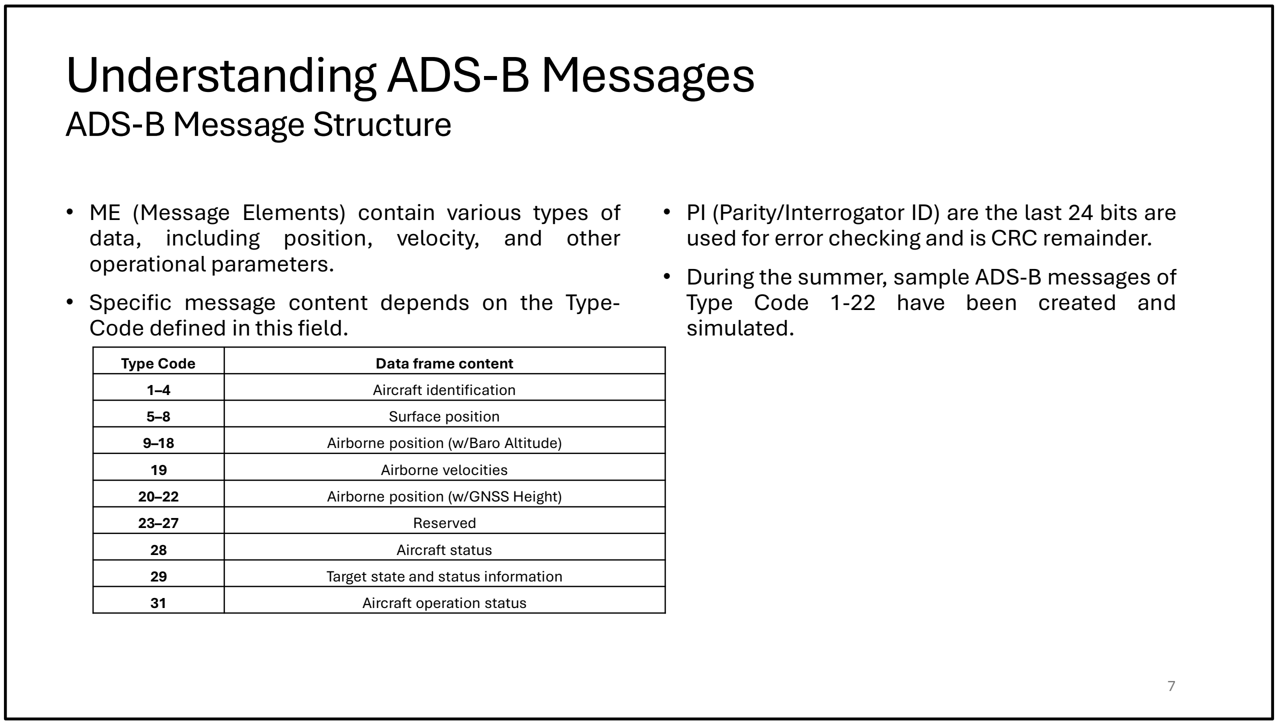
Bit	No. bits	Abbreviation	Information
1–5	5	DF	Downlink Format
6–8	3	CA	Transponder capability
9–32	24	ICAO	ICAO aircraft address
33–88	56	ME	Message, extended squitter
(33–37)	(5)	(TC)	(Type code)
89–112	24	PI	Parity/Interrogator ID

- DF is set to 1001 (17) for ADS-B messages
- A unique ICAO address is assigned to each Mode S transponder of an aircraft and serves as the unique identifier for each aircraft
- CA provides information about the transponder's capabilities.

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

6

- ADS-B messages have a structured format that efficiently packs crucial information into 112 bits.
- The first part of an ADS-B message is the Downlink Format. This acts like a label that identifies the type of message being sent. For ADS-B, it consistently indicates that the message is related to aircraft surveillance.
- ICAO is like a license plate number in the sky. This ensures that each aircraft can be accurately tracked and distinguished from others.
- The Capability field provides insights into what equipment the aircraft has onboard and its communication capabilities.



Understanding ADS-B Messages

ADS-B Message Structure

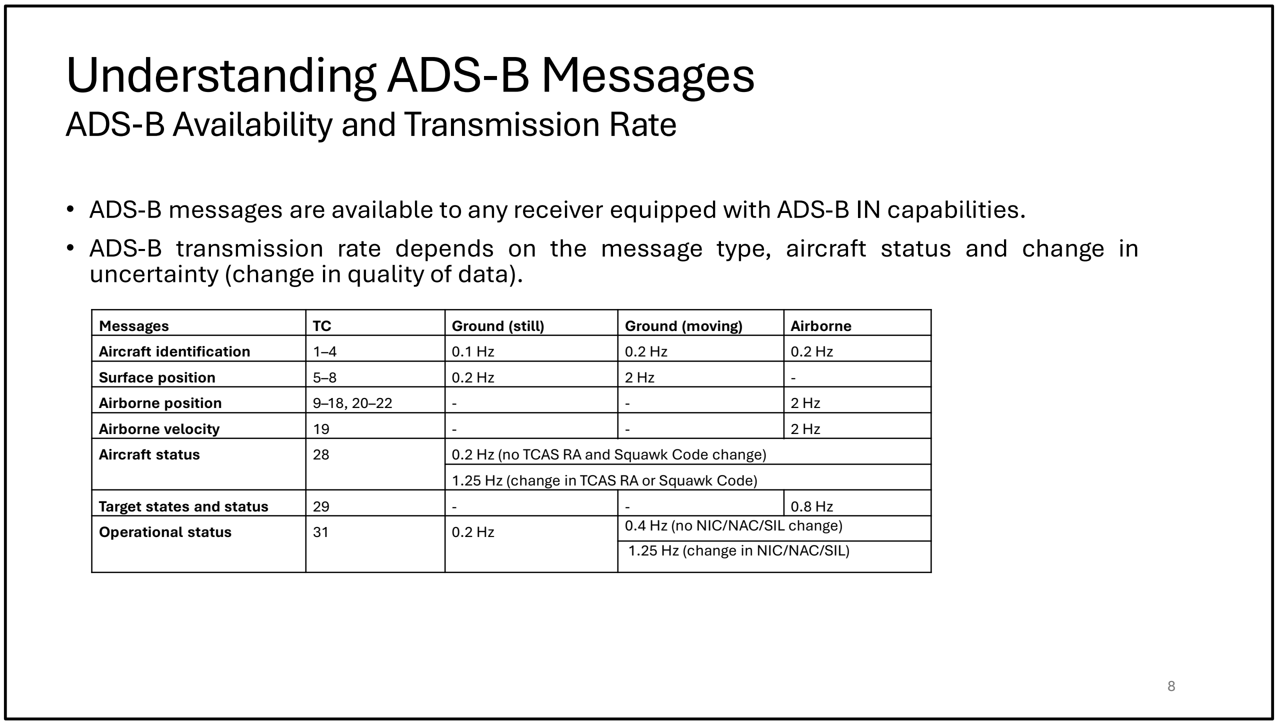
- ME (Message Elements) contain various types of data, including position, velocity, and other operational parameters.
- Specific message content depends on the Type-Code defined in this field.

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

- PI (Parity/Interrogator ID) are the last 24 bits are used for error checking and is CRC remainder.
- During the summer, sample ADS-B messages of Type Code 1-22 have been created and simulated.

7

- The message element field is a flexible field, allowing for different types of data to be transmitted based on the aircraft's current situation and capabilities.
- The first 5 bits of message element are dedicated to the Type Code, which acts as a key to interpret the rest of the message. Depending on this, the ME field can convey a wide range of data, as defined in the table.
- The final 24 bits of the ADS-B message are dedicated to the Parity field, which plays a crucial role in ensuring data integrity.
- In this project, I have developed MATLAB functions that can generate ADS-B messages from type code 1-22 with user-defined data.



Understanding ADS-B Messages

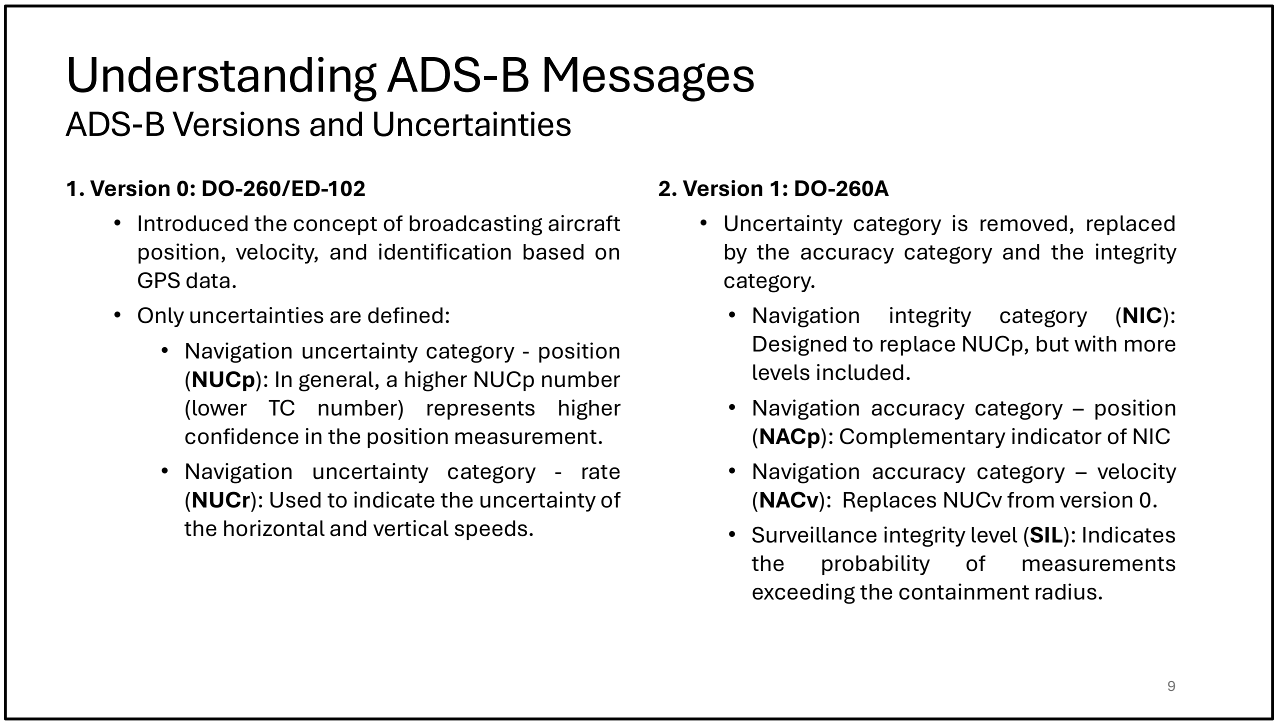
ADS-B Availability and Transmission Rate

- ADS-B messages are available to any receiver equipped with ADS-B IN capabilities.
- ADS-B transmission rate depends on the message type, aircraft status and change in uncertainty (change in quality of data).

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)	

8

- ADS-B is designed to be an open system, meaning its messages are accessible to any receiver with ADS-B IN capabilities.
- ADS-B transmission is not a one-size-fits-all system; instead, it adapts based on several pre-defined factors.
- For example, position updates might be sent more frequently than aircraft identification information.



Understanding ADS-B Messages

ADS-B Versions and Uncertainties

1. Version 0: DO-260/ED-102

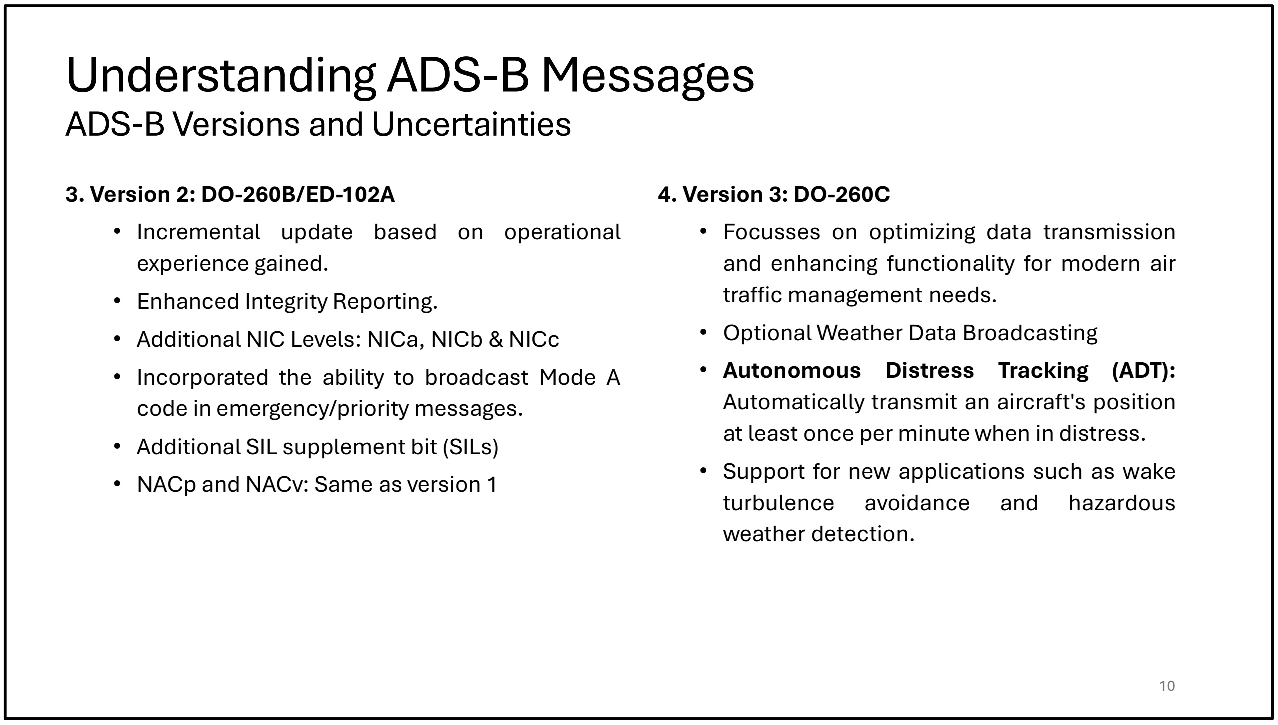
- Introduced the concept of broadcasting aircraft position, velocity, and identification based on GPS data.
- Only uncertainties are defined:
  - Navigation uncertainty category - position (**NUCp**): In general, a higher NUCp number (lower TC number) represents higher confidence in the position measurement.
  - Navigation uncertainty category - rate (**NUCr**): Used to indicate the uncertainty of the horizontal and vertical speeds.

2. Version 1: DO-260A

- Uncertainty category is removed, replaced by the accuracy category and the integrity category.
  - Navigation integrity category (**NIC**): Designed to replace NUCp, but with more levels included.
  - Navigation accuracy category – position (**NACp**): Complementary indicator of NIC
  - Navigation accuracy category – velocity (**NACv**): Replaces NUCv from version 0.
  - Surveillance integrity level (**SIL**): Indicates the probability of measurements exceeding the containment radius.

9

- The first version introduced the basic concept of broadcasting aircraft position, velocity, and identification using GPS data. It established initial measures of uncertainty in position and speed.
- Each subsequent version has enhanced the system's ability to provide more precise position data, better integrity checks, and additional features for safety and efficiency.



Understanding ADS-B Messages

ADS-B Versions and Uncertainties

3. Version 2: DO-260B/ED-102A

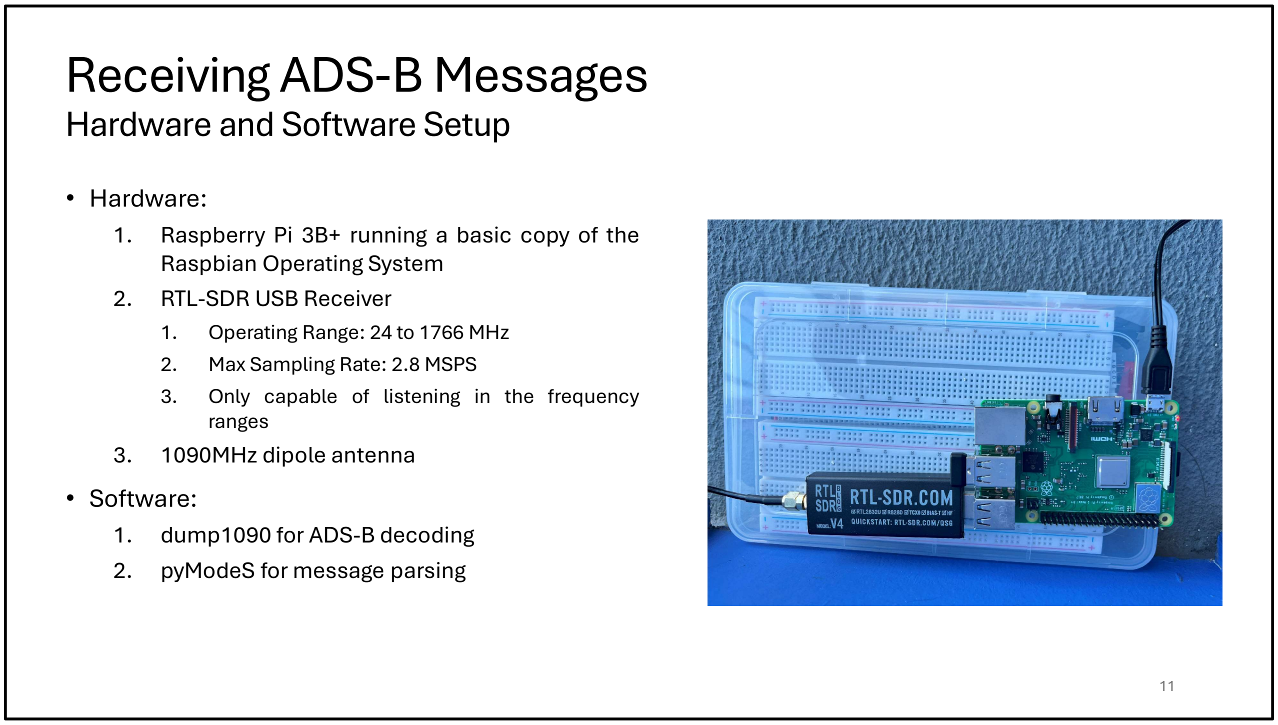
- Incremental update based on operational experience gained.
- Enhanced Integrity Reporting.
- Additional NIC Levels: NICa, NICb & NICc
- Incorporated the ability to broadcast Mode A code in emergency/priority messages.
- Additional SIL supplement bit (SILs)
- NACp and NACv: Same as version 1

4. Version 3: DO-260C

- Focusses on optimizing data transmission and enhancing functionality for modern air traffic management needs.
- Optional Weather Data Broadcasting
- **Autonomous Distress Tracking (ADT):** Automatically transmit an aircraft's position at least once per minute when in distress.
- Support for new applications such as wake turbulence avoidance and hazardous weather detection.

10

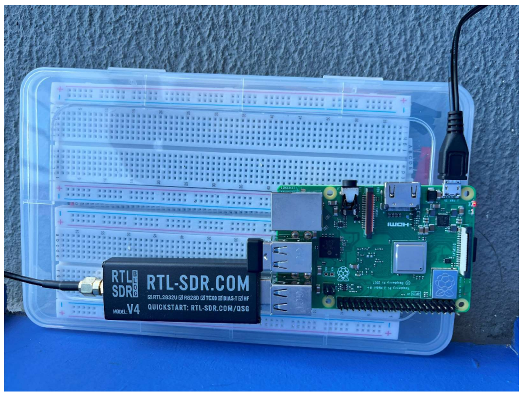
- The latest versions have introduced capabilities like autonomous distress tracking, which can automatically alert authorities if an aircraft is in trouble.



Receiving ADS-B Messages

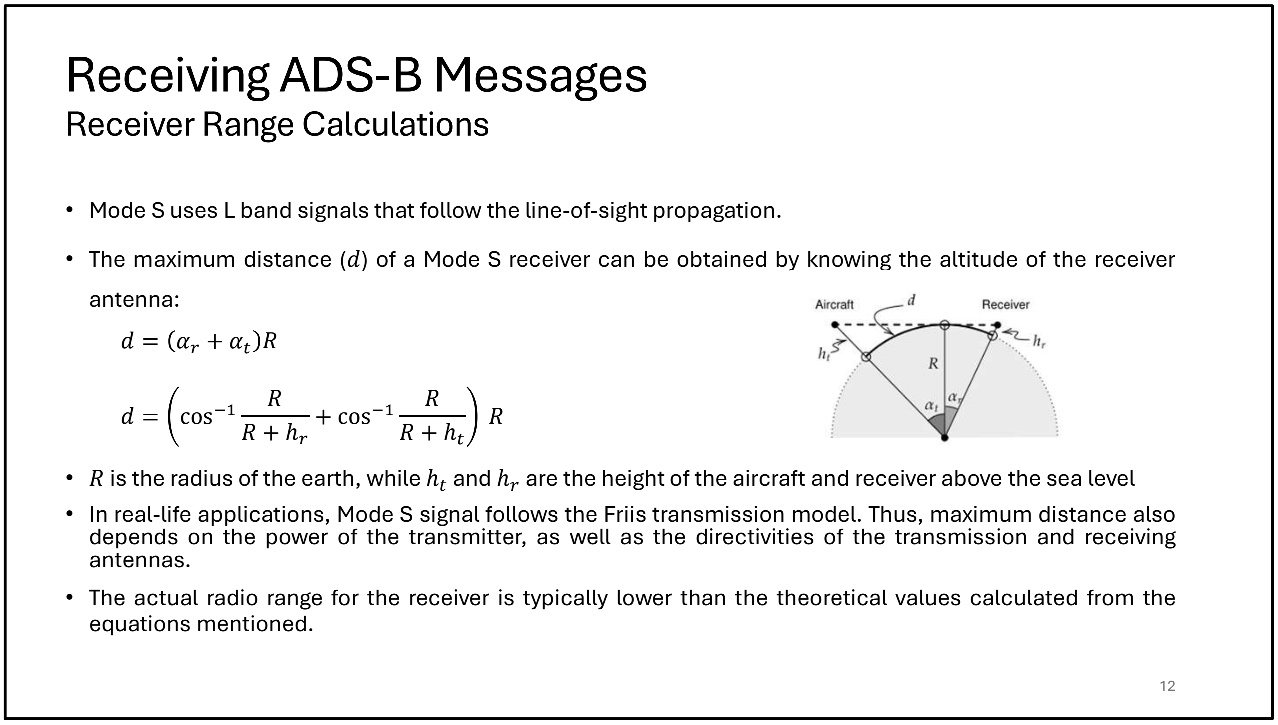
Hardware and Software Setup

- Hardware:
    1. Raspberry Pi 3B+ running a basic copy of the Raspbian Operating System
    2. RTL-SDR USB Receiver
        1. Operating Range: 24 to 1766 MHz
        2. Max Sampling Rate: 2.8 MSPS
        3. Only capable of listening in the frequency ranges
    3. 1090MHz dipole antenna
- Software:
    1. dump1090 for ADS-B decoding
    2. pyModeS for message parsing



11

- So before making ADS-B messages, I decided to capture a few to understand and break them down.
- I created a make-shift receiver that can capture, store and parse ADS-B messages.
- The ADS-B receiving setup uses a Raspberry Pi 3B+ running Raspbian OS as the main hardware.
- An RTL-SDR USB receiver, which can tune into frequencies from 24 MHz to 1.7 GHz, is used to capture the 1090MHz signals. A 1090 MHz dipole antenna is used to optimize reception of ADS-B signals while minimizing interference.
- For software, we run dump1090 to decode ADS-B messages in real-time. I use pyModeS, a Python library that helps parse the decoded messages for further analysis.



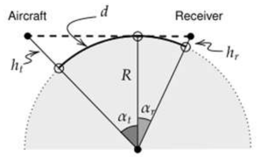
Receiving ADS-B Messages

Receiver Range Calculations

- Mode S uses L band signals that follow the line-of-sight propagation.
- The maximum distance (d) of a Mode S receiver can be obtained by knowing the altitude of the receiver antenna:

$$d = (\alpha_r + \alpha_t)R$$

$$d = \left(\cos^{-1}\frac{R}{R+h_r} + \cos^{-1}\frac{R}{R+h_t}\right)R$$



- 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
- In real-life applications, Mode S signal follows the Friis transmission model. Thus, maximum distance also depends on the power of the transmitter, as well as the directivities of the transmission and receiving antennas.
- The actual radio range for the receiver is typically lower than the theoretical values calculated from the equations mentioned.

12

- The range of Mode S receivers is primarily governed by line-of-sight principles, similar to how far you can see from a high point.
- Theoretically, we can calculate a maximum range based on the heights of the aircraft and receiver, considering the Earth as a perfect sphere. However, real-world signal propagation is influenced by factors like signal power, antenna characteristics, and environmental conditions.

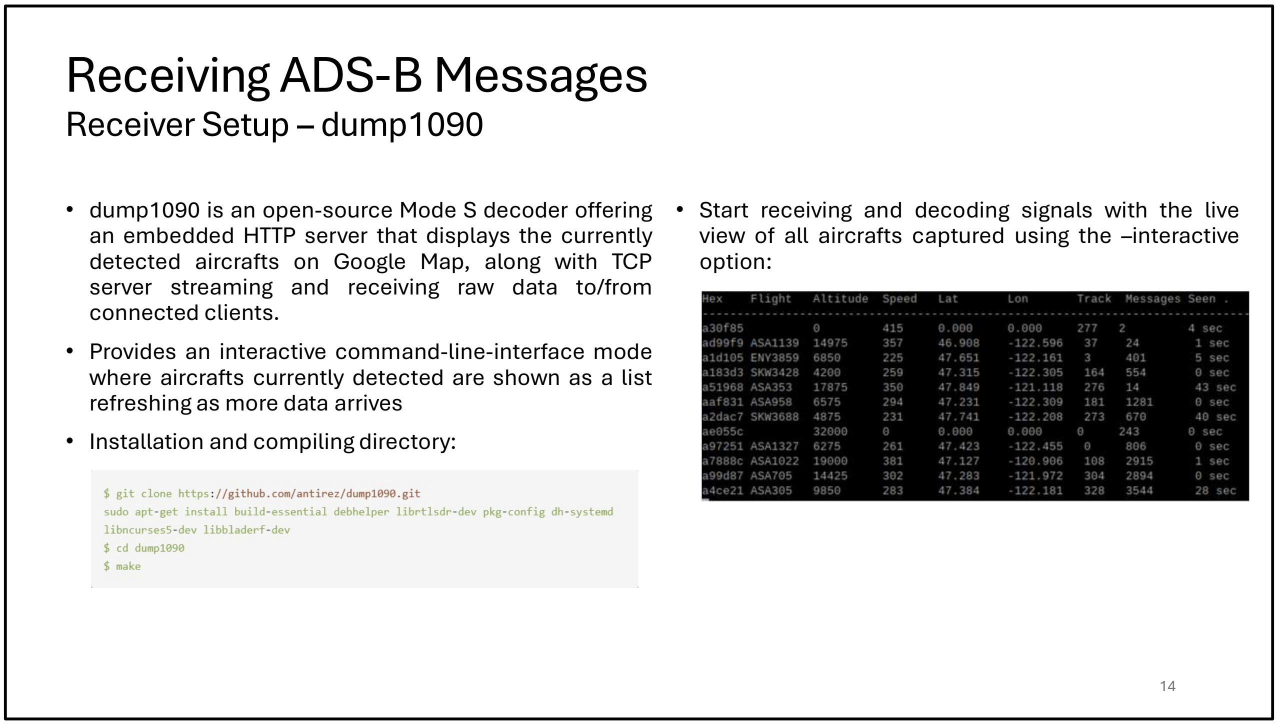
Receiving ADS-B Messages

Antenna Setup

- Any antenna designed for radio frequency around 1 GHz can be used for receiving Mode S signals.
- The carrier frequency of Mode S is 1090 MHz, which corresponds to the wavelength (λ) of 27.5 cm.
- $\lambda = \frac{c}{f}$ where $c = 3 \times 10^8\ m/s\ and\ f = 1090\ MHz$
- A simple metal wire (conductor) and a coaxial feeder cable was used to design a dipole antenna.
- Half-wave antenna implemented with a total conductor length of 13.75 cm

13

- Antenna selection for Mode S reception is flexible, as any antenna designed for frequencies around 1 GHz can be used.
- I opted for a dipole antenna with an appropriate length to capture the ADS-B packets.



Receiving ADS-B Messages

Receiver Setup – dump1090

- dump1090 is an open-source Mode S decoder offering an embedded HTTP server that displays the currently detected aircrafts on Google Map, along with TCP server streaming and receiving raw data to/from connected clients.
- Provides an interactive command-line-interface mode where aircrafts currently detected are shown as a list refreshing as more data arrives
- Installation and compiling directory:

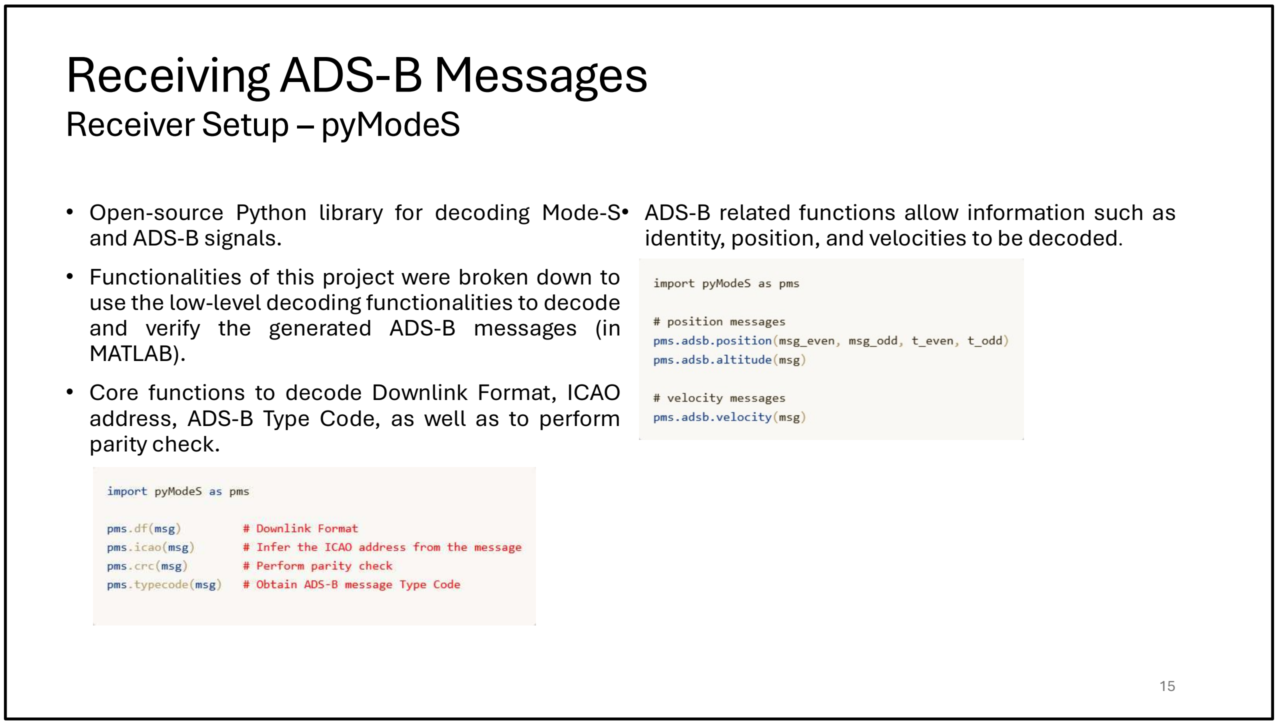
```
$ git clone https://github.com/antirez/dump1090.git
sudo apt-get install build-essential debhelper librtlsdr-dev pkg-config dh-systemd
libncurses5-dev libbladerf-dev
$ cd dump1090
$ make
```

- Start receiving and decoding signals with the live view of all aircrafts captured using the –interactive option:

```
Hex    Flight   Altitude  Speed   Lat       Lon       Track  Messages Seen .
--------------------------------------------------------------------------------
a30f85          0         415     0.000     0.000     277   2          4 sec
ad99f9 ASA1139  14975     357     46.908    -122.596   37    24         1 sec
a1d105 ENY3859  6850      225     47.651    -122.161   3     401        5 sec
a183d3 SKW3428  4200      259     47.315    -122.305   164   554        0 sec
a51968 ASA353   17875     350     47.849    -121.118   276   14         43 sec
aaf831 ASA958   6575      294     47.231    -122.309   181   1281       0 sec
a2dac7 SKW3688  4875      231     47.741    -122.208   273   670        40 sec
ae055c          32000     0       0.000     0.000     0     243        0 sec
a97251 ASA1327  6275      261     47.423    -122.455   0     806        0 sec
a7888c ASA1022  19000     381     47.127    -120.906   108   2915       1 sec
a99d87 ASA705   14425     302     47.283    -121.972   304   2894       0 sec
a4ce21 ASA305   9850      283     47.384    -122.181   328   3544       28 sec
```

14

- dump1090 is an open-source command-line interface that decodes Mode S signals.
- It offers a user-friendly interface by embedding an HTTP server, allowing users to visualize aircraft positions on Google Maps in real-time.
- The software's TCP server capabilities facilitate integration with other systems and applications.



Receiving ADS-B Messages

Receiver Setup – pyModeS

- Open-source Python library for decoding Mode-S and ADS-B signals.
- Functionalities of this project were broken down to use the low-level decoding functionalities to decode and verify the generated ADS-B messages (in MATLAB).
- Core functions to decode Downlink Format, ICAO address, ADS-B Type Code, as well as to perform parity check.

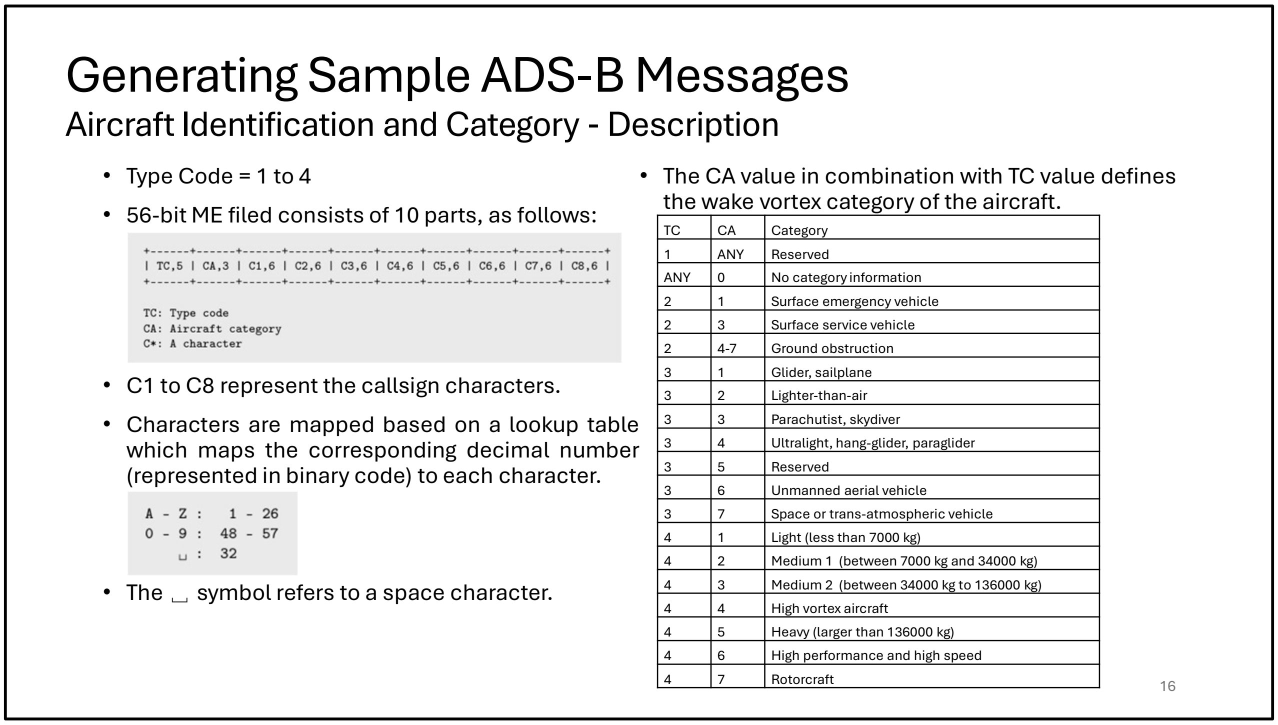
```
import pyModeS as pms

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.

```
import pyModeS as pms

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

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

15

- pyModeS is also an open-source Python library that provides powerful tools for decoding Mode-S signals.
- I focused on utilizing pyModeS' low-level decoding functionalities, which allowed us to break down and verify each component of our generated ADS-B messages.
- The parity check function in pyModeS provided an additional layer of validation.
- This cross-platform verification process, using Python to check MATLAB outputs, demonstrates the interoperability and reliability of the ADS-B message generation.



Generating Sample ADS-B Messages

Aircraft Identification and Category - Description

- Type Code = 1 to 4
- 56-bit ME filed consists of 10 parts, 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
```

- C1 to C8 represent the callsign characters.
- Characters are mapped based on a lookup table which maps the corresponding decimal number (represented in binary code) to each character.

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

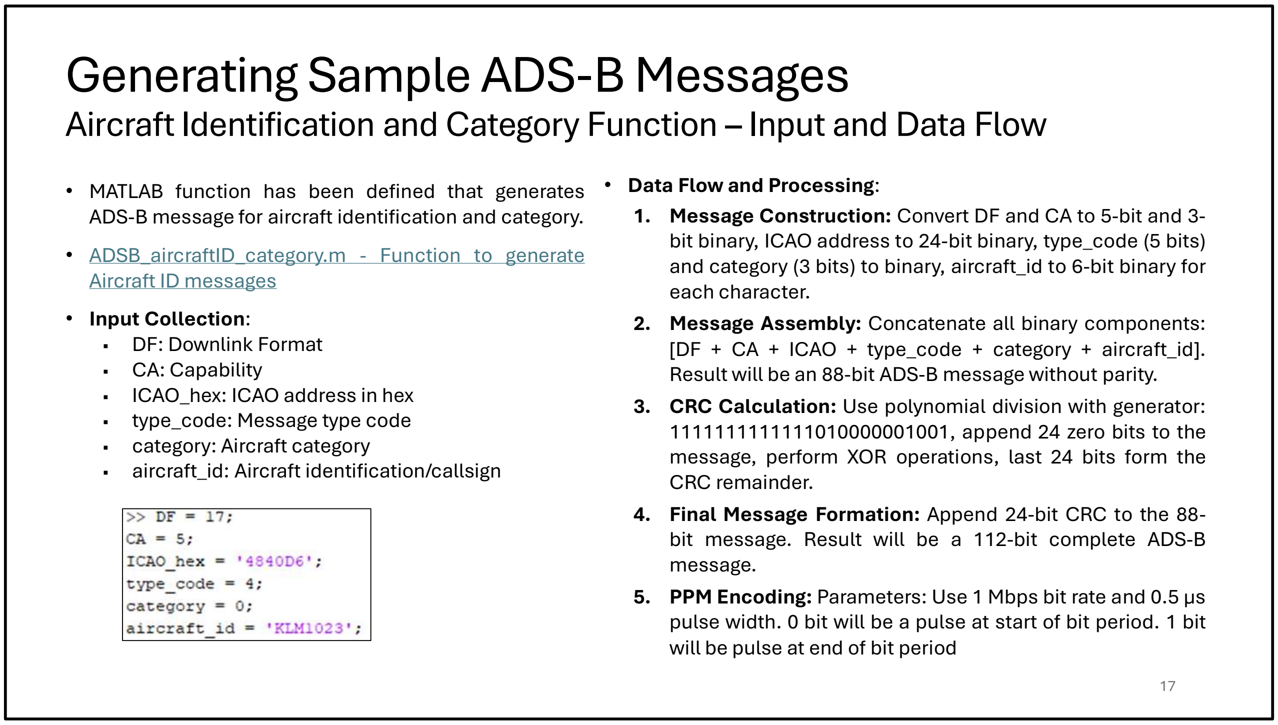
- The ␣ symbol refers to a space character.

- The CA value in combination with TC value defines the wake vortex category of the aircraft.

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 trans-atmospheric 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 and high speed
4	7	Rotorcraft

16

- Aircraft Identification messages are identified by Type Codes 1 to 4. The 56-bit ME field in these messages is divided into 10 parts, each serving a specific purpose in conveying aircraft information.
- The character mapping for the aircraft’s callsign uses a clever binary-to-text encoding scheme, translating binary numbers into readable characters based on a predefined lookup table.
- The CA (Capability) and TC (Type Code) values work together to define the aircraft's wake vortex category. The wake vortex category determines the safe separation distance between aircrafts.



Generating Sample ADS-B Messages

Aircraft Identification and Category Function – Input and Data Flow

- MATLAB function has been defined that generates ADS-B message for aircraft identification and category.
- ADSB_aircraftID_category.m - Function to generate Aircraft ID messages
- **Input Collection**:
  - DF: Downlink Format
  - CA: Capability
  - ICAO_hex: ICAO address in hex
  - type_code: Message type code
  - category: Aircraft category
  - aircraft_id: Aircraft identification/callsign

```
>> DF = 17;
CA = 5;
ICAO_hex = '4840D6';
type_code = 4;
category = 0;
aircraft_id = 'KLM1023';
```

- **Data Flow and Processing**:
  1. **Message Construction:** Convert DF and CA to 5-bit and 3-bit binary, ICAO address to 24-bit binary, type_code (5 bits) and category (3 bits) to binary, aircraft_id to 6-bit binary for each character.
  2. **Message Assembly:** Concatenate all binary components: [DF + CA + ICAO + type_code + category + aircraft_id]. Result will be an 88-bit ADS-B message without parity.
  3. **CRC Calculation:** Use polynomial division with generator: 1111111111111010000001001, append 24 zero bits to the message, perform XOR operations, last 24 bits form the CRC remainder.
  4. **Final Message Formation:** Append 24-bit CRC to the 88-bit message. Result will be a 112-bit complete ADS-B message.
  5. **PPM Encoding:** Parameters: Use 1 Mbps bit rate and 0.5 μs pulse width. 0 bit will be a pulse at start of bit period. 1 bit will be pulse at end of bit period

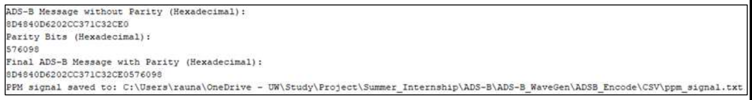
17

- I’ve developed a MATLAB function to generate ADS-B messages specifically for aircraft identification and category information.
- The function takes key inputs like Downlink Format, Capability, ICAO address, message type code, aircraft category, and callsign.
- The function follows several steps of data conversion and assembly, transforming various inputs into their binary representations. This is further encoded with PPM which prepares the message for transmission in a format suitable for ADS-B systems.

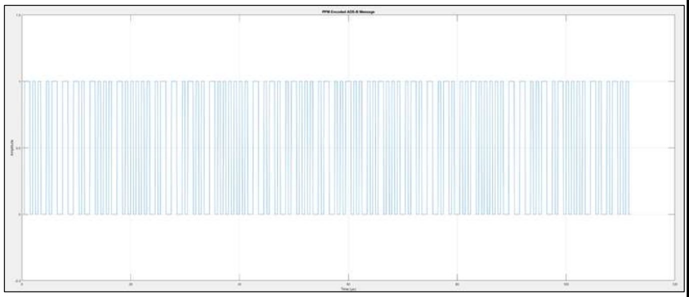
Generating Sample ADS-B Messages

Aircraft Identification and Category Function - Output

- **Outputs:**
  1. **ADS-B Message without Parity**
     Hexadecimal: 8D4840D6202CC371C32CE0
     Breakdown:
     1. 8D: DF (17) and CA (5)
     2. 4840D6: ICAO address
     3. 20: Type Code (4) and Category (1)
     4. 2CC371C32CE0: Encoded Aircraft ID
  2. **Parity Bits**
     Hexadecimal: F1B562
  3. **Final ADS-B Message with Parity**
     Hexadecimal: 8D4840D6202CC371C32CE0576098
     First 88 bits (22 hex characters): Original message
     Last 24 bits (6 hex characters): Appended parity
  4. **PPM Signal**
     Saved as txt file. Contains time and amplitude data.
     Represents the 112-bit message in Pulse Position Modulation format.

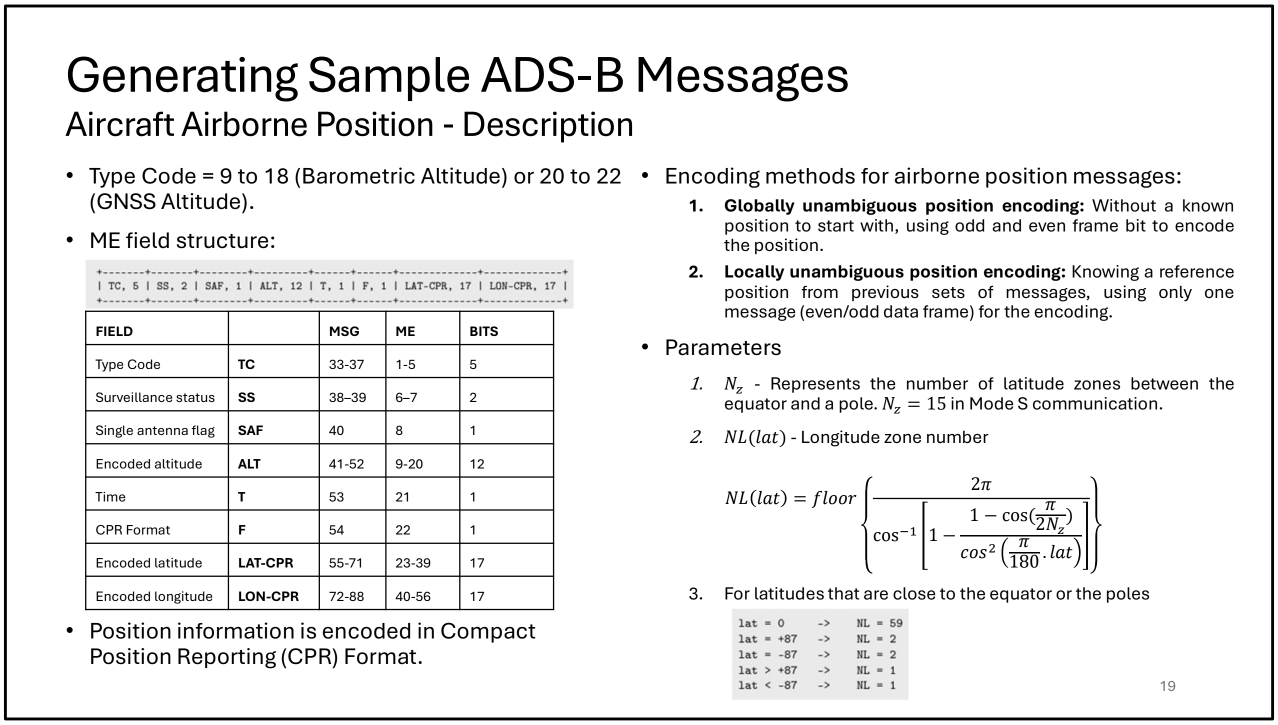


```
ADS-B Message without Parity (Hexadecimal):
8D4840D6202CC371C32CE0
Parity Bits (Hexadecimal):
576098
Final ADS-B Message with Parity (Hexadecimal):
8D4840D6202CC371C32CE0576098
PPM signal saved to: C:\Users\rauna\OneDrive - UW\Study\Project\Summer_Internship\ADS-B\ADS-B_WaveGen\ADSB_Encode\CSV\ppm_signal.txt
```



18

- The function generates a complete ADS-B message.
- The output is presented in multiple formats, from raw hexadecimal to a breakdown of each component.
- The final output includes a Pulse Position Modulation (PPM) signal, representing how the message would be encoded for actual transmission.
- The PPM waveform is then exported as a time-amplitude pairs in a txt file.



Generating Sample ADS-B Messages

Aircraft Airborne Position - Description

- Type Code = 9 to 18 (Barometric Altitude) or 20 to 22 (GNSS Altitude).
- ME field structure:

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

FIELD		MSG	ME	BITS
Type Code	**TC**	33-37	1-5	5
Surveillance status	**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	**F**	54	22	1
Encoded latitude	**LAT-CPR**	55-71	23-39	17
Encoded longitude	**LON-CPR**	72-88	40-56	17

- Position information is encoded in Compact Position Reporting (CPR) Format.

- Encoding methods for airborne position messages:
  1. **Globally unambiguous position encoding:** Without a known position to start with, using odd and even frame bit to encode the position.
  2. **Locally unambiguous position encoding:** Knowing a reference position from previous sets of messages, using only one message (even/odd data frame) for the encoding.
- Parameters
  1. N_z - Represents the number of latitude zones between the equator and a pole. $N_z = 15$ in Mode S communication.
  2. $NL(lat)$ - Longitude zone number

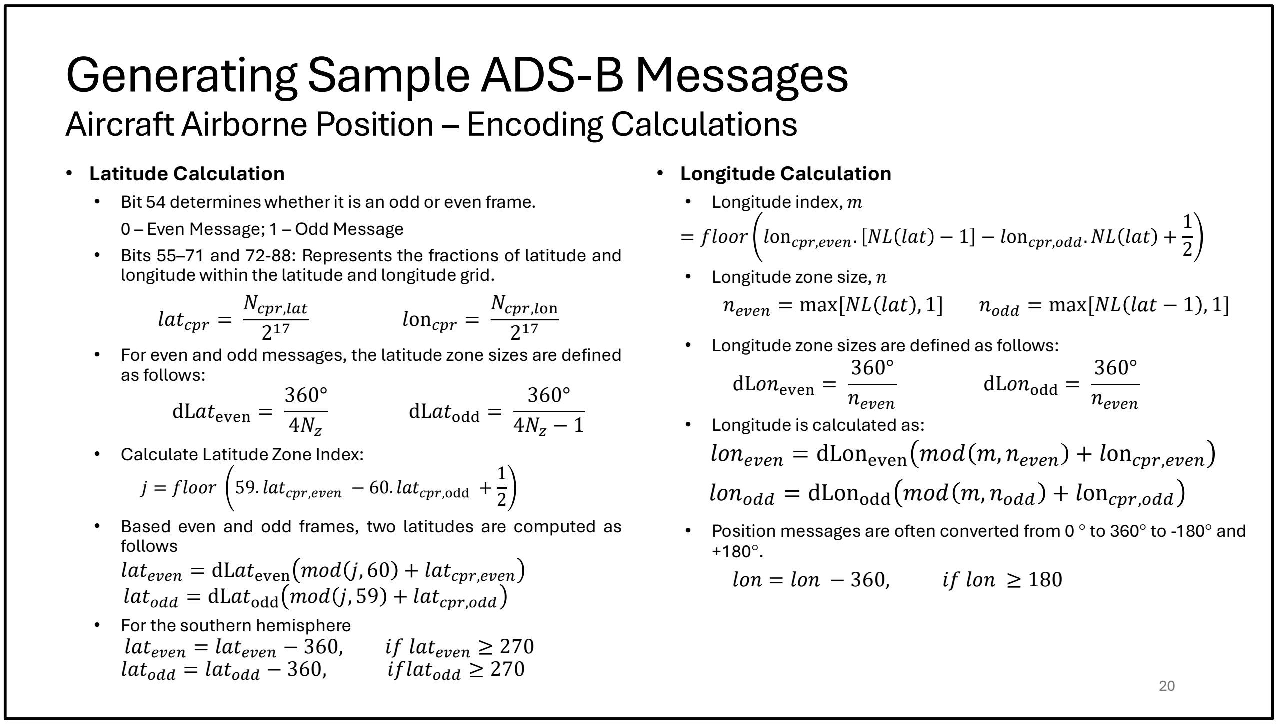
$$NL(lat) = floor\left\{\frac{2\pi}{\cos^{-1}\left[1 - \frac{1-\cos(\frac{\pi}{2N_z})}{cos^2\left(\frac{\pi}{180}.lat\right)}\right]}\right\}$$

  3. For latitudes that are close to the equator or the poles

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

19

- Airborne Position messages are crucial for tracking aircraft location, using Type Codes 9 to 18 for barometric altitude and 20 to 22 for GNSS altitude.
- There are two methods for encoding airborne positions: globally unambiguous (using both odd and even frames) and locally unambiguous (using a single frame with a known reference position).
- The encoding process uses Compact Position Reporting (CPR) format. It efficiently represents latitude and longitude using fewer bits.



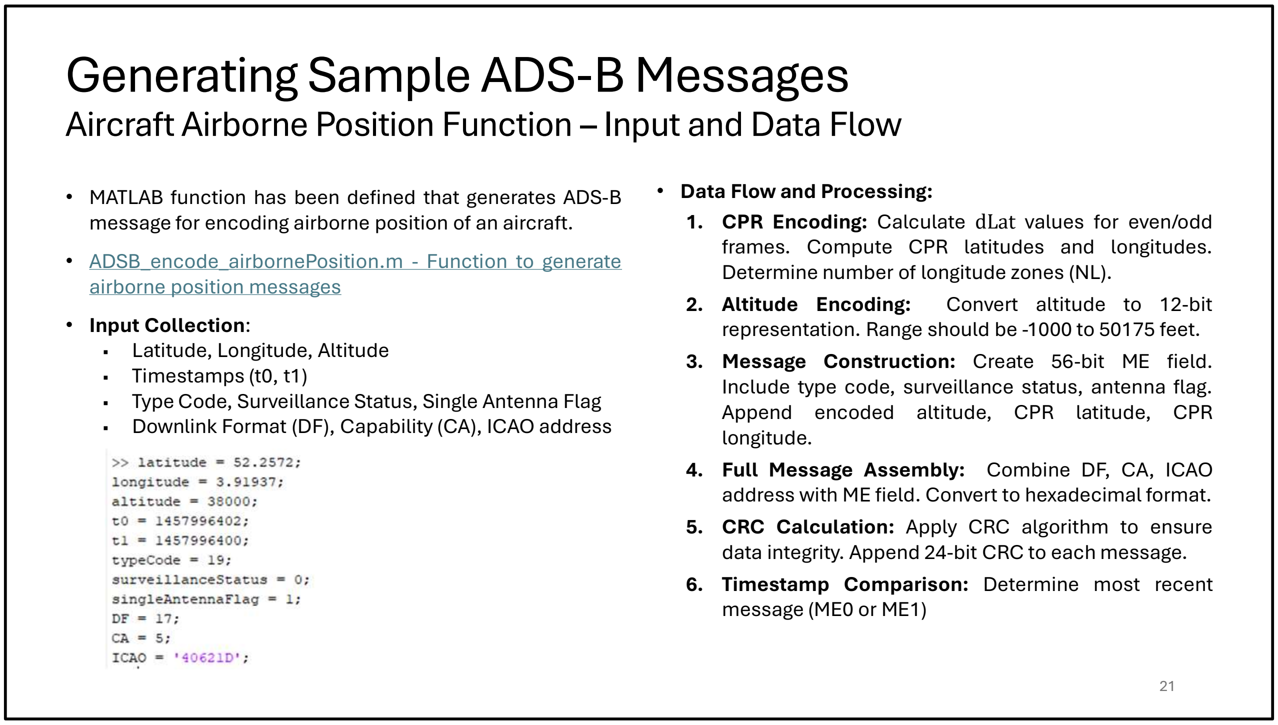
Generating Sample ADS-B Messages

Aircraft Airborne Position – Encoding Calculations

- **Latitude Calculation**
  - Bit 54 determines whether it is an odd or even frame.
    0 – Even Message; 1 – Odd Message
  - Bits 55–71 and 72-88: Represents the fractions of latitude and longitude within the latitude and longitude grid.
    $$lat_{cpr} = \frac{N_{cpr,lat}}{2^{17}} \qquad \mathrm{lon}_{cpr} = \frac{N_{cpr,\mathrm{lon}}}{2^{17}}$$
  - For even and odd messages, the latitude zone sizes are defined as follows:
    $$\mathrm{dL}at_{\mathrm{even}} = \frac{360°}{4N_z} \qquad \mathrm{dL}at_{\mathrm{odd}} = \frac{360°}{4N_z - 1}$$
  - Calculate Latitude Zone Index:
    $$j = floor\left(59.\, lat_{cpr,even} - 60.\, lat_{cpr,\mathrm{odd}} + \frac{1}{2}\right)$$
  - Based even and odd frames, two latitudes are computed as follows
    $$lat_{even} = \mathrm{dL}at_{\mathrm{even}}\left(mod(j,60) + lat_{cpr,even}\right)$$
    $$lat_{odd} = \mathrm{dL}at_{\mathrm{odd}}\left(mod(j,59) + lat_{cpr,odd}\right)$$
  - For the southern hemisphere
    $$lat_{even} = lat_{even} - 360, \qquad if\ lat_{even} \geq 270$$
    $$lat_{odd} = lat_{odd} - 360, \qquad if lat_{odd} \geq 270$$
- **Longitude Calculation**
  - Longitude index, m
    $$= floor\left(\mathrm{lon}_{cpr,even}.\,[NL(lat) - 1] - \mathrm{lon}_{cpr,odd}.\, NL(lat) + \frac{1}{2}\right)$$
  - Longitude zone size, n
    $$n_{even} = \max[NL(lat), 1] \qquad n_{odd} = \max[NL(lat - 1), 1]$$
  - Longitude zone sizes are defined as follows:
    $$\mathrm{dL}on_{\mathrm{even}} = \frac{360°}{n_{even}} \qquad \mathrm{dL}on_{\mathrm{odd}} = \frac{360°}{n_{even}}$$
  - Longitude is calculated as:
    $$lon_{even} = \mathrm{dLon}_{\mathrm{even}}\left(mod(m, n_{even}) + \mathrm{lon}_{cpr,even}\right)$$
    $$lon_{odd} = \mathrm{dLon}_{\mathrm{odd}}\left(mod(m, n_{odd}) + \mathrm{lon}_{cpr,odd}\right)$$
  - Position messages are often converted from 0 ° to 360° to -180° and +180°.
    $$lon = lon - 360, \qquad if\ lon \geq 180$$

20

- Calculations have been defined for calculating the latitude and longitude from ADS-B data. These calculations have been reversed to encode user-defined latitude and longitude into the ME fields.



Generating Sample ADS-B Messages

Aircraft Airborne Position Function – Input and Data Flow

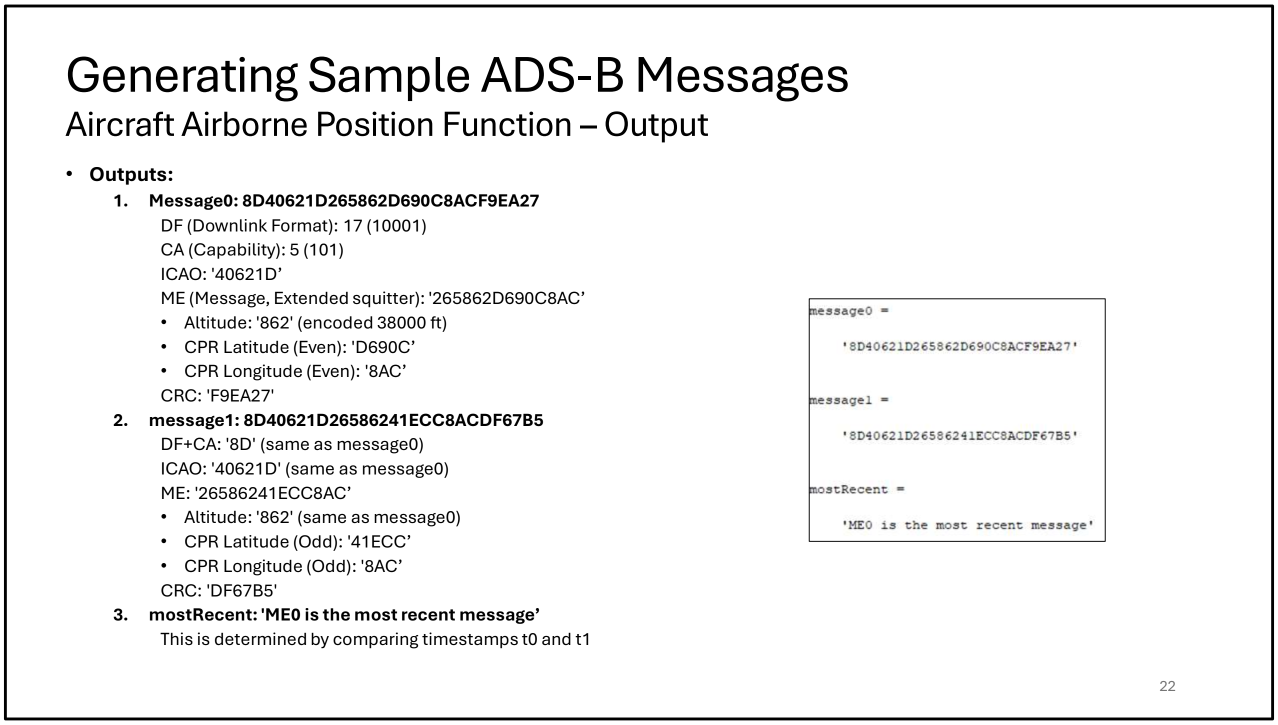
- MATLAB function has been defined that generates ADS-B message for encoding airborne position of an aircraft.
- ADSB_encode_airbornePosition.m - Function to generate airborne position messages
- **Input Collection**:
  - Latitude, Longitude, Altitude
  - Timestamps (t0, t1)
  - Type Code, Surveillance Status, Single Antenna Flag
  - Downlink Format (DF), Capability (CA), ICAO address

```
>> latitude = 52.2572;
longitude = 3.91937;
altitude = 38000;
t0 = 1457996402;
t1 = 1457996400;
typeCode = 19;
surveillanceStatus = 0;
singleAntennaFlag = 1;
DF = 17;
CA = 5;
ICAO = '40621D';
```

- **Data Flow and Processing:**
  1. **CPR Encoding:** Calculate dLat values for even/odd frames. Compute CPR latitudes and longitudes. Determine number of longitude zones (NL).
  2. **Altitude Encoding:** Convert altitude to 12-bit representation. Range should be -1000 to 50175 feet.
  3. **Message Construction:** Create 56-bit ME field. Include type code, surveillance status, antenna flag. Append encoded altitude, CPR latitude, CPR longitude.
  4. **Full Message Assembly:** Combine DF, CA, ICAO address with ME field. Convert to hexadecimal format.
  5. **CRC Calculation:** Apply CRC algorithm to ensure data integrity. Append 24-bit CRC to each message.
  6. **Timestamp Comparison:** Determine most recent message (ME0 or ME1)

21

- A MATLAB function has been developed to generate ADS-B messages specifically for encoding aircraft airborne positions.
- Input collection is comprehensive, including data like latitude, longitude, altitude, and timestamps, as well as technical parameters such as Type Code and ICAO address.
- The data flow is designed to encode the input data into a 56-bit ME field.
- This encoding message follows Globally unambiguous position encoding. It produces an even and odd ADS-B message, marking the most recent one as per the timestamps.



Generating Sample ADS-B Messages

Aircraft Airborne Position Function – Output

- **Outputs:**
  1. **Message0: 8D40621D265862D690C8ACF9EA27**
     DF (Downlink Format): 17 (10001)
     CA (Capability): 5 (101)
     ICAO: '40621D'
     ME (Message, Extended squitter): '265862D690C8AC'
     - Altitude: '862' (encoded 38000 ft)
     - CPR Latitude (Even): 'D690C'
     - CPR Longitude (Even): '8AC'

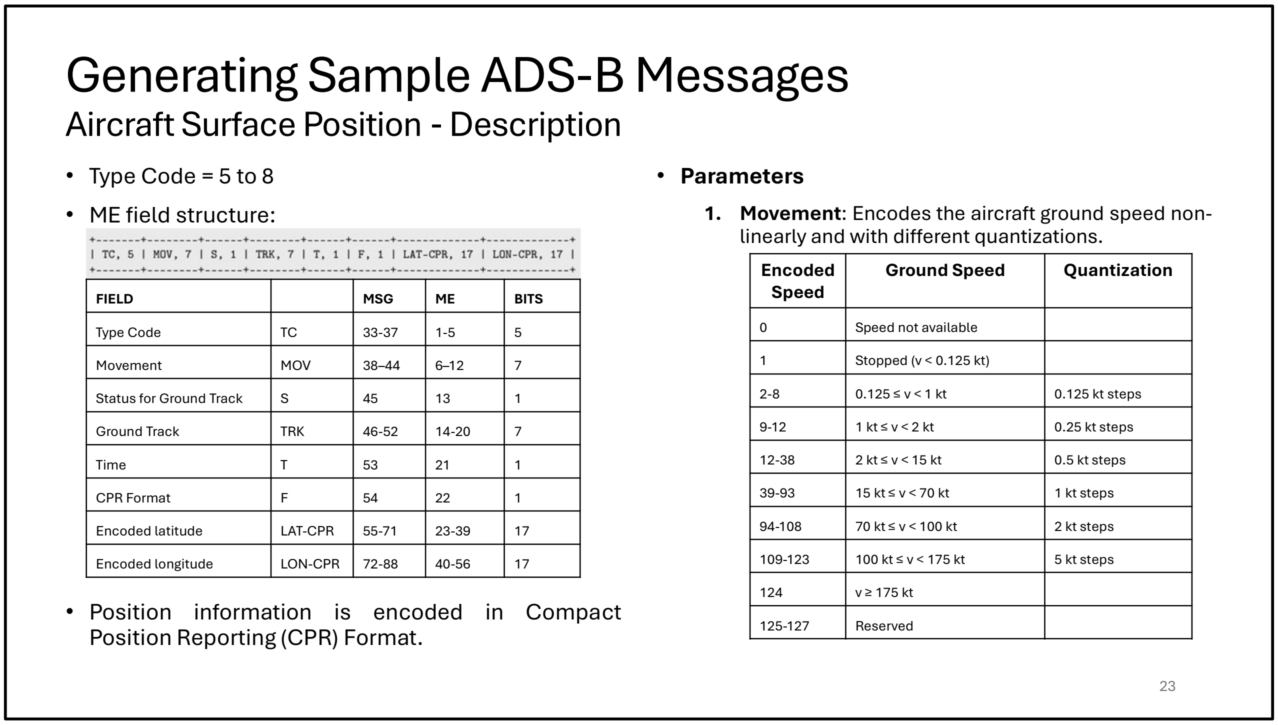
     CRC: 'F9EA27'
  2. **message1: 8D40621D26586241ECC8ACDF67B5**
     DF+CA: '8D' (same as message0)
     ICAO: '40621D' (same as message0)
     ME: '26586241ECC8AC'
     - Altitude: '862' (same as message0)
     - CPR Latitude (Odd): '41ECC'
     - CPR Longitude (Odd): '8AC'

     CRC: 'DF67B5'
  3. **mostRecent: 'ME0 is the most recent message'**
     This is determined by comparing timestamps t0 and t1

```
message0 =

    '8D40621D265862D690C8ACF9EA27'


message1 =

    '8D40621D26586241ECC8ACDF67B5'


mostRecent =

    'ME0 is the most recent message'
```

22

- The function produces two complete ADS-B messages, typically referred to as Message0 and Message1, representing even and odd frames respectively.
- Altitude information is consistent across both messages, while CPR latitude and longitude values differ between even and odd frames.
- The function determines which message is the most recent based on timestamp comparison.



Generating Sample ADS-B Messages

Aircraft Surface Position - Description

- Type Code = 5 to 8
- ME field structure:

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

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

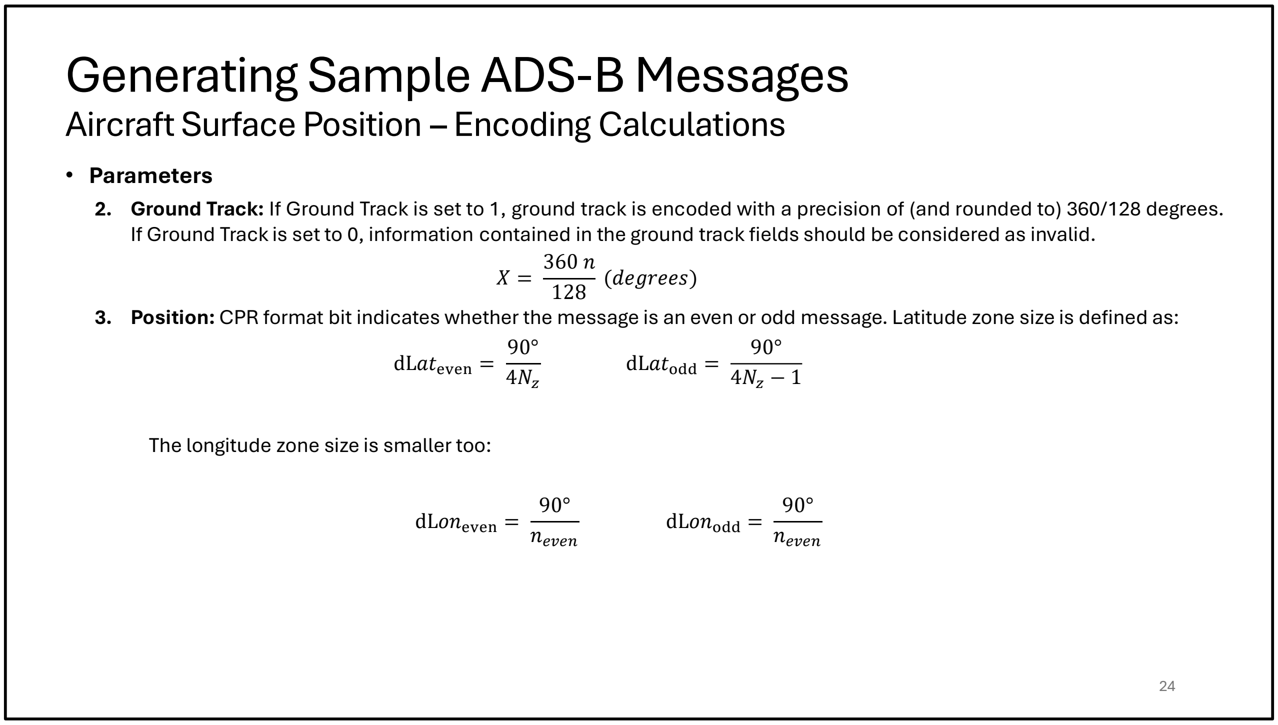
- Position information is encoded in Compact Position Reporting (CPR) Format.

- **Parameters**
  1. **Movement**: Encodes the aircraft ground speed non-linearly and with different quantizations.

Encoded Speed	Ground Speed	Quantization
0	Speed not available	
1	Stopped (v < 0.125 kt)	
2-8	0.125 ≤ v < 1 kt	0.125 kt steps
9-12	1 kt ≤ v < 2 kt	0.25 kt steps
12-38	2 kt ≤ v < 15 kt	0.5 kt steps
39-93	15 kt ≤ v < 70 kt	1 kt steps
94-108	70 kt ≤ v < 100 kt	2 kt steps
109-123	100 kt ≤ v < 175 kt	5 kt steps
124	v ≥ 175 kt	
125-127	Reserved	

23

- Surface Position messages, identified by Type Codes 5 to 8, are specifically designed for aircraft on the ground.
- Like the last 2 message types, this follows a strict encoding scheme with the CPR format.



Generating Sample ADS-B Messages

Aircraft Surface Position – Encoding Calculations

- **Parameters**
  2. **Ground Track:** If Ground Track is set to 1, ground track is encoded with a precision of (and rounded to) 360/128 degrees. If Ground Track is set to 0, information contained in the ground track fields should be considered as invalid.

$$X = \frac{360\,n}{128}\ (degrees)$$

  3. **Position:** CPR format bit indicates whether the message is an even or odd message. Latitude zone size is defined as:

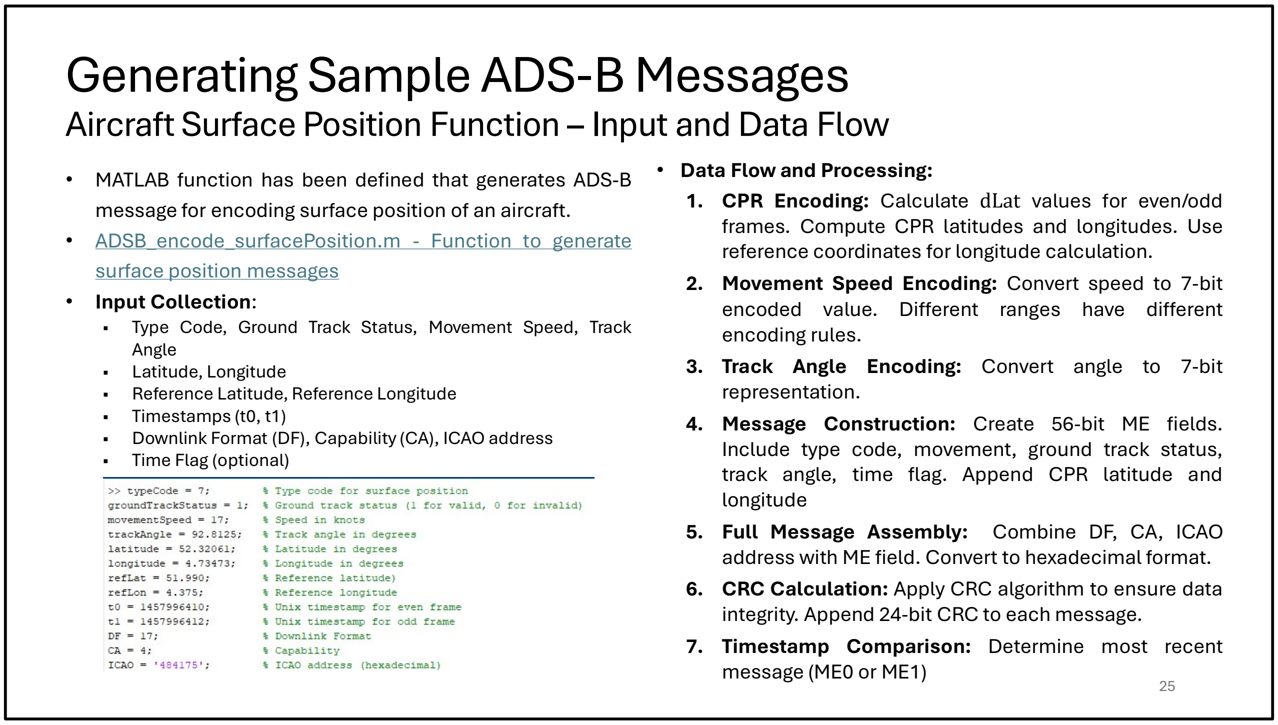
$$\mathrm{dL}at_{\mathrm{even}} = \frac{90°}{4N_z} \qquad \mathrm{dL}at_{\mathrm{odd}} = \frac{90°}{4N_z - 1}$$

The longitude zone size is smaller too:

$$\mathrm{dL}on_{\mathrm{even}} = \frac{90°}{n_{even}} \qquad \mathrm{dL}on_{\mathrm{odd}} = \frac{90°}{n_{even}}$$

24

- Fundamental parameters are considered and used to encode user-defined data into the 56bit ME field.



Generating Sample ADS-B Messages

Aircraft Surface Position Function – Input and Data Flow

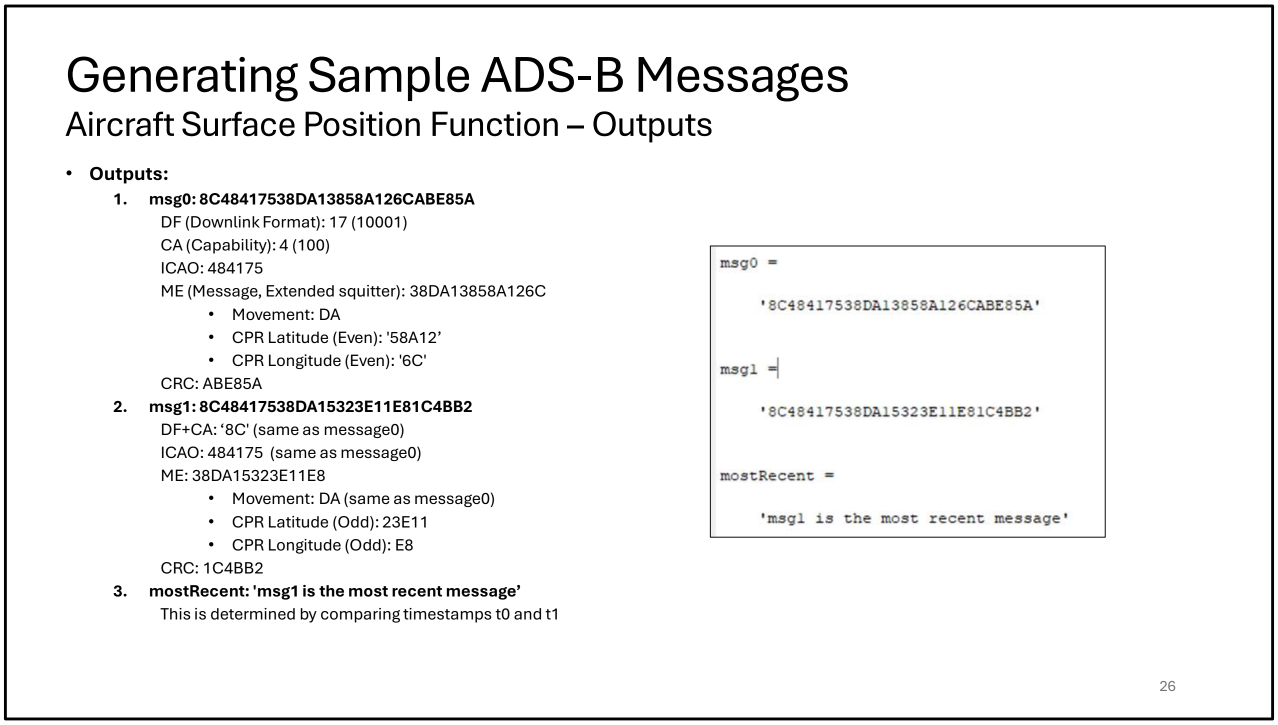
- MATLAB function has been defined that generates ADS-B message for encoding surface position of an aircraft.
- ADSB_encode_surfacePosition.m - Function to generate surface position messages
- **Input Collection**:
  - Type Code, Ground Track Status, Movement Speed, Track Angle
  - Latitude, Longitude
  - Reference Latitude, Reference Longitude
  - Timestamps (t0, t1)
  - Downlink Format (DF), Capability (CA), ICAO address
  - Time Flag (optional)

```
>> typeCode = 7;          % Type code for surface position
groundTrackStatus = 1;  % Ground track status (1 for valid, 0 for invalid)
movementSpeed = 17;     % Speed in knots
trackAngle = 92.8125;   % Track angle in degrees
latitude = 52.32061;    % Latitude in degrees
longitude = 4.73473;    % Longitude in degrees
refLat = 51.990;        % Reference latitude)
refLon = 4.375;         % Reference longitude
t0 = 1457996410;        % Unix timestamp for even frame
t1 = 1457996412;        % Unix timestamp for odd frame
DF = 17;                % Downlink Format
CA = 4;                 % Capability
ICAO = '484175';        % ICAO address (hexadecimal)
```

- **Data Flow and Processing:**
  1. **CPR Encoding:** Calculate dLat values for even/odd frames. Compute CPR latitudes and longitudes. Use reference coordinates for longitude calculation.
  2. **Movement Speed Encoding:** Convert speed to 7-bit encoded value. Different ranges have different encoding rules.
  3. **Track Angle Encoding:** Convert angle to 7-bit representation.
  4. **Message Construction:** Create 56-bit ME fields. Include type code, movement, ground track status, track angle, time flag. Append CPR latitude and longitude
  5. **Full Message Assembly:** Combine DF, CA, ICAO address with ME field. Convert to hexadecimal format.
  6. **CRC Calculation:** Apply CRC algorithm to ensure data integrity. Append 24-bit CRC to each message.
  7. **Timestamp Comparison:** Determine most recent message (ME0 or ME1)

25

- The MATLAB function developed to encode these type of messages takes the comprehensive input collection, encodes the required parameters and creates the 56 bits for the ME field.
- As usual, full message assembly and CRC calculation ensure the integrity and completeness of the ADS-B message.



Generating Sample ADS-B Messages

Aircraft Surface Position Function – Outputs

- **Outputs:**
  1. **msg0: 8C48417538DA13858A126CABE85A**
     DF (Downlink Format): 17 (10001)
     CA (Capability): 4 (100)
     ICAO: 484175
     ME (Message, Extended squitter): 38DA13858A126C
     - Movement: DA
     - CPR Latitude (Even): '58A12'
     - CPR Longitude (Even): '6C'

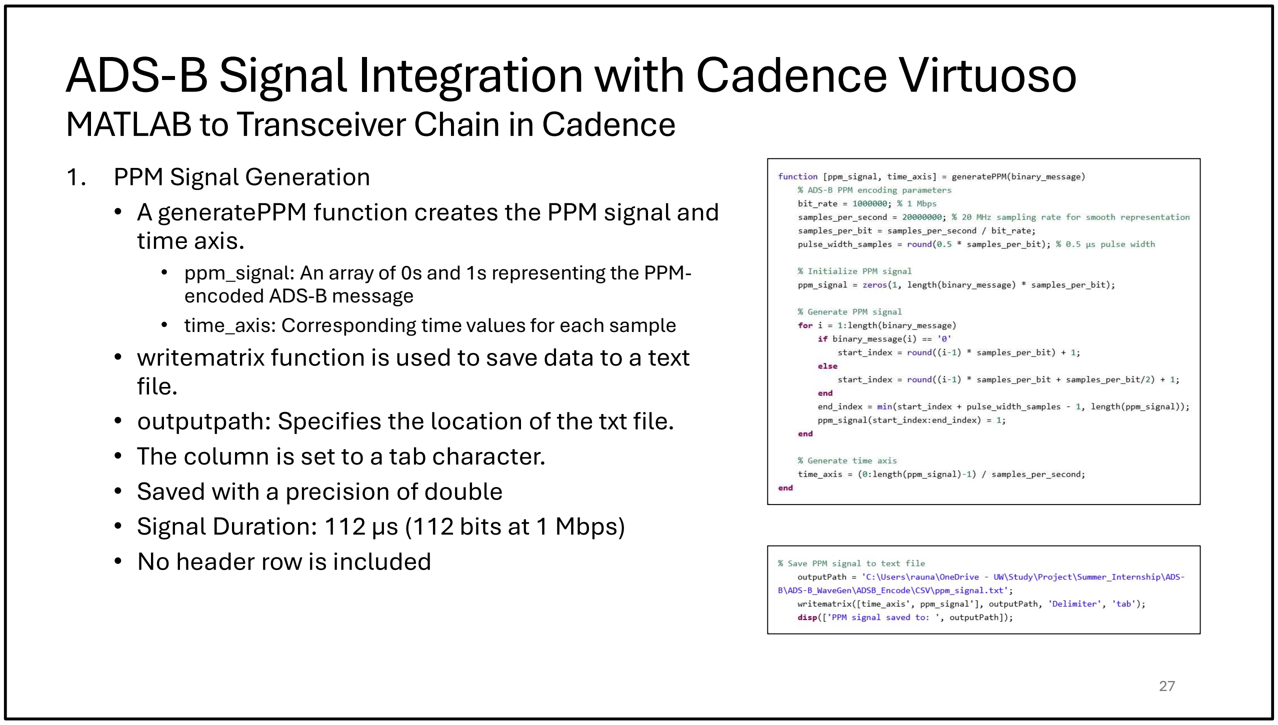
     CRC: ABE85A
  2. **msg1: 8C48417538DA15323E11E81C4BB2**
     DF+CA: '8C' (same as message0)
     ICAO: 484175 (same as message0)
     ME: 38DA15323E11E8
     - Movement: DA (same as message0)
     - CPR Latitude (Odd): 23E11
     - CPR Longitude (Odd): E8

     CRC: 1C4BB2
  3. **mostRecent: 'msg1 is the most recent message'**
     This is determined by comparing timestamps t0 and t1

```
msg0 =

    '8C48417538DA13858A126CABE85A'


msg1 =

    '8C48417538DA15323E11E81C4BB2'


mostRecent =

    'msg1 is the most recent message'
```

26

- The function produces two complete ADS-B messages, msg0 and msg1, representing even and odd frames for surface position reporting.
- The function determines which message is the most recent based on timestamp comparison.



ADS-B Signal Integration with Cadence Virtuoso

MATLAB to Transceiver Chain in Cadence

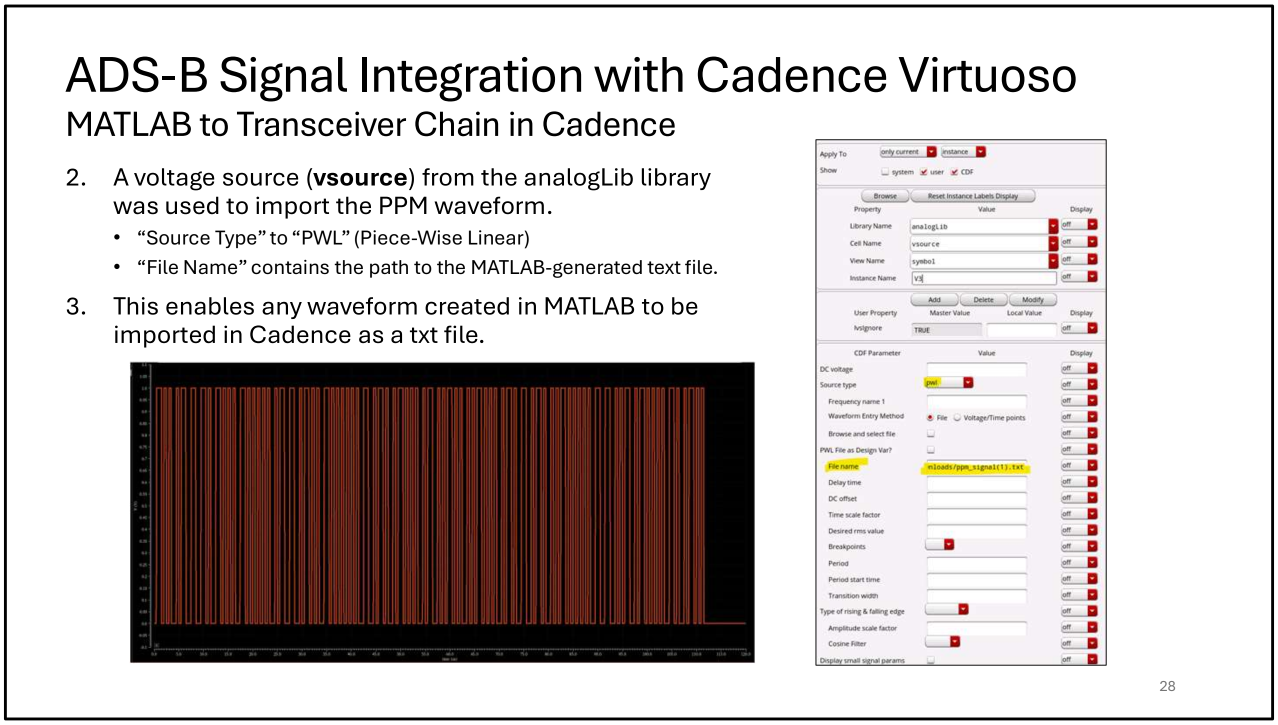
1. PPM Signal Generation
   - A generatePPM function creates the PPM signal and time axis.
     - ppm_signal: An array of 0s and 1s representing the PPM-encoded ADS-B message
     - time_axis: Corresponding time values for each sample
   - writematrix function is used to save data to a text file.
   - outputpath: Specifies the location of the txt file.
   - The column is set to a tab character.
   - Saved with a precision of double
   - Signal Duration: 112 μs (112 bits at 1 Mbps)
   - No header row is included

```
function [ppm_signal, time_axis] = generatePPM(binary_message)
    % ADS-B PPM encoding parameters
    bit_rate = 1000000; % 1 Mbps
    samples_per_second = 20000000; % 20 MHz sampling rate for smooth representation
    samples_per_bit = samples_per_second / bit_rate;
    pulse_width_samples = round(0.5 * samples_per_bit); % 0.5 μs pulse width

    % Initialize PPM signal
    ppm_signal = zeros(1, length(binary_message) * samples_per_bit);

    % Generate PPM signal
    for i = 1:length(binary_message)
        if binary_message(i) == '0'
            start_index = round((i-1) * samples_per_bit) + 1;
        else
            start_index = round((i-1) * samples_per_bit + samples_per_bit/2) + 1;
        end
        end_index = min(start_index + pulse_width_samples - 1, length(ppm_signal));
        ppm_signal(start_index:end_index) = 1;
    end

    % Generate time axis
    time_axis = (0:length(ppm_signal)-1) / samples_per_second;
end
```

```
% Save PPM signal to text file
    outputPath = 'C:\Users\rauna\OneDrive - UW\Study\Project\Summer_Internship\ADS-
B\ADS-B_WaveGen\ADSB_Encode\CSV\ppm_signal.txt';
    writematrix([time_axis', ppm_signal'], outputPath, 'Delimiter', 'tab');
    disp(['PPM signal saved to: ', outputPath]);
```

27

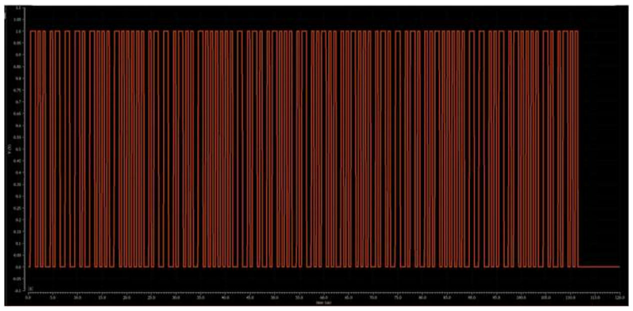
- The generatePPM function creates two outputs: a PPM (Pulse Position Modulation) signal and a corresponding time axis, representing the ADS-B message in a waveform format.
- The signal is 112μs long, corresponding to 112 bits at 1 Mbps.
- The output text file is formatted with tab-separated columns and double precision, ensuring compatibility with Cadence.
- This function enables the ADS-B messages to be imported into Cadence for further analysis and testing.

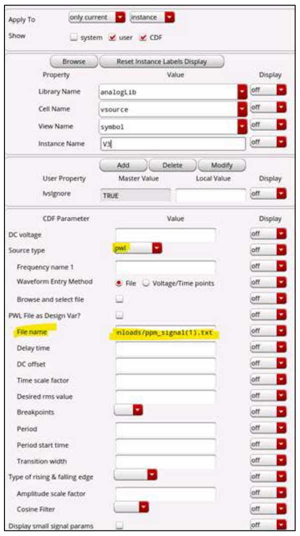


ADS-B Signal Integration with Cadence Virtuoso

MATLAB to Transceiver Chain in Cadence

2. A voltage source (**vsource**) from the analogLib library was used to import the PPM waveform.
   - "Source Type" to "PWL" (Piece-Wise Linear)
   - "File Name" contains the path to the MATLAB-generated text file.
3. This enables any waveform created in MATLAB to be imported in Cadence as a txt file.

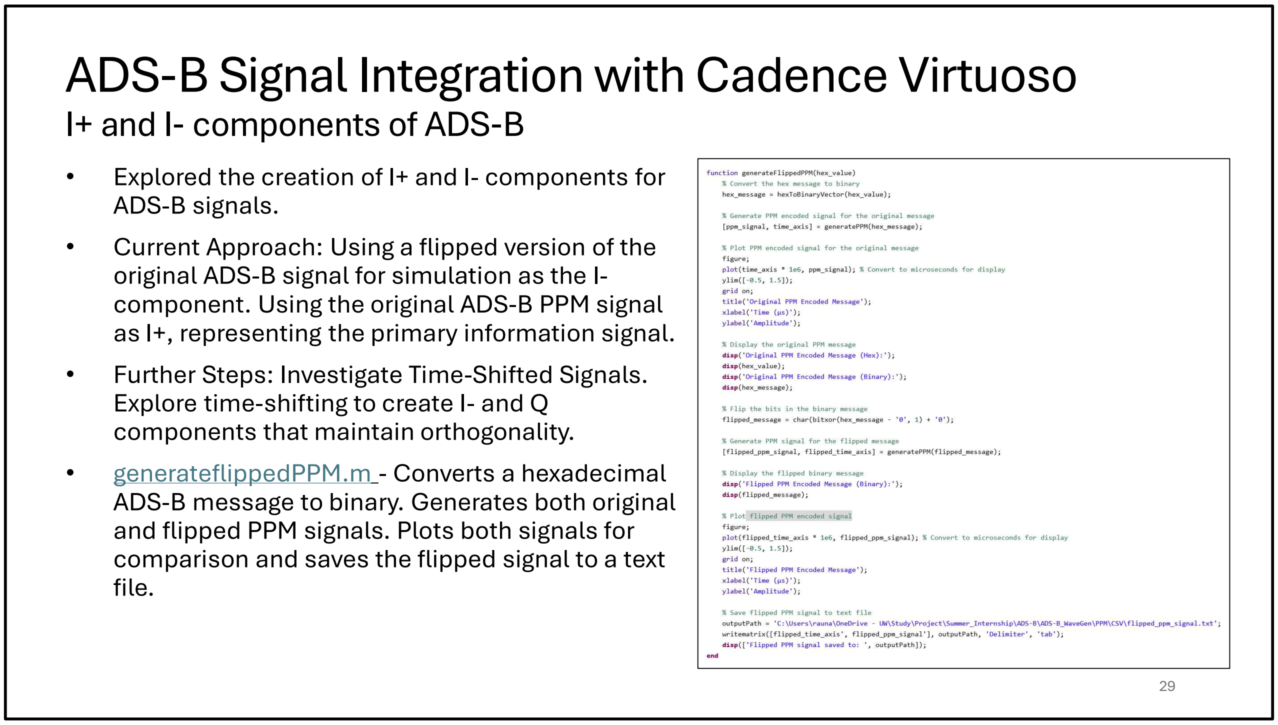




Apply To	only current / instance	
Show	[ ] system [✓ user] [✓ CDF]	
Property	Value	Display
Library Name	analogLib	off
Cell Name	vsource	off
View Name	symbol	off
Instance Name	V3	off
User Property	Master Value	Display
lvsIgnore	TRUE	off
CDF Parameter	Value	Display
DC voltage		off
Source type	pwl	off
Frequency name 1		off
Waveform Entry Method	[✓ File] Voltage/Time points	off
Browse and select file		off
PWL File as Design Var?		off
File name	nloads/ppm_signal(1).txt	off
Delay time		off
DC offset		off
Time scale factor		off
Desired rms value		off
Breakpoints		off
Period		off
Period start time		off
Transition width		off
Type of rising & falling edge		off
Amplitude scale factor		off
Cosine Filter		off
Display small signal params		off

28

- In Cadence, A voltage source (vsource) from the analogLib library is utilized for importing the PPM waveform.
- With using the vsource as a Piece-Wise Linear source, we can directly import any waveform created in MATLAB into Cadence.



ADS-B Signal Integration with Cadence Virtuoso

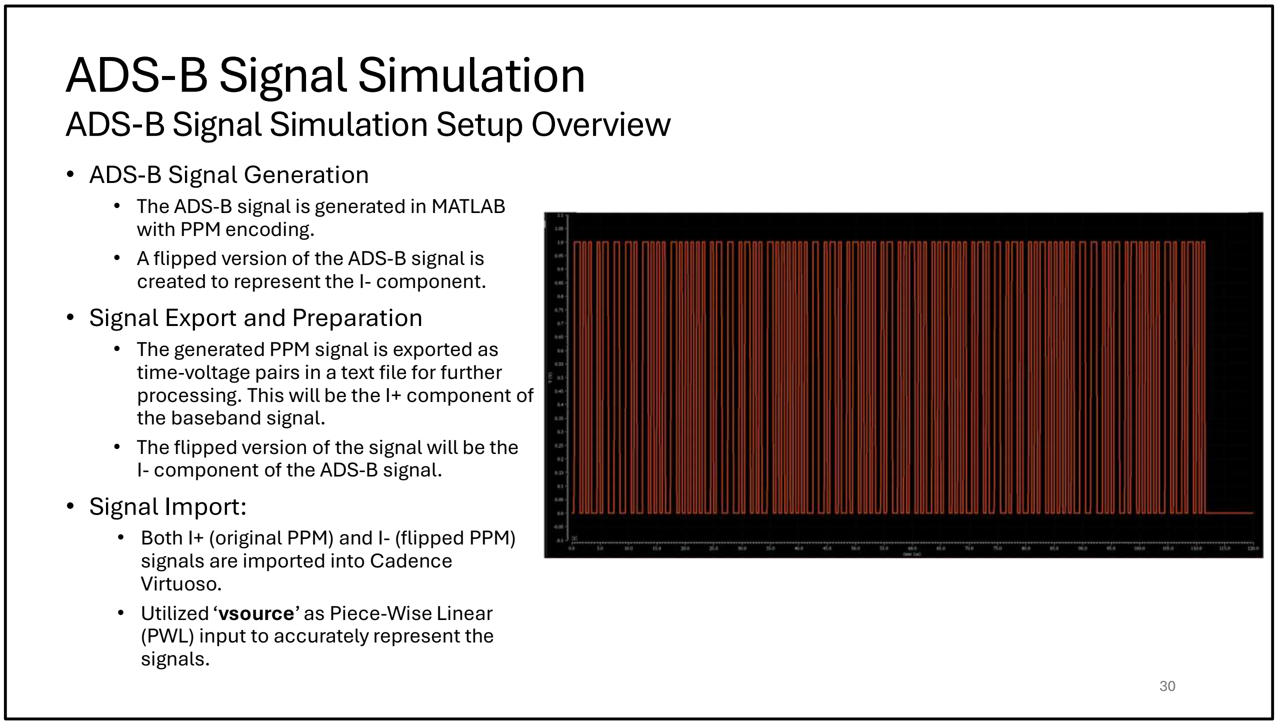
I+ and I- components of ADS-B

- Explored the creation of I+ and I- components for ADS-B signals.
- Current Approach: Using a flipped version of the original ADS-B signal for simulation as the I-component. Using the original ADS-B PPM signal as I+, representing the primary information signal.
- Further Steps: Investigate Time-Shifted Signals. Explore time-shifting to create I- and Q components that maintain orthogonality.
- generateflippedPPM.m - Converts a hexadecimal ADS-B message to binary. Generates both original and flipped PPM signals. Plots both signals for comparison and saves the flipped signal to a text file.

```
function generateFlippedPPM(hex_value)
    % Convert the hex message to binary
    hex_message = hexToBinaryVector(hex_value);

    % Generate PPM encoded signal for the original message
    [ppm_signal, time_axis] = generatePPM(hex_message);

    % Plot PPM encoded signal for the original message
    figure;
    plot(time_axis * 1e6, ppm_signal); % Convert to microseconds for display
    ylim([-0.5, 1.5]);
    grid on;
    title('Original PPM Encoded Message');
    xlabel('Time (µs)');
    ylabel('Amplitude');

    % Display the original PPM message
    disp('Original PPM Encoded Message (Hex):');
    disp(hex_value);
    disp('Original PPM Encoded Message (Binary):');
    disp(hex_message);

    % Flip the bits in the binary message
    flipped_message = char(bitxor(hex_message - '0', 1) + '0');

    % Generate PPM signal for the flipped message
    [flipped_ppm_signal, flipped_time_axis] = generatePPM(flipped_message);

    % Display the flipped binary message
    disp('Flipped PPM Encoded Message (Binary):');
    disp(flipped_message);

    % Plot flipped PPM encoded signal
    figure;
    plot(flipped_time_axis * 1e6, flipped_ppm_signal); % Convert to microseconds for display
    ylim([-0.5, 1.5]);
    grid on;
    title('Flipped PPM Encoded Message');
    xlabel('Time (µs)');
    ylabel('Amplitude');

    % Save flipped PPM signal to text file
    outputPath = 'C:\Users\rauna\OneDrive - UW\Study\Project\Summer_Internship\ADS-B\ADS-B_WaveGen\PPM\CSV\flipped_ppm_signal.txt';
    writematrix([flipped_time_axis', flipped_ppm_signal'], outputPath, 'Delimiter', 'tab');
    disp(['Flipped PPM signal saved to: ', outputPath]);
end
```

29

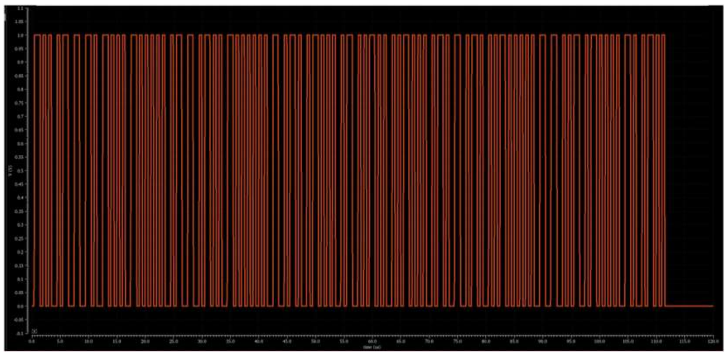
- Our current approach utilizes the original ADS-B PPM signal as the I+ component, representing the primary information signal. For the I-component, we're employing a flipped version of the original ADS-B signal.
- Another MATLAB function has been developed to do this process, converting hexadecimal ADS-B messages to binary and generating both original and flipped PPM signals.
- The flipped signal is saved to a text file, allowing for easy integration with Cadence.



ADS-B Signal Simulation

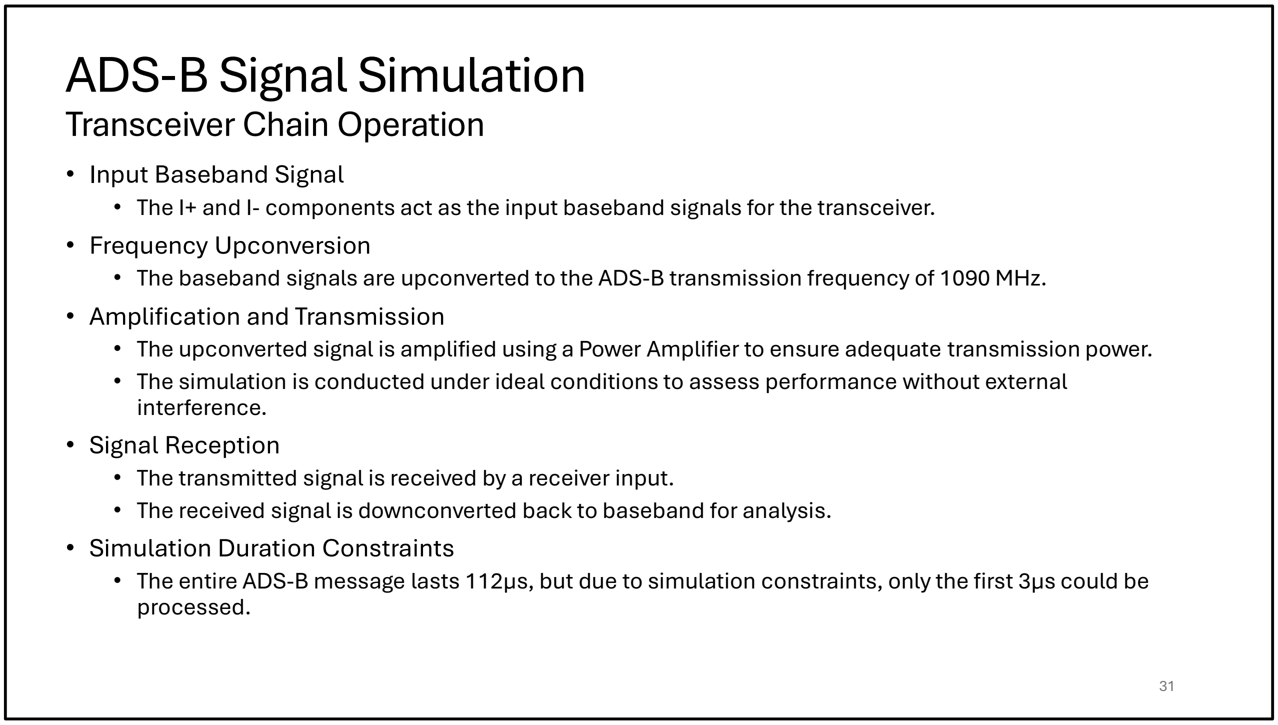
ADS-B Signal Simulation Setup Overview

- ADS-B Signal Generation
  - The ADS-B signal is generated in MATLAB with PPM encoding.
  - A flipped version of the ADS-B signal is created to represent the I- component.
- Signal Export and Preparation
  - The generated PPM signal is exported as time-voltage pairs in a text file for further processing. This will be the I+ component of the baseband signal.
  - The flipped version of the signal will be the I- component of the ADS-B signal.
- Signal Import:
  - Both I+ (original PPM) and I- (flipped PPM) signals are imported into Cadence Virtuoso.
  - Utilized '**vsource**' as Piece-Wise Linear (PWL) input to accurately represent the signals.



30

- Summarizing the work till now, the process begins in MATLAB with the generation of ADS-B signals using Pulse Position Modulation (PPM) encoding.
- The original PPM signal is designated as the I+ component, while its flipped counterpart becomes the I- component.
- Both signals are exported from MATLAB as time-voltage pairs in text files.
- In Cadence, we import both the I+ and I- signals using voltage sources (vsource) configured as Piece-Wise Linear (PWL) inputs.
- The signals are now ready to be simulated through the transceiver chain.



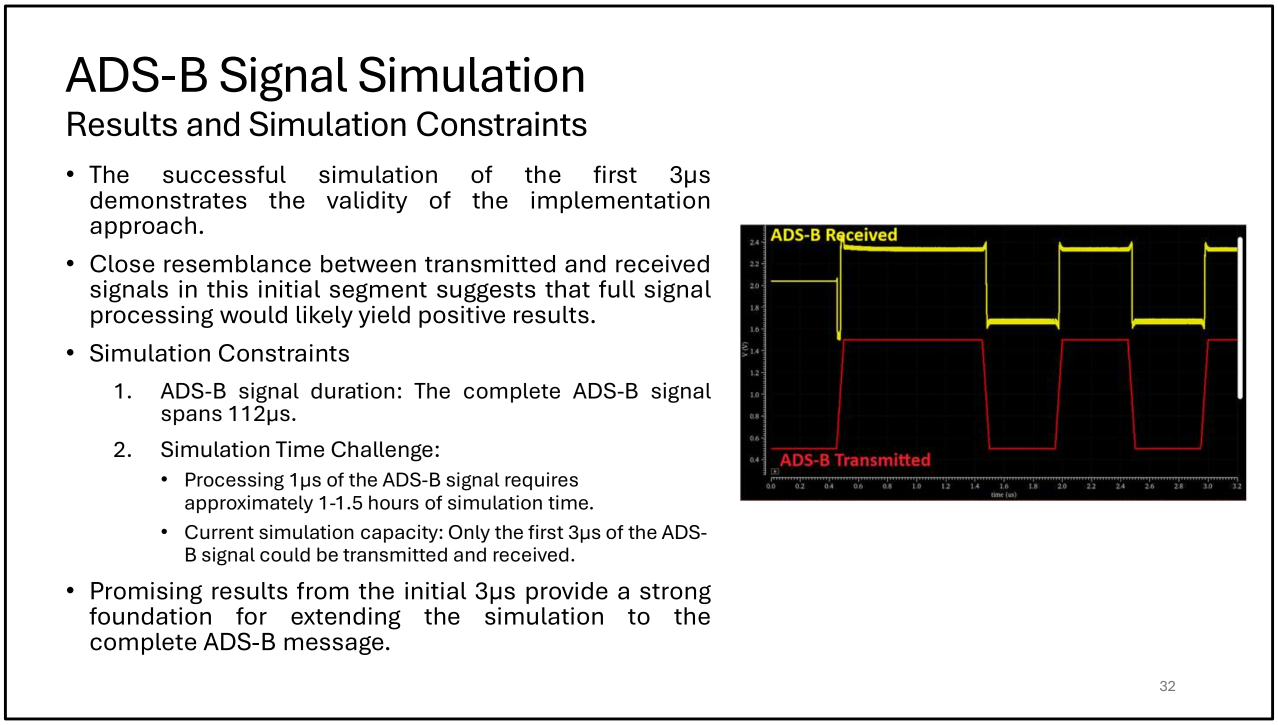
ADS-B Signal Simulation

Transceiver Chain Operation

- Input Baseband Signal
  - The I+ and I- components act as the input baseband signals for the transceiver.
- Frequency Upconversion
  - The baseband signals are upconverted to the ADS-B transmission frequency of 1090 MHz.
- Amplification and Transmission
  - The upconverted signal is amplified using a Power Amplifier to ensure adequate transmission power.
  - The simulation is conducted under ideal conditions to assess performance without external interference.
- Signal Reception
  - The transmitted signal is received by a receiver input.
  - The received signal is downconverted back to baseband for analysis.
- Simulation Duration Constraints
  - The entire ADS-B message lasts 112μs, but due to simulation constraints, only the first 3μs could be processed.

31

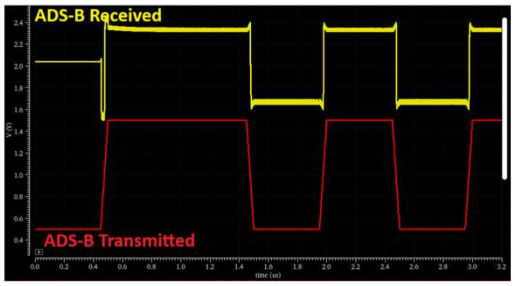
- The baseband signals undergo frequency upconversion to the ADS-B transmission frequency of 1090 MHz..
- On the receiving end, The received signal is then downconverted back to baseband.
- While a complete ADS-B message spans 112μs, simulation time constraints limit processing to the first 3μs for now.



ADS-B Signal Simulation

Results and Simulation Constraints

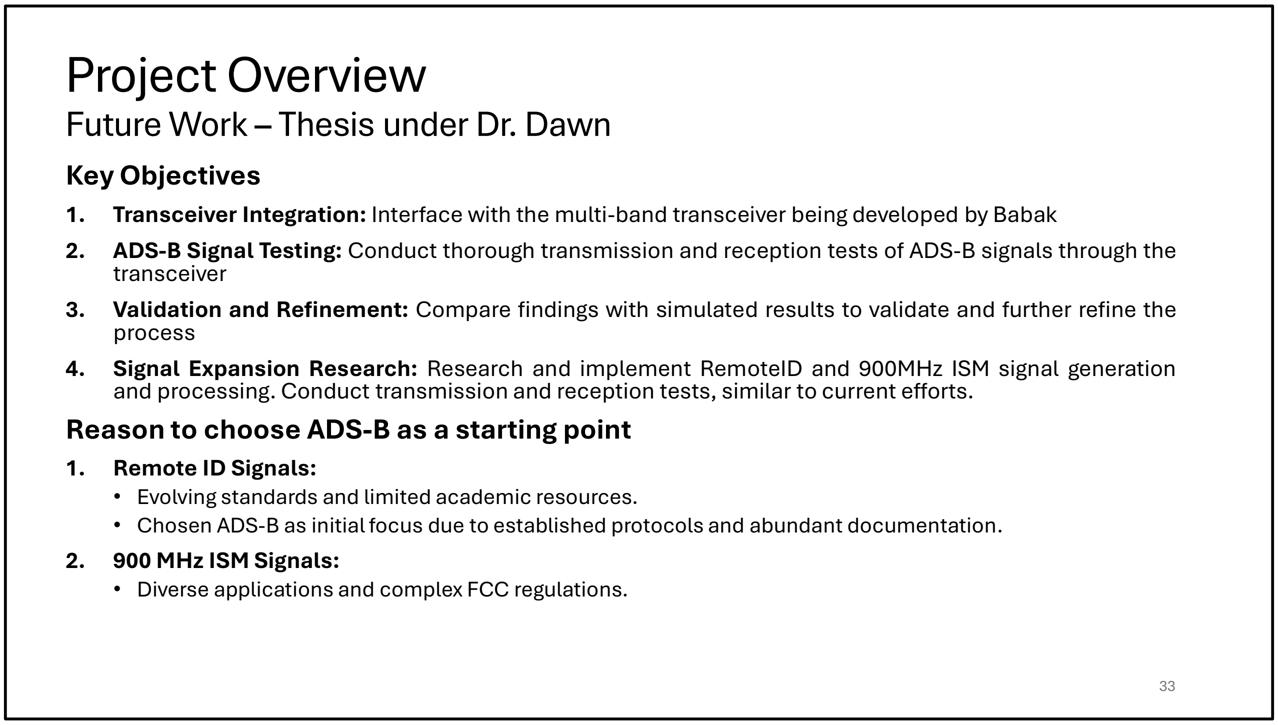
- The successful simulation of the first 3μs demonstrates the validity of the implementation approach.
- Close resemblance between transmitted and received signals in this initial segment suggests that full signal processing would likely yield positive results.
- Simulation Constraints
  1. ADS-B signal duration: The complete ADS-B signal spans 112μs.
  2. Simulation Time Challenge:
     - Processing 1μs of the ADS-B signal requires approximately 1-1.5 hours of simulation time.
     - Current simulation capacity: Only the first 3μs of the ADS-B signal could be transmitted and received.
- Promising results from the initial 3μs provide a strong foundation for extending the simulation to the complete ADS-B message.



32

- The successful simulation of the first 3μs of the ADS-B signal validates the implementation approach.
- The received signal clearly replicates the PPM encoding of the transmitted signal, indicating that with a full 112μs simulation, we could effectively decode the PPM bits and reconstruct the original ADS-B message.
- However, we currently face simulation constraints due to the complexity of processing ADS-B signals through the entire transceiver chain.

- Babak can explain in a more detailed manner about this behavior of the transceiver chain.



Project Overview

Future Work – Thesis under Dr. Dawn

Key Objectives

1. **Transceiver Integration:** Interface with the multi-band transceiver being developed by Babak
2. **ADS-B Signal Testing:** Conduct thorough transmission and reception tests of ADS-B signals through the transceiver
3. **Validation and Refinement:** Compare findings with simulated results to validate and further refine the process
4. **Signal Expansion Research:** Research and implement RemoteID and 900MHz ISM signal generation and processing. Conduct transmission and reception tests, similar to current efforts.

Reason to choose ADS-B as a starting point

1. **Remote ID Signals:**
   - Evolving standards and limited academic resources.
   - Chosen ADS-B as initial focus due to established protocols and abundant documentation.
2. **900 MHz ISM Signals:**
   - Diverse applications and complex FCC regulations.

33

- The key objective moving forward is to integrate our ADS-B signal processing with the multi-band transceiver being developed.
- We have a strong groundwork to compare the transmission and reception from the transceiver IC to the simulations.
- I chose ADS-B as our starting point due to its well-established protocols and abundant documentation.
- Future work will expand into RemoteID and 900MHz ISM signal generation and processing

- The choice to focus initially on ADS-B over RemoteID was mainly because of the evolving nature of RemoteID standards and limited academic resources in this area.
- Similarly, the complexity of 900 MHz ISM signals, with their diverse applications and intricate FCC regulations, makes them a natural next step for me.