Automatic Dependent Surveillance Band (Virtual Radar)

Presented by IEEE Signal Processing Society

IEEE Signal Processing Society
UNIVERSITY OF FLORIDA CHAPTER

ADS-Band Overview

The Automatic Dependent Surveillance Band consists of two different specific frequencies 1090 MHz and 978 MHz.

1090 MHz is the global ADS-B/S-Mode link used by majority of commercial and high-altitude aircraft while 978 MHz Universal Access Transceiver is a U.S. ADS-B link for general aviation below 18,000 ft with free weather and traffic. The UAT was created by the FAA to relieve congestion on the 1090 MHz band.

Why we want 1090 MHz specifically:

It's the global channel for ADS-B output on commercial, business, and high-altitude traffic, and more importantly it also carries S-Mode transponders replies. Especially since it's used globally across controlled airspace, airports, airlines, business jets, air-to-air, air-to-ground, as well as satellite reception. This gives a range of outputs that can we tap into to make our basic radar system.

How Do We Start This Pipeline: Hardware

We Need 3 Things:

- Antenna
- SDR
- Laptop

Not every antenna or SDR dongle will work the hardware needs to match the specific signal processing application.

So what we know is, we want to tap into 1090 MHz frequencies.

How Do We Start This Pipeline: Hardware (SDR)

What's important is picking a SDR (Software Defined Radio) that will be strong enough to sample 1090 MHz effectively. If we want to avoid aliasing information, we have to make sure our SDR can sample twice the bandwidth of the signal therefore meeting Nyquist Shannon Theorem (2xβ). The bandwidth of 1090 MHz is about 2 MHz so we have to sample at 4 Msps to avoid aliasing, filter-roll off, and any frequency error. Below are the respective sampling limits of two SDRs.

SDR Dongle: 3.2 Msps

VS

USRP B210: 61.44 Msps

How Do We Start This Pipeline: Hardware (Antenna)

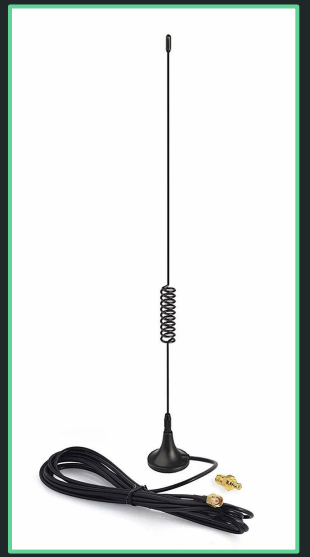
The next important thing, while choosing hardware is that we need to choose a specific antenna that is specifically designed for the type of wavelength that comes from the 1090 MHz spectrum.

The wavelength for this signal is 27.5 cm so we would need to purchase an antenna fits the physical dimensions and element spacing that are set as fractions of the signal's wavelength, so resonance (e.g., quarter-wave), impedance match, bandwidth, and radiation pattern all scale with **λ**.

λ = (3 x 10^8 m/s) / (1090 x 10^6 Hz) = 27.5 cm

$$\text{Freq} = \frac{C}{\lambda} \qquad \lambda = \frac{C}{\text{Freq}}$$

Needed Antenna:



Collecting Samples: Antenna Positioning

When collecting samples we have to position our antenna in the most optimal location to boost signal as much as possible:

With the equation for log-distance path loss model, with antenna positioning we can change γ if we have free space γ = 2 if the antenna is indoors then γ = 3-4. We want to lower γ because loss grows more slowly with distance, which is why performance is better outside. So in turn, lowering our γ will actually boost our signal which will result in better reception.

Model [edit]

Log-distance path loss model is formally expressed as:

$$L = L_{\text{Tx}} - L_{\text{Rx}} = L_0 + 10\gamma \log_{10} \frac{d}{d_0} + X_{\text{g}}$$

How Do We Start This Pipeline: Software

Programs:

- GNURadio - Python Language Based
- Simulink - Matlab Language Based
- 3rd Party Decoders

Which One to Use:

For engineers in college, companies like Lockheed Martin, L3Harris, RTX, and etc want to see you develop a whole signal chain or pipeline with a coding language you used in school but using dump1090 or other third party decoders shows no or less skill.

Experience:

Excellent potential project for radar, DSP or communication systems engineering especially for electrical engineering major.

MATLAB® & SIMULINK®

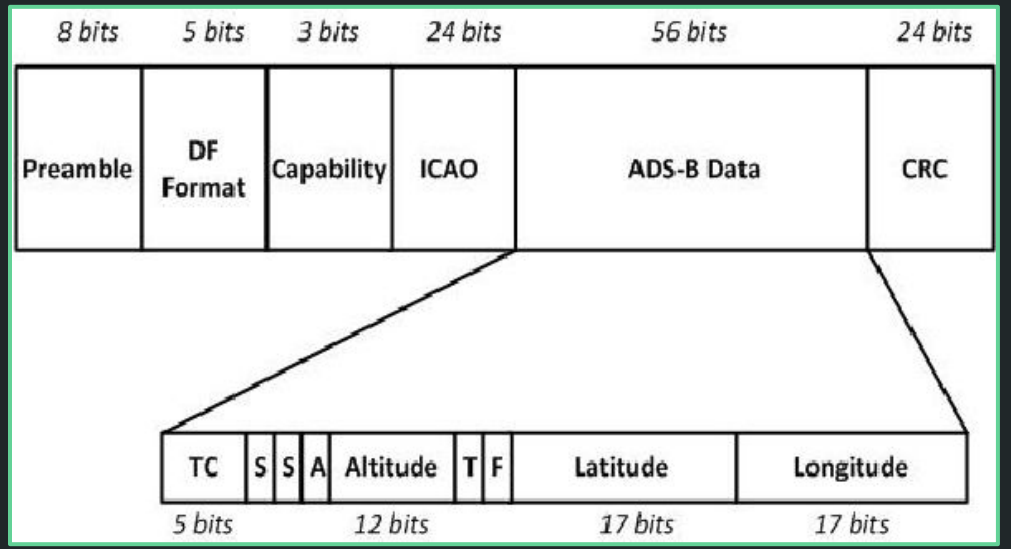
GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

S-Mode Frame Overview: What Are We Capturing?

We are capturing S-Mode Frames from ADS-Band. The entire frame is 120 µseconds long, this includes an 8 µsecond preamble followed by a 112 µsecond payload.

ADS-Mode S Extended Squitter Frame:

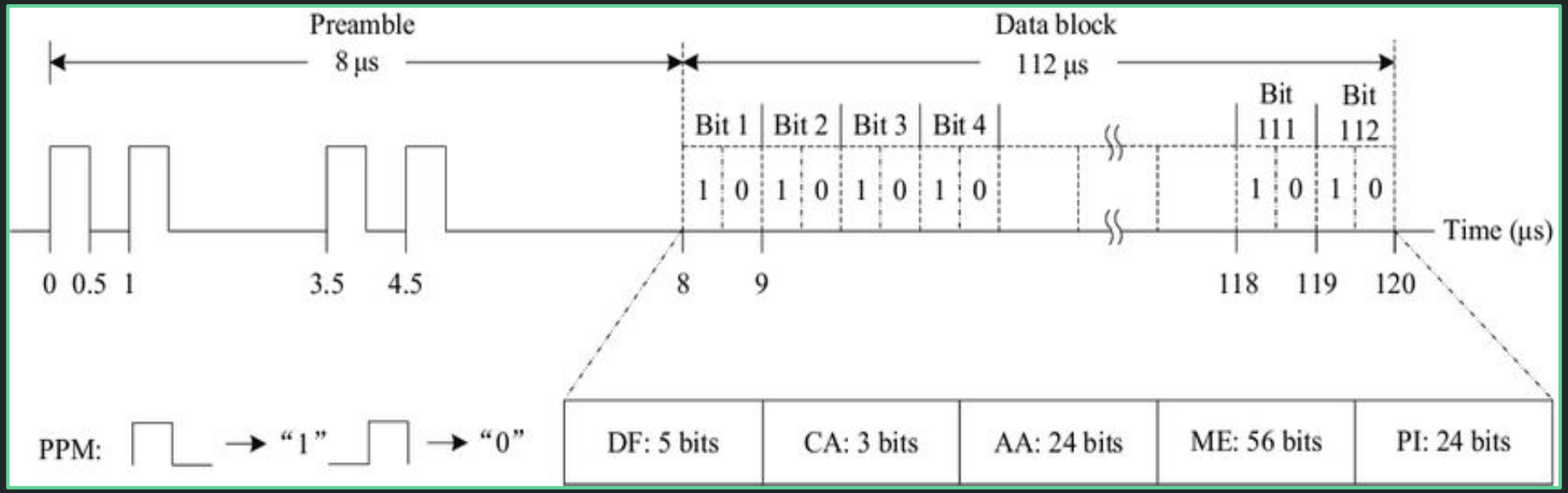
We are specifically looking to grab DF17 (Downlink Format) S Mode Extended Squitter Frames because these frames are released typically by transponder equipped civil aircraft, and based on a specific type code we can retrieve the latitude, longitude, and unique identifier address we're looking for.



S-Mode Frame Overview: S Mode Message Structure

The next couple slides we're going to dive deeper in each specific part of the frame and their variations, so we can understand what're capturing and where to look:

Preamble | Downlink Format | Transponder Capability | ICAO Aircraft Address | Message | Typecode | Parity/Interrogator ID



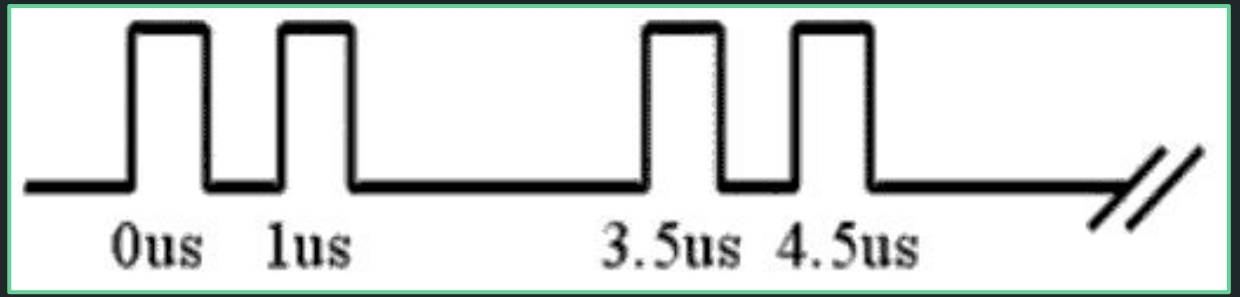
S-Mode Frame Overview: Finding the Preamble

The total length of the preamble is 8 µseconds which presents itself with bits of 0 or 1 every 0.5 µsecond.

Preamble Bit Array (Sampled at 2 Mhz):

[1,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0]

8 µseconds total



This is what we're looking for in our baseband complex IQ source file or stream so we can successfully lock the timing of a S Mode Extended Squitter Frame. Also as you sample more, the preamble gets bigger but bitwise stays the same.

Preamble Bit Array (Sampled at 4 Mhz):
[1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

S-Mode Frame Overview: Downlink Format

DF (0-31) is what determines the structure of the payload.

Here are all the most relevant downlink formats, anything number not on here is reserved for future use or other specialized use cases. We want to filter through the frames we collect through DF17 Extended Squitter Transponder Frames so we can collect the information we need for our basic virtual radar.

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)

S-Mode Frame Overview: Transponder Capability

Transponder Capability (CA) is a 3-bit value (0–7) in Mode S messages that declares which Mode S features the transponder supports— e.g., basic surveillance, ACAS/TCAS, Comm-B, extended squitter—so interrogators/decoders know what services to expect and how to deal with it.

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

S-Mode Frame Overview: ICAO Address & Message Type Code

In each ADS-B message, the ICAO Aircraft Address (24-bit) is a international unique identifier number that is given to an airplane transponder, used to identify aircraft and the **Message Type Code (5-bit)** is the start of DF17/DF18 extended squitter message payload that shows what the content of the ADS-B message and this gives us information on how to decode the rest of the frame properly. To retrieve latitude and longitude of airborne aircraft coordinates we want to look for and isolate **Type Codes 9-18** specifically.

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

S-Mode Frame Overview: Availability & Transmission Rate

Varying ADS-messages have different transmission rates which relates to type code. The frequency updates based on aircraft being on the ground, airborne, and even if they're still moving when on the ground. This is something that has to be taken into consideration when demodulating and decoding these messages. This is important for pairing messages for our decoding process that we're going to be using which help us to know how long to buffer and give up.

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*)	

S-Mode Frame Overview: Parity/Interrogator ID

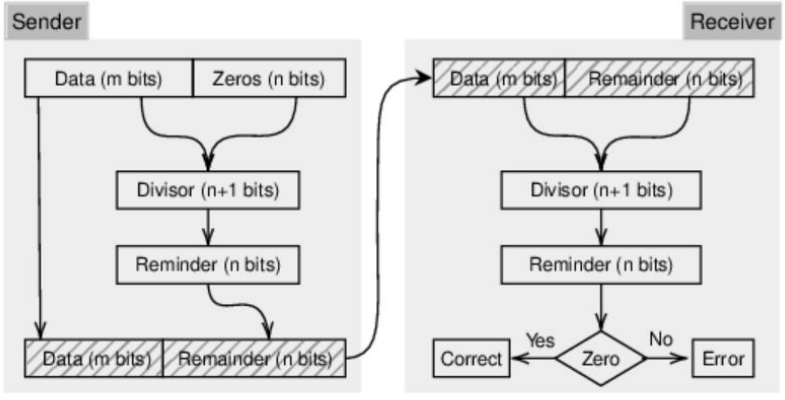
The final bits of the ADS-B message, are allocated for Mode S Parity (PI) which is essential for error control. This is important because if we have a high bit error rate a cyclic redundancy check (CRC) based on the 24 bits located on the last part of the frame if checked by CRC will lead to a checksum failure. This way we can control the quality of a ADS-B message and only validate ADS-B frames that pass this CRC check. This is even more important when you realize that air traffic controllers actively respond to these types of message so error control is essential to maintain message integrity from flipped bits.

For DF17, the parity is “address-annexed” (PI = CRC ⊕ ICAO address) so it detects errors and carries the address and DF18 uses typically uses non-address parity. P(x) is the remainder left, which is the CRC.

Polynomial CRC Bit Math:

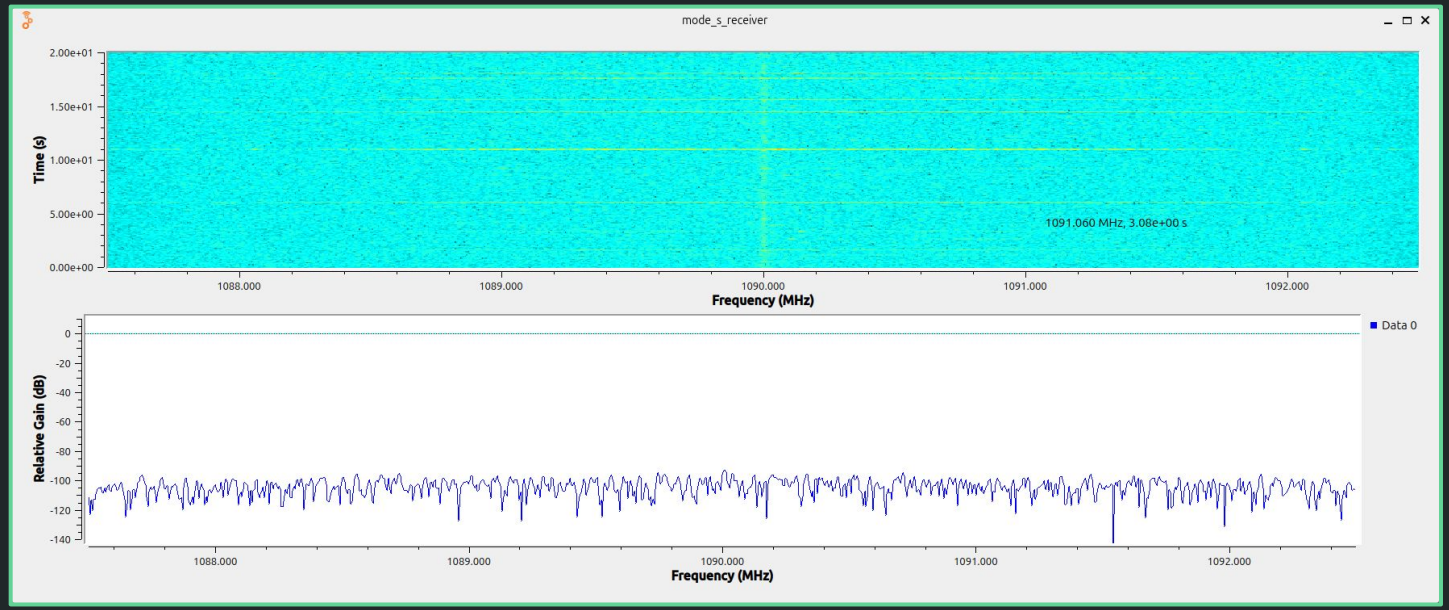
$$M(x) = \sum_{i=0}^{87} a_i x^i, \quad a_i \in (0,1)$$
$$P(x) = M(x) \,\%\, G(x)$$

$$G(x) = x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^3 + 1$$



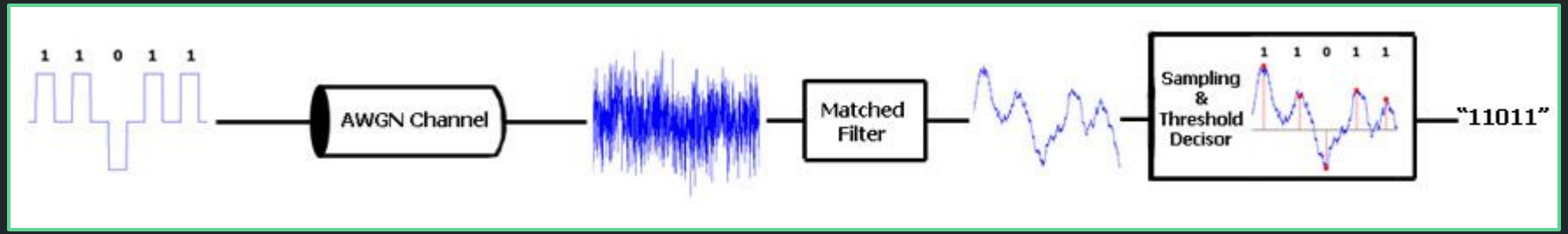
Pipeline Development: Data Collection

So now that we know and understand what we're collecting, we have to retrieve the information with the highest signal to noise ratio (SNR) possible so we can achieve the lowest bit error rate (BER) so we can get these S Mode Extended Squitter DF17 Frames past cyclic redundancy checks (CRC) or checksum failure. Validating these frames successfully starts with the retrieval of good sample collection combined with the proper filtering. We start by sampling data At 4 Msps and configuring gain to optimize SNR using the USRP B210 Software Defined Radio (SDR) and then move on to matched filtering to optimize SNR at a greater level.



Pipeline Development: Matched Filtering

When collecting complex baseband signal samples from the RF spectrum, we typically encounter additive white gaussian or normal noise (AWGN) which tends to lower our signal to noise ratio dramatically causing flipped bits or high bit error rate. So a great technique that is also used in radar systems is matched filtering or cross correlation to build a filter specifically optimized for mitigating AWGN as much as possible. Maybe you remember cross correlation from DSP, but in this application a matched filter is the cross-correlation of the received signal with the time-reversed, complex conjugated template, under AWGN it maximizes the SNR at the decision instant which in turns yields a sharp detection peak.



Pipeline Development: Throttle/Preamble Detection

Throttle:

After the signal to noise ratio is optimized we want to start finding the frames in the signal. Before we do that we have to throttle the signal source so the in the pipeline there is a steady flow of frames because keep in mind there can be dozens if not hundreds of transponders S-mode frame messages within your area.

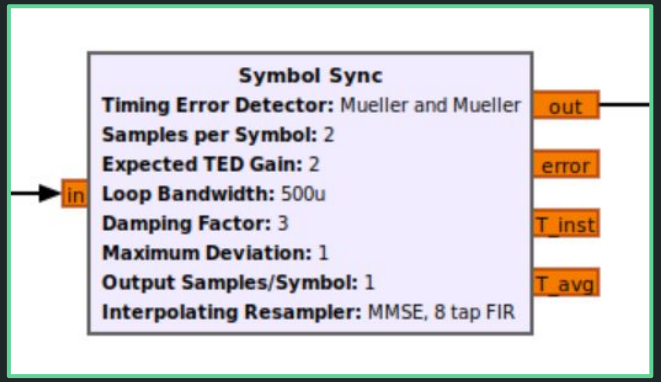
Preamble Detection:

Within customized blocks in GNURadio, using a Complex to Mag^2 block before matched filter we can take the slow stream output from the matched filter and then with S-Mode preamble template of 8 µseconds we can apply a threshold to pick t0 peaks, which will allows us to be able to lock our 1 Mbps symbol timing.

Demodulation: 1 Mbps Symbol Synchronization

After we have developed a pipeline to have preamble detection with a threshold produce t0 that we can lock in for the 1 Mbps Symbol Timing needed for demodulation. One symbol is 1 µsecond and in that each 0.5 µsecond a 0 or 1 is encoded. The 8 µsecond preamble is for synchronization and isn't counted as symbols; after the preamble you have 56 symbols for shorter frames and 112 symbols for long frames.

So, we detect the 8 µsecond preamble to get t0; Compute samples-per-symbol which is Sps = Fs/1e6, and then we align 1 µsecond symbol boundaries all over the frame. The 8 µsecond preamble isn't counted, then 56 or 112 symbols follow. After locking, then we can make bit decisions to produce 0/1 bit stream.

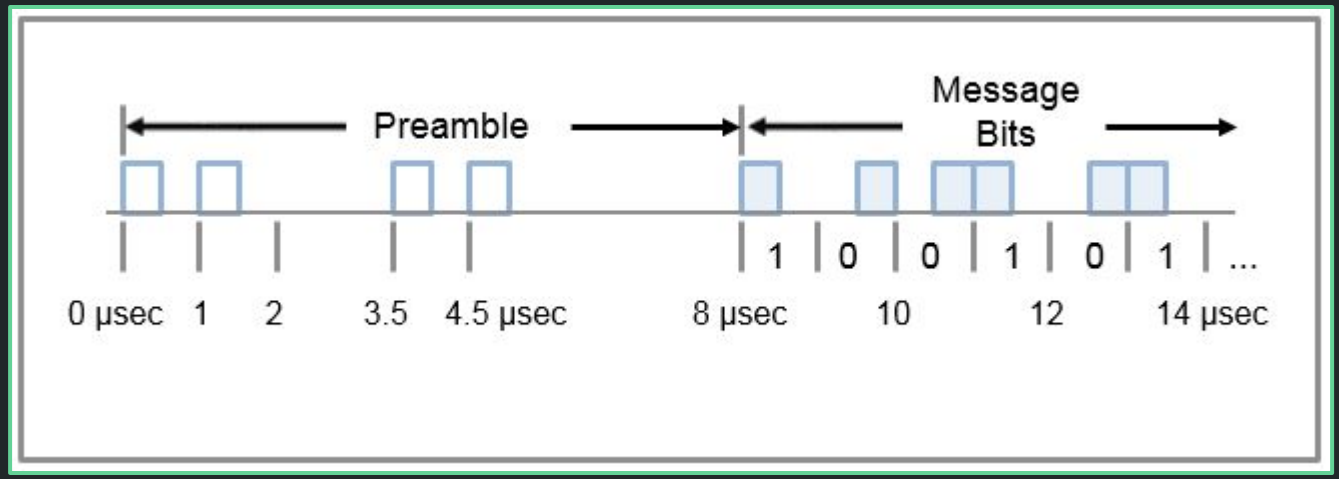


Fs is the sampling rate.

t0 is estimated start-of-frame time obtained from preamble correlation peak.

Demodulation: PPM Bit Decisions at 4 Msps

Since we sampled at 4 Msps, each 1 µsecond symbol is technically 4 samples. So after the 8 µsecond, split each symbol into an early half (meaning first 2 samples) and a late half (next 2). Do the sum of mag^2 in each half; if the larger sum exceeds the threshold decide 1 if early half > late and 0 if late > early, if below threshold none of these mark as erasure. This per-symbol split decision rule will yield the 56/112-bit stream that is necessary for decoding.



Decoding Process: CRC Pass/Fail

Input: N = 56 or 112 bits, MSB-first as received. Data = bits 0..N-25, Parity PI = bits N-24..N-1.

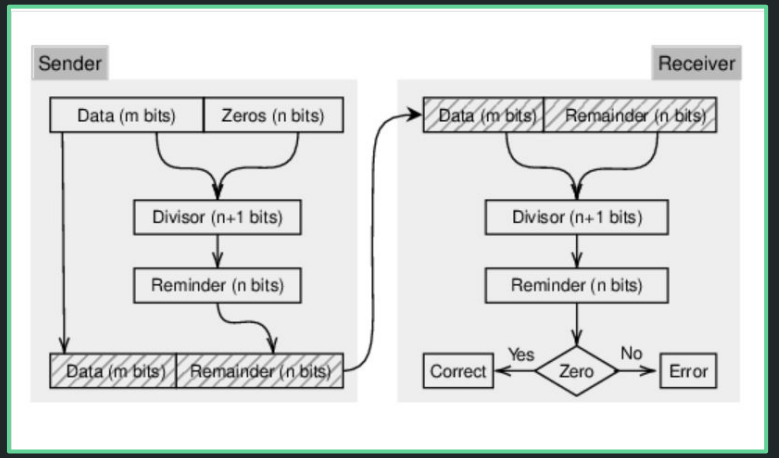
Compute the Mode S CRC-24 remainder R over the data bits (MSB-first). Implement either:

- Bitmask table: precompute a 24-bit mask per bit position; start R=0 and XOR R with the mask for each data bit that is 1.
- Or polynomial division: shift/XOR with the Mode S generator, MSB-first, to get a 24-bit remainder.

Interpret:

- Non-addressed Squitters (e.g., DF17/DF18): Pass if R == PI.
- Addressed replies (e.g., DF11 and most interrogated DFs):
- PI = R XOR ICAO; recover ICAO as AA = R ⊕ PI

If pass, continue to DF/Type Code parsing; if fail, drop the frame.



Decoding Process: DF/TC Frame Parsing

Assuming S-Mode frames are passed by CRC checks:

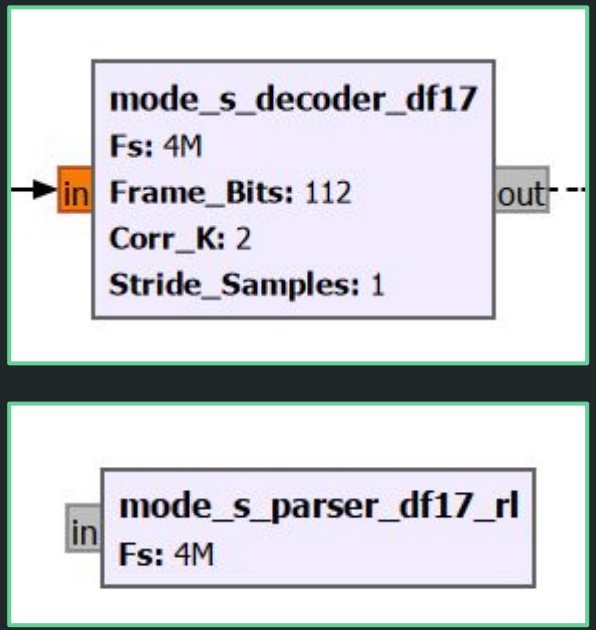
We can extract the bits sort each 112 bit frame into bit sections, DF (1-5), CA (6-8), ICAO Aircraft Address (9-32), Type Code (33-37), ME(33-88), PI (89-112).

Next, Isolate Frames that Contain:

DF (Bit Value) = 17 for S-Mode Transponder Extended Squitter

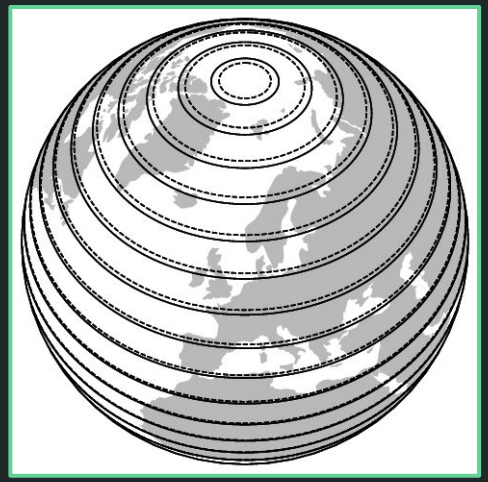
TC = 9-18 for Airborne Position

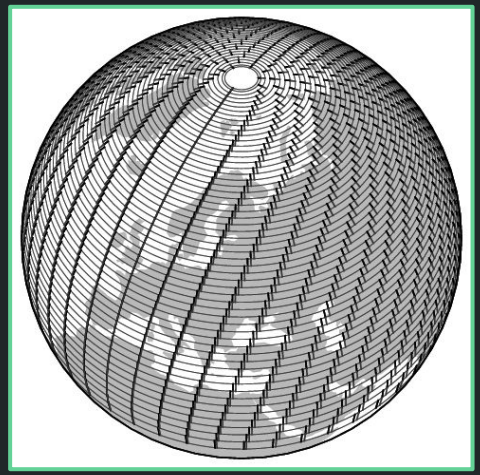
This parsing can all be done in custom python blocks in GNURadio.



Decoding Process: Compact Position Reporting (CPR)

After Isolating S-Mode Frames (DF= 17 & TC= 9-18):
So CPR packs latitude and longitude coordinates into 17-bit fractions on two different grids, one even and one odd. Each DF17 airborne frame carries x (lon) and y (lat) for it's grid plus a parity bit (even/odd). You cache the last even and odd per ICAO; when you receive both within 10 seconds.





Conceptually: Two interleaved grids plus a pair of frames lets you pinpoint what cell an aircraft is in, giving a global latitude and longitude coords.

Data Retrieval: Civilian Aircraft Virtual Radar

After a CRC-passed even/odd pair for the same ICAO, compute global lat/lon using the newer frame's parity; accept if pair age ≤ ~10 s and NL(lat) ≥ 1 (if the newer frame is odd, require NL(lat) > 1 for longitude), then update the aircraft map (lat, lon, altitude, timestamp); otherwise discard.

NL(lat) = Count of Longitude Zone at that Latitude; Sets Longitude Width in CPR

Software Setup QR Code (For Part 2)

Windows & Mac Compatible

ANACONDA®

GNURadio

THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM