

**RTCA Special Committee 186, Working Group 5**

**ADS-B UAT MOPS**

**Meeting #7**

**Draft 1 of Appendix H:  
UAT Synchronization Issues**

**Prepared by Warren J. Wilson  
The MITRE Corp.**

SUMMARY
<p>This paper is a draft of Appendix H, which provides guidance on issues related to the UAT synchronization process. It combines the information previously contained in UAT-WP-4-12 and UAT-WP-5-11A.</p>

## H-1 Introduction

This Appendix examines some issues related to the UAT burst synchronization process. The primary purpose of any synchronization process (including that of UAT) is to detect the presence of a signal and to determine as accurately as possible the correct timing of the signal so that it can be successfully demodulated.

For UAT, synchronization is supported by a 36-bit sequence that occurs at the beginning of every burst. A simplified version of the synchronization process might consist of comparing samples of the incoming bit stream with the expected sequence. If more than a certain number (say, 31) of these bits were correct, a valid signal would be considered detected. The length of the sequence, the particular sequence used, and the threshold are chosen so that there is a reasonably low probability of false alarm (“synchronizing” with no valid sequence actually present or with the timing incorrect by more than one bit period). In order to insure that the synchronization process determines the bit sampling time to within a small fraction of a bit period (to optimize performance), the waveform is typically sampled more than once per bit during the synchronization process. In the discussion below, we will assume that the sampling rate is six times the bit rate (i.e., 6.25 Msps).

Synchronization for the UAT ADS-B messages is complicated by the fact that they are transmitted pseudorandomly by each user and there is, therefore, a high probability of signal overlap. Overlap is not necessarily an overwhelming problem because UAT is fairly robust in the presence of self-interference. This is primarily due to the fact that the waveform is basically binary FM. Very good performance results when the desired-to-undesired ratio (D/U) is as little as 6 dB (with a single interferer). However, this property could be rendered irrelevant if receivers which have already synchronized to weak signals cannot resynchronize (or “retrigger”) on strong signals arriving slightly later. Thus, good performance in the presence of a heavy self-interference environment is dependent upon the ability to continue to search for synchronization while demodulating a signal. If a new synchronization correlation is found during the course of demodulating a burst, the receiver can either switch from processing the old signal to the new one, or it can attempt to demodulate both. There is some danger in the switching option because the sequence used for synchronization (or something very close to it) may be embedded in a valid message. In the next section we will show an example of how to implement the option of demodulating both.

*Caveat:* The discussions in this Appendix pertain to a particular implementation of a transceiver. Other techniques may also be viable.

## H-2. Synchronization Process Description

Figure H-1 is a top-level diagram of a possible UAT synchronization scheme. It shows that after the received signal passes through an IF filter, it is detected by a limiter/discriminator that generates a string of ones and zeros at a rate of 6 times per symbol. The sample stream then goes through a correlator, which is looking for the

synchronization pattern. As the samples stream out of the correlator, they enter a 6 by N byte memory. This is arranged so that each “row” of the memory contains samples separated by the bit period. In the meantime, whenever the box labeled “threshold” determines that a signal is present, it will attach a flag to the appropriate location in the memory. The processor will then read out bits from the memory (on a row-by-row basis) according to the locations of the flags. This read-out process will allow the processor to deal with overlapping messages, whether or not they are in the same row of the memory. Note that *all* samples follow this path, even when “information” is being processed. Thus, the receiver can continuously look for new synchronization patterns.

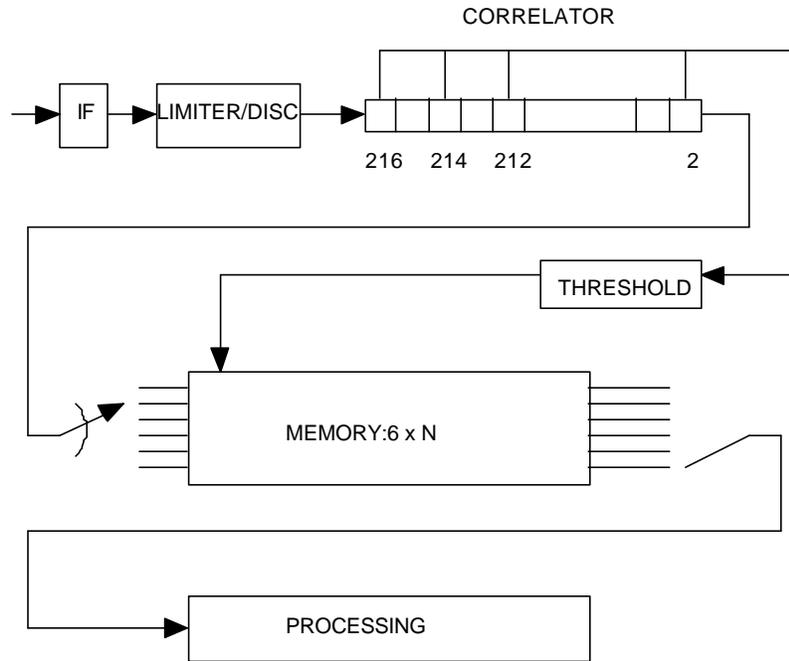


Figure H-1: UAT Receiver Processing

In the subparagraphs below, some of the components of Figure H-1 are dealt with in a little more detail.

The **correlator** is basically a tapped delay line whose length is long enough to hold about  $36 \times 6 = 216$  samples. The taps are arranged so that every other location is used to compare the incoming sample sequence with the fixed synchronization sequence. The expected sequence consists of triplets of ones and zeros. The output of the taps is a sequence of correlation scores that can range anywhere from 0 to 108. A score of 108 indicates that every sample is correct. A value of zero (0) indicates that every sample is incorrect. Normally, this latter case would only occur if a ground message were present since the ground message synchronization sequence is just the “opposite” of the ADS-B sequence. These correlation scores are fed at a rate of 6.25 million scores per second to the threshold detector (see below), and the bits exiting the back end of the delay line are fed at a rate of 6.25 Mbps into the box labeled “memory.”

The **threshold detector** compares the sequence of correlation samples and looks for values at or above a threshold. In order to find the center (or optimum sampling point) of each bit, the threshold detector will, when confronted with a string of successive threshold crossings, choose the highest one of the group. Having chosen a potential ADS-B burst start point, the threshold detector attaches a marker to the appropriate location in the “memory” (see below), e.g., the first bit sample of the information portion of the message. This process runs continuously.

The **6xN memory** accepts bit samples from the correlator at a rate of 6.25 Msps and arranges them into six separate sequences. Each separate sequence represents a separate string of potential information bits. Those bit strings flagged by the threshold detector are passed to the box called “processing” to determine which comprise real bursts. Note that the receiver has, at this point, no way to determine whether a threshold crossing corresponds to a long or a short ADS-B message. This will be determined in the “processing” box. In the meantime, all ADS-B messages are treated as if they were long ones. At any given time there can be many potential burst starts indicated by pointers in the memory. These can be due to real messages or false alarms that may or may not overlap with other real messages or false alarms. These are all sorted out in the processor.

The **message processing** begins with each potential ADS-B message being read into a first-in-first-out (FIFO) memory based on the locations of the flags set by the threshold detector. As stated previously every message is temporarily assumed to be a long one. Each message, in turn, is then subjected to Reed-Solomon decoding. Those that are successfully decoded are assumed to be valid long ADS-B messages. Those that fail are then subjected to Reed-Solomon decoding assuming they are short ADS-B messages. Those that pass this test are assumed to be valid. Potential messages that fail to decode either way are discarded. Because of the very low undetected error probabilities of the Reed-Solomon codes employed by the ADS-B messages, this method of sorting through the potential messages should be extremely reliable.

Note that the ability to process multiple overlapping messages is limited only by the size of the 6xN memory and the ability of the Reed-Solomon decoder to process all potential messages. It appears that neither of these is a real problem. The memory requirement is very small (see section H-5), and the decoder chip in the UAT prototype can handle many times the maximum required decode rate.

### **H-3. Synchronization Performance**

The correlation process described above is designed to provide a high probability of detection, a low false alarm rate, and a reasonably accurate determination of the optimum sampling point for the information bits. To provide the necessary background, we will assume that the transmitted baseband signal has been generated by passing the bit sequence through a Nyquist filter prior to the FM modulation process. The purpose of the filter is to restrict the transmitted spectrum in order to minimize interference to and from other nearby systems (e.g., DME). The filter is assumed to be a raised-cosine with a

roll-off factor of 0.5. This is theoretically an infinite impulse response (IIR) filter, but it can be implemented using truncation at plus and minus 3 bit periods. The resulting filter shape is shown in Figure H-2.

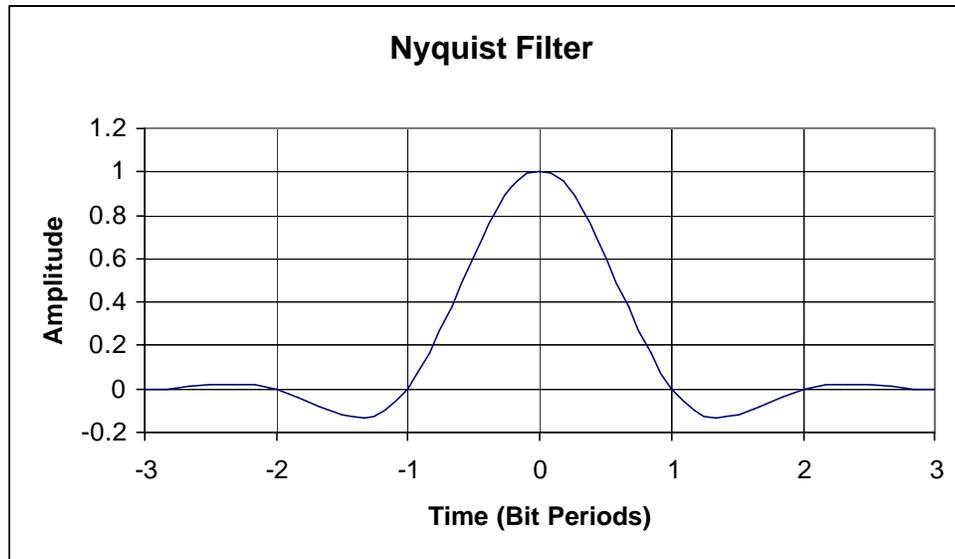


Figure H-2. Baseband Transmit Filter. Truncated Raised-Cosine Nyquist Filter With Roll-Off Factor = 0.5

When the synchronization sequence is passed through this filter the result is a fluctuating signal that can be used to modulate the transmitted frequency. The profile for the sequence is shown in Figure H-3. In Figure H-3, the synchronization sequence is represented so that a 1 is a shift up in frequency by  $F_0$  and a 0 is a shift down by  $F_0$ . For UAT, the value of  $F_0$  is 312.5 kHz. The correct frequency values for the synchronization sequence are achieved at the 36 integer values spanning 0 through 35. Due to the action of the filter, the instantaneous frequency smoothly varies from +1 to -1 at noninteger values. Sometimes the instantaneous frequency is actually larger than 1. Note that prior to the synchronization sequence (i.e., during ramp up), the waveform is assumed to be modulated with zeroes as specified in the MOPS.

A small portion of the sequence is shown in Figure H-4. This picture emphasizes that the frequency deviation is often reduced significantly between the ideal sampling points. This fact might seem to indicate that the most reliable performance would be available to a scheme that relies only on these robust bits (i.e., only use one sample per bit). This may be the case; but, as we will show below, such a method may not accurately determine the location of the waveform's optimum sampling point. To address this issue, the proposed method uses every other sample of 6 samples per bit period to generate a correlation score (number of correct bits) that can vary between 0 and 108 for any given measurement. The general idea behind this scheme is that when the central sample is indeed at the center of the bit period, the 2 "outrigger" samples will typically have the same polarity. On the other hand, if the central sample is off by as little as one sixth of a

bit period, some of the outrigger samples will have a high probability of being incorrect (refer to Figure H-4).

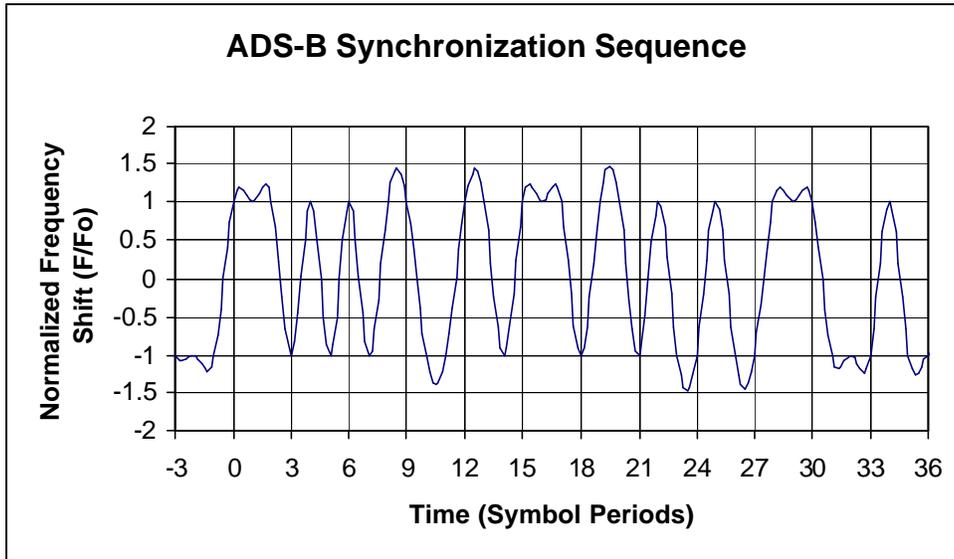


Figure H-3. ADS-B Synchronization Sequence

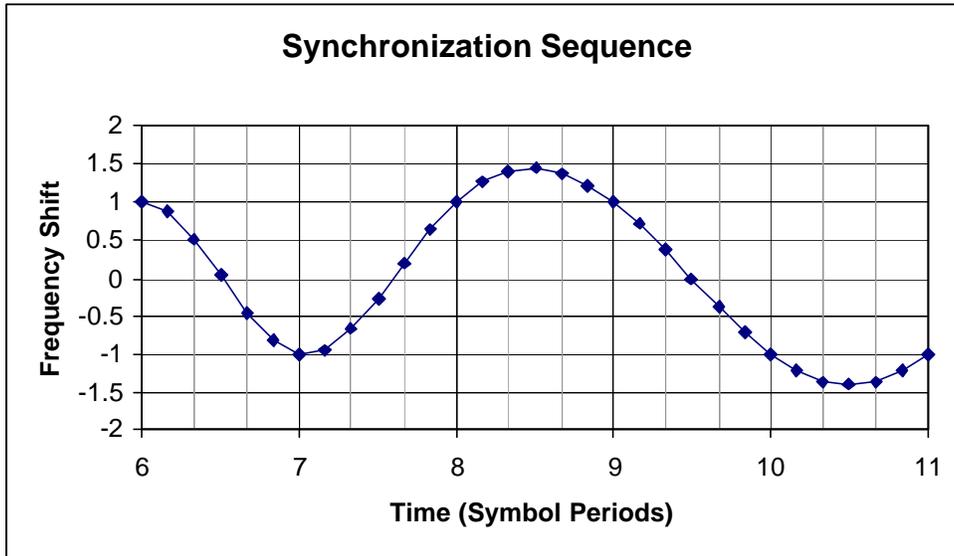


Figure H-4. A Portion of the ADS-B Synchronization Sequence

To analyze how this works we need to establish some mathematical notation. First, assume that the synchronization sequence is given by the vector

$$\vec{B} = (b_1, b_2, b_3, \dots, b_{36}).$$

Each component of this vector is +1 or -1 depending on whether the bit is a one or a zero. From this we can define a new vector,  $\vec{F}$ , whose components are all zero except that

$$F_{6i} = b_i.$$

We can now define a third vector that specifies the correlator tap weights:

$$T_i = F_i + F_{i+2} + F_{i+4}.$$

All the odd tap weights are zero (0). There are 108 nonzero tap weights (the even ones from 2 to 216).

If the incoming sample sequence at any given time (again, expressed as +1 or -1) is given the notation

$$S_i \quad \text{with} \quad i = 1 \text{ to } 216,$$

then, the correlation value is given by

$$Corr = \left( 108 + \sum_{n=1}^{108} T_{2n} S_{2n} \right) / 2.$$

This counts the number of correct samples out of a possible 108.

To estimate performance, we will assume that the incoming sample sequence is generated by a valid synchronization sequence (as portrayed in Figures H-3 and H-4) plus additive white Gaussian noise. (These are only estimates, certain approximations are inherent in this analysis.) In other words,

$$\begin{aligned} S_i &= +1 \quad \text{if} \quad x_i = \sum N_{ij} F_j + g_i \geq 0 \\ S_i &= -1 \quad \text{if} \quad x_i = \sum N_{ij} F_j + g_i < 0 \end{aligned}$$

where the  $N_{ij}$  are the Nyquist filter coefficients and the  $g_i$  are Gaussian noise samples. It can be shown that when noise is absent from the input sequence and there is perfect time alignment between the input and the expected sequence the correlation score will be 108. With finite noise and a possible offset, the expected correlation score can be approximated by

$$\langle Corr(\mathbf{g}, m) \rangle = 108 - \frac{1}{2} \sum_{i=1}^{108} \operatorname{erfc} \left( T_{2i} x_{2i+m} \sqrt{\frac{\mathbf{g}}{2}} \right)$$

where the value  $m$  is an integer that indicates the relative timing offset in units of sample periods.  $g$  is the signal-to-noise ratio (SNR). The results of this equation for various values of SNR and offset are shown in Figure H-5.

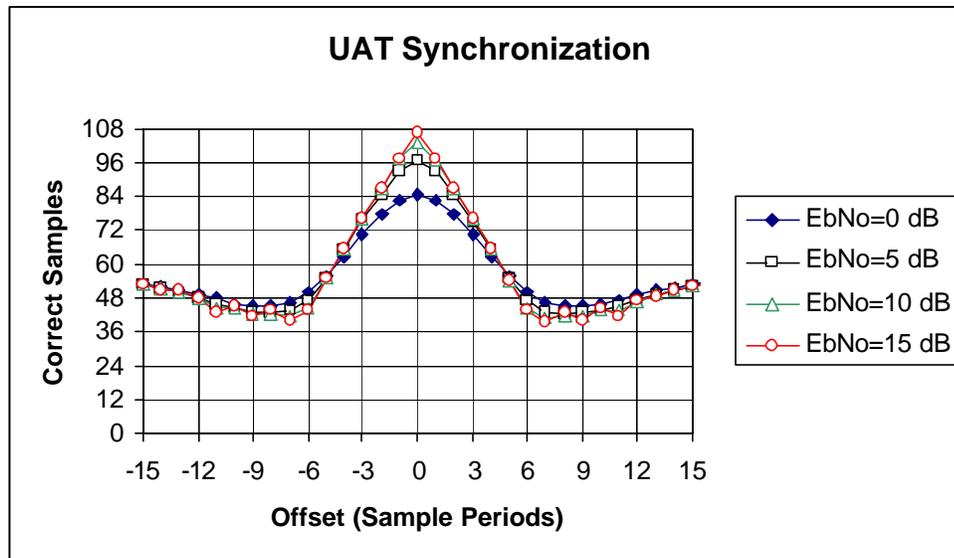


Figure H-5. Performance of the Three Sample per Bit Scheme

Note that the curves in Figure H-5 have some desirable features. In particular the curves are quite peaked whenever the SNR (or  $E_b/N_0$ ) is greater than 5 dB. This feature should make it comparatively easy to identify the sample time with the smallest offset. This performance can be compared with a scheme based on one sample per bit. In that case the tap weights would be given by

$$T_i = F_i$$

and, the correlation value would be given by

$$Corr = \left( 36 + \sum_{n=1}^{36} T_{6n} S_{6n} \right) / 2.$$

The expected correlation value is given by an equation similar to the one for the three-sample case. It is evaluated for various values of SNR and time offset in Figure H-6. In the one sample per bit scheme the correlation “peak” is actually a plateau, and at high SNR there is a very good likelihood that several samples will have the maximum possible value. The optimum sampling point will be very difficult to identify. This provides the rationale for the first scheme.

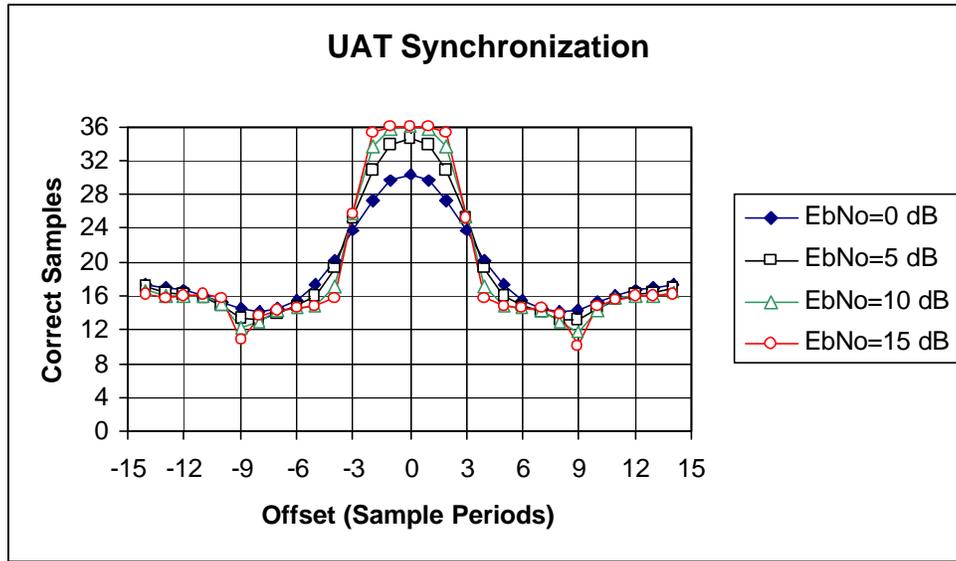


Figure H-6. Performance of the One Sample per Bit Scheme

#### H-4. False Alarm Rate

The false alarm rate is related to the probability that a random bit sequence (or simply noise) creates a correlation value that exceeds the synchronization threshold. A reasonable rate of false alarms can be tolerated provided that the receiver has a retrigger capability. If all of the 108 samples that are used to create the correlation value were statistically independent, then we could write the false alarm probability as

$$P_{fa} = (0.5)^{108} \sum_{n=0}^{108-T} \frac{108!}{n!(108-n)!}$$

where  $T$  is the synchronization threshold. In that case the false alarm rate (number per second) would be

$$FAR = 5 \times 10^6 P_{fa}$$

since the ADS-B portion of each second is 0.8 s and the samples are taken at a rate of 6.25 MHz. This method is clearly not correct because it fails to take into account the finite bandwidth involved in the overall synchronization process. The narrow IF filter limits the bandwidth, and the addition of three samples per bit acts as a crude version of a digital filter matched to the transmitter Nyquist filter. Neighboring estimates of the correlation score are *not* independent. An extreme manifestation of this effect would be to interpret the synchronization as being based on only 36 independent bit samples occurring once each bit period. This would mean that the false alarm probability would be given by

$$P_{fa} = (0.5)^{36} \sum_{n=0}^{36-T/3} \frac{36!}{n!(36-n)!}$$

and the FAR by

$$FAR = 833333 P_{fa} .$$

In actuality the truth probably lies somewhere between these two extremes. As an *ad hoc* guess, we can try the following:

$$P_{fa} = (0.5)^{NE} \sum_{n=0}^{NE-T(NE/108)} \frac{NE!}{n!(NE-n)!}$$

and

$$FAR = 2.5 \times 10^6 (NE / 108) P_{fa}$$

where  $NE$  is the effective number of bits. In Figure H-7 we plot the FAR versus  $T$  for three hypothetical values of  $NE$ . These curves we have plotted with measured data. The curve fits well if we assume the effective number of bits in the correlation is 72.

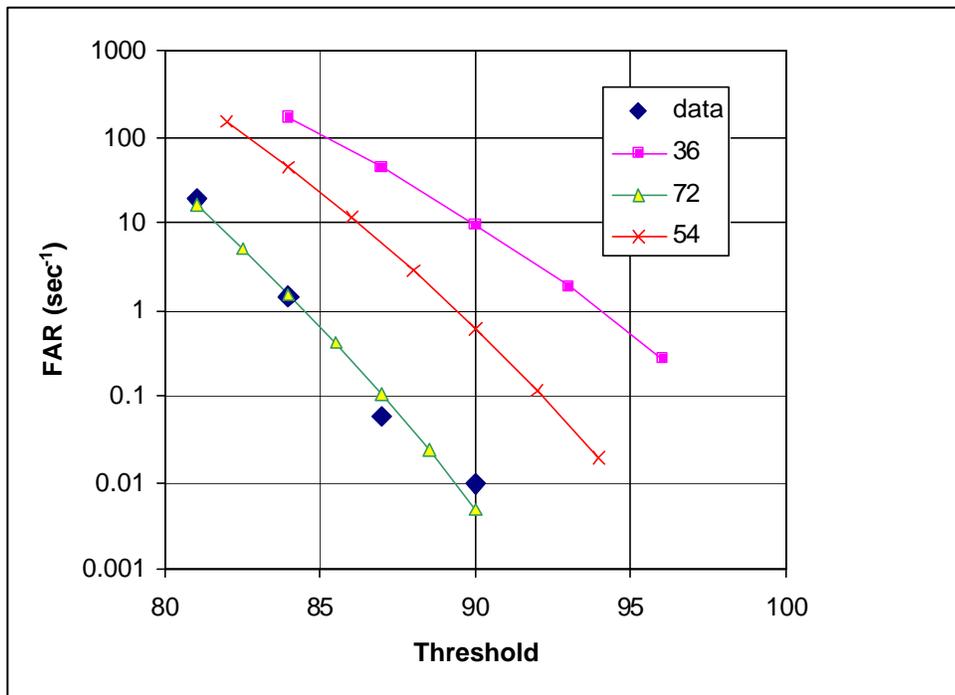


Figure H-7. False Alarm Rate versus Threshold  
(Parameter is Effective Number of Bits,  $NE$ )

The measurements indicate that the threshold, which was originally set at 94, could be lowered without generating a large number of false alarms per second. This threshold lowering could increase the robustness of the synchronization process, if desirable. However, it should be borne in mind that the measurements were made with an IF filter whose bandwidth was nominally 1.75 MHz. This filter may be replaced by a narrower one. A narrower filter may result in lowering the value of  $NE$ . If that is the case, and if the analysis presented here is correct, the FAR might increase somewhat. Therefore, the FAR should be re-measured with the final filter in place to determine if the threshold can be safely lowered.

### **H-5. Required Memory Size**

The block labeled “Memory:  $6xN$ ” is provided to allow the receiver to sort out numerous (possibly overlapping) received signals and feed potentially valid bit streams into the “processing” block in the form of a sequence of long ADS-B RS code words. This block is essentially an elaborate deinterleaver. It is assumed that the bits are fed into the memory at a rate of 6.25 Mbps and can be read out at a rate of 6.25 Mbps at the same time. In other words, it must be capable of reading in one bit and reading out one bit during the same clock cycle (160 nanosecond). The question addressed here is the required size of the memory, or “What should  $N$  be?”

To answer that question, a simulation was created which generated 2200 ADS-B message starts each second. This corresponds to the worst-case loading scenario. These messages were then sorted according to time-of-arrival. It was then assumed that every one of these messages would result in a crossing of the synchronization threshold, i.e., every message would need to pass through to the processing block. That is actually an unrealistic assumption since many of the synchronization attempts may fail due to self-interference. Thus, this analysis will provide an overestimate of the memory requirement.

The simulation operated by marching along in time and noting each message start. When a message start was encountered, a backlog of bits to be processed was added to a running total. This total was incremented when encountering new messages and decremented by feeding bits into the processing block. The maximum size of the backlog is the length of the memory,  $N$ . The simulation was run repeatedly to determine the probability that the backlog would overflow versus the memory size. This is equivalent to the probability that an ADS-B message will be dropped because it gets overwritten before it can be transferred to the processing block. The results of the simulation are shown in Figure H-8.

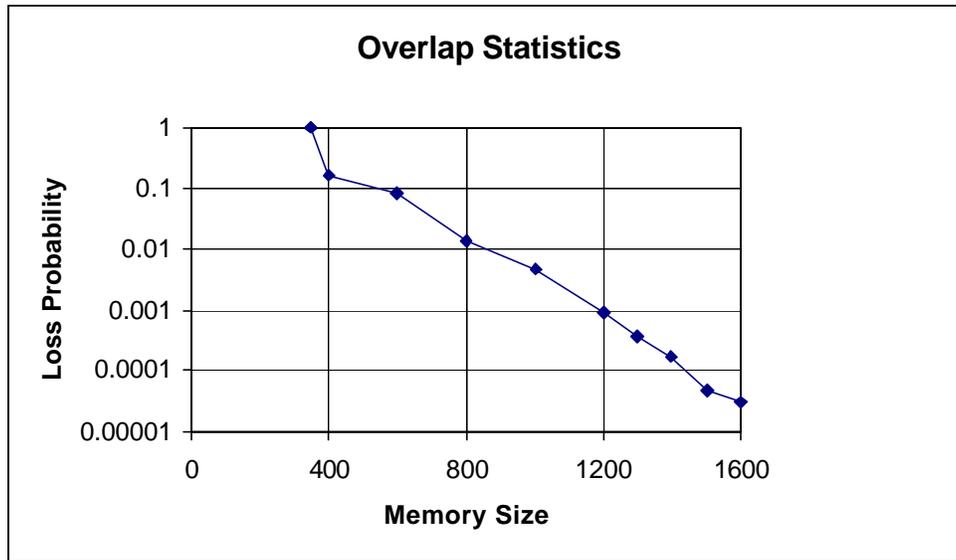


Figure H-8. Memory Length Determination (Memory Size in bits)

Figure H-8 indicates that if the memory length is 1200 bits, the probability that an ADS-B message will be dropped is 0.001. If that loss rate was acceptable, the total memory size required to implement the sorting function would be about  $6 \times 1200/8 = 900$  bytes.

### H-6. Summary

This paper has described a possible mechanism for allowing a UAT receiver to synchronize to and demodulate multiple overlapping ADS-B signals. This capability is essential for successful operation in environments with many aircraft. The suggested technique does not require extensive resources in terms of memory space or processing speed.

The paper also briefly examines the choice of the threshold value for the synchronization process. Using certain simplifying assumptions, it is shown that the original setting of 94 appears to be acceptable, but it could be lowered somewhat. A prudent approach to choosing the correct threshold value would be to measure the synchronization rate of an actual UAT receiver when no valid signal is present. The threshold could then be set by choosing a value that gives rise to an acceptably small number of false alarms per second.