2011

Simultaneous Digital Demodulation and RDS Extraction of FM Radio Signals

Jonathan Ford

John Nevins Andrews Scholars
Andrews University Honors Program

Honors Thesis

Simultaneous Digital Demodulation and RDS Extraction of FM Radio Signals

Jonathan Ford & Garret Catron

April 1, 2011

Advisor: Dr. Don DeGroot & Dr. Gunnar Lovhoiden

Primary Advisor Signature:_____

Department: _____

Abstract

FM radio plays a large part in many peoples' lives. A digital stream of information known as the Radio Data System (RDS) can be transmitted alongside the audio signal in an FM radio broadcast. This digital signal may contain information such as the current song, traffic alerts, and emergency notices. Using MATLAB, a method is presented by which the RDS data may be digitally extracted from multiple FM broadcasts simultaneously. This method's parallel nature makes Digital Signal Processing (DSP) technology, such as a Field Programmable Gate Array (FPGA), an ideal platform for implementation.

Introduction

FM Radio is a technology that has reached maturity and continues to serve a large role in our society.[1] Within the FM broadcast standard lies the ability to send digital information alongside the audio signal. This information is known as the Radio Data System (RDS) and contains information such as the current song being played, alternative frequencies, traffic, and weather. Newer FM radio receivers often have the ability to decode the RDS information sent for the single channel they are currently tuned to. While useful to see the RDS information from one station, it would be highly advantageous to simultaneously see the RDS information being broadcast from all nearby stations. In order to do this we developed a digital method that determines which stations are tunable and then extracts the RDS information from each in real time. The key components of our digital method include antenna design and signal down-converting, digitization, filtering, channel identification, FM demodulation, and finally the decoding of the RDS information itself. To illustrate how our new digital method differs from the traditional analog method of FM reception we will briefly discuss the analog method before outlining our new digital method.

Traditional Analog Method

FM radio is typically tuned using a superheterodyne receiver. This type of receiver contains two main sections. First the signal from the antenna is amplified, filtered and translated down to a fixed intermediate frequency. The second part of this receiver provides further filtering and demodulates this fixed intermediate frequency signal.[2][3]

In order to change stations the front end of the superhetrodyne receiver is retuned to a new frequency. This translates a different station onto the same fixed intermediate frequency which is then processed exactly the same way as any other station would be.

In order to gather RDS information from all tunable stations an analog system would need to do one of the following: retune very rapidly, sequentially update stations waiting for at least 5 seconds on each station, or contain numerous superheterodyne receivers. The RDS data changes every 0.84 milliseconds (ms). This means that the receiver system must be tuned to every station we are interested in within less than a millisecond. Some commercially available analog FM receivers have a seek/tune time as high as 60ms and thus cannot accomplish this timing requirement.[4] Waiting for the 5 seconds to collect all the RDS data before it repeats could work well for 2-3 stations where the station data is updated only every 10-15 seconds. However, it is common to have over 10 tunable stations in a given location. This introduces refresh times approaching one minute which no longer has the desired effect of being real time. Multiple parallel FM receivers could be used. This would require much more hardware as well as create a potential restriction on the number of stations that could be processed at one time.[5]

Our New Method

With the recent developments in Digital Signal Processing (DSP), many analog processes are being replaced by digital processing due to several key advantages. Some of these advantages include: a reduced need for dedicated hardware, re-configurability, and design flexibility. One common tool used for digital signal processing is the Field Programmable Gate Array (FPGA). This is a device that contains programmable logic blocks that can be configured to perform a wide variety of logical and mathematical operations.

While implementation falls outside the scope of this project, the method has been designed with FPGA implementation in mind.

A system block diagram of our method is shown in Figure 1. The following sections will outline and discuss each block.

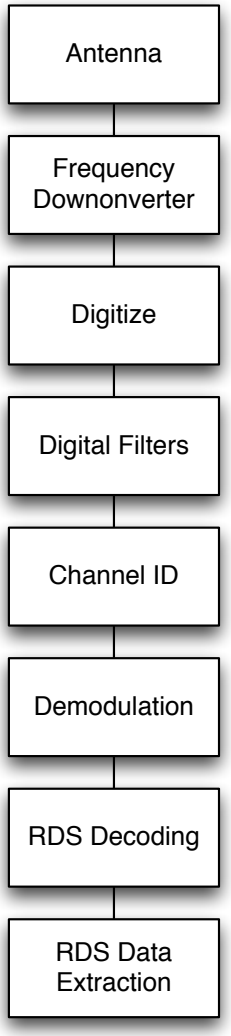

Figure 1 - System Block Diagram

Antenna

Both this method and the traditional single-channel method require an FM band antenna. Two common antennas used for FM radio receivers are the half wave and quarter wave antennas. These are typically metallic conductors with a linear length equal to ½ or ¼ the wavelength of the desired frequency. In order to receive the FM radio band (87.8MHz to 108.0MHz) an antenna tuned to a middle frequency of 97.9MHz is used. To find the wavelength of this frequency we use Equation 1,

Equation 1

$$c = \lambda \cdot f$$

Where c is the speed of light at $3\text{x}10^8$m/s, λ is the wavelength in meters (m), and f is the frequency in Hertz (Hz).

Solving for λ when $f = 97.9\text{x}10^6$Hz,

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8 \frac{m}{s}}{97.9 \cdot 10^6 \frac{1}{s}}$$

we arrive at a wavelength (λ)

$$\lambda = 3.064m$$

We can now specify the required antenna length for receiving our FM band signal. For a half-wave antenna we need an antenna length of 1.53m and for a quarter wave antenna we need a length of 0.766m.

$$\frac{\lambda}{2} = \frac{3.064}{2} m = 1.53m$$

$$\frac{\lambda}{4} = \frac{3.064}{2} m = 0.766m$$

Downconvert and Digitize

The signal that is received from the antenna is an analog signal. To do the digital processing we must have discrete numbers to compare and operate on. In order to represent the analog signal using discrete samples an analog to digital converter (ADC) is used. This device measures the input amplitude from the receiver at discrete moments in time and outputs a digital bit sequence representing the measured amplitude. The resolution of each measurement is determined by the number bits in each sample. The greater the resolution, the more accurately the original signal can be represented.

For the original signal to be represented accurately, the Nyquist sampling criterion states that the

samples must be taken at greater than twice the rate of the fastest part of the measured signal. This means that to sample the FM radio band digitally would require sampling at a rate of 2 x 108.0MHz or 216MHz at a minimum. Sampling rates four to five times the maximum signal frequency are commonly used to better represent the signal.

Digital sampling rates above the minimum required sampling rate of 216MHz are quite realizable, however, processing data at those rates often causes complications. For the purposes of this project, sampling at four times the fastest signal rate, or around 400MHz would be ideal. For the desired resolution we would like to have 14 bits per sample, which is fairly typical for ADCs. These design specifications would result in a data rate of 5.6 Gbits/second. This is equivalent to processing the capacity of one 700MB CD every second.

While hardware exists to process data at these rates it can be quite costly and complex. To aid in overcoming this obstacle we note that the FM radio band does not have useful frequency content from 0Hz up through 87.8MHz. This allows us to shift the 20.2MHz wide band of useful frequencies down to a slower frequency using a mixer. The superheterodine receiver uses a similar process to tune a particular station to the lower intermediate frequency, around which the rest of the receiver is setup to function. By shifting the center of the FM spectrum (97.9MHz) to a frequency such as 14.9MHz we will have a maximum signal frequency of 25.0MHz rather than the 108.0MHz. This translation in frequency allows us to sample at a more reasonable rate of 100MHz, a quarter of the speed required before the frequency downconversion.

If one were to sample slower than the Nyquist required rate, ambiguities in the digital information called aliasing is liable to occur.

Given that the signal has been downconverted and filtered to remove unwanted frequency content outside of the 20.2MHz wide radio band, the next step in the process is to digitize the data. An ADC that is properly configured for a particular signal would typically output digital values ranging from 0 at the minimum input value to 2^{n-1} at the maximum input value, where n is the number of bits per sample.

Filters

Throughout the process of extracting the RDS information we will need to look at only certain frequency ranges of the acquired signal. In order to achieve this the signal must be filtered. In an analog system this is typically done with analog filters made from resistor and capacitor networks. Digital systems can also filter signals. There are two general forms of digital filters, infinite impulse response (IIR) and finite impulse response (FIR) filters.

IIR filters utilize both feedforward (from previous inputs) as well as feedback (from previous outputs) to calculate the current output. An IIR filter has the mathematical form shown in Equation 2,

Equation 2

$$y[n] = \sum c[k] \cdot x[n-k] + \sum d[j] \cdot y[n-j]$$

where k and j specify the order of the filter, y[n] is the current output, x[n] is the current input and c and d are the filter coefficients which are constants that determine the frequency response of the filter. The IIR filter uses feedback, the y[n-j] summation term in Equation 2, to aid in producing the desired frequency response. The calculations needed to determine the coefficient values for a filter with feedback can be quite intensive.

An alternative to the IIR filter is the FIR filter, which is designed with only feedforward terms. The general form for a k-order FIR filter is shown in Equation 3.

Equation 3

$$y[n] = \sum c[k] \cdot x[n-k]$$

The coefficients, c[k], are found by first determining the filter specifications. Software such as the MATLAB Filter Toolbox can generate these coefficients from a list of filter parameters. These include the type of filter, the order of the

filter, the sampling frequency, cut off frequencies, and the design method for coefficient generation.

The most common types of filters are the bandpass, lowpass, and highpass. A bandpass filter allows frequencies between the low cutoff frequency and high cutoff frequency to pass through the filter while attenuating all frequencies outside this range. A highpass filter is a bandpass filter with the high cutoff frequency set at infinity. Similarly a lowpass filter is a bandpass with the low cutoff frequency set to zero Hertz.

The order of a filter defines the number of coefficients that are used. An ideal filter completely attenuates all frequencies outside of the passband and does not attenuate those inside. Most filters approach this idealized "brick-wall" shape as the number of coefficients is increased.[6]

Channel Identification

As this method is looking for the information in all tunable FM stations, we must determine which stations are tunable. This is accomplished by taking a Fast Fourier Transform (FFT) of the signal. This efficient algorithm of the Discrete Fourier Transform (DFT) decomposes a time domain sequence into its' frequency domain representation. The frequency spectrum is then sent through a threshold detector that records the frequency if the amplitude is above an experimentally determined value. This will be dependent on the noise level from the antenna, RF amplifier and filters. This threshold level will also determine the number of stations that will be detected. A frequency versus signal power plot is shown in Figure 2.[7][8][9]

Each triggered frequency is coerced to the nearest know FM station frequency. This frequency list is stored in such a way that other aspects of the method can access it for use with displaying as well as setting the individual channel configurations.

Channel Processing

Having a list of *n* FM frequencies that may be tunable, we must now split the work into *n* parallel tasks, one for each station. The full 20.2MHz wide time domain signal will be wired to the top of each of these parallel processes, at which point a bandpass filter will attenuate all but the particular station's frequency content. These bandpass filters should have a width of 250kHz centered at one of the detected frequencies. The pre-calculated coefficients for tuning all 200 FM stations would require a relatively small amount of memory and could be stored in a lookup table. Each detected frequency from the FFT would link the appropriate coefficients to each parallel task.

Frequency Demodulation

The first processing task after each channel is isolated is to demodulate the frequency modulated signal (Equation 4).[10]

Equation 4 – Frequency Modulated Signal

$$x_c(t) = A_c \cos\left[w_c t + 2\pi f_\Delta \int^t x_m(\lambda) d\lambda\right]$$

Several methods for frequency demodulation include zero crossing demodulation, slope detection and quadrature demodulation.[11]

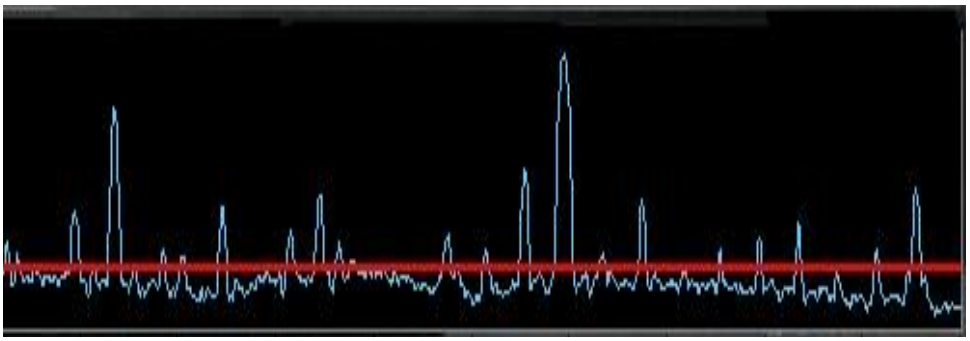

Figure 2 - Frequency Spectrum - Channel Identification (graphic from UC Berkeley Labs LABVIEW software radio)

The quadrature method utilizes two signals, inphase (I) and quadrature (Q), that typically have been modulated by the same message but have had the Q signal delayed by 90 degrees. The methods that use IQ signaling tend to have the best fidelity, however, due to the additional complexity of regenerating the IQ signals and the need for just a digital bit stream (low fidelity) for this method these were not seen as a worthwhile option.

Zero crossing demodulation is a fairly simple technique where the time between zero crossings is measured and used to estimate the instantaneous frequency of the FM signal. A decrease in time between zero crossings indicates a higher instantaneous frequency of the FM signal. The reverse can be observed for a lower instantaneous frequency of the FM signal. The instantaneous frequency of the modulated carrier minus the constant carrier frequency is directly proportional to the amplitude of the message signal. The frequency of the message signal is determined by the rate that the instantaneous frequency changes. Once we have estimated the instantaneous frequency we have demodulated the message signal. A few advantages of the zero crossing demodulation technique are its' simplicity in computation and its ability to be readily implementable in DSP hardware such as a FPGA. Another advantage is that since it looks at zero crossings it is immune to clipping of the carrier. MATLAB code demonstrating a zero crossing demodulation algorithm is included in Appendix 1.

The slope detection method of frequency demodulation is conceptually the simplest form of FM demodulation. It is also the least computationally complex. By differentiating the FM signal, Equation 4, we can observe the instantaneous frequency as an envelope of the resulting signal. This is shown in Equation 5.

Equation 5 - Differentiated FM Signal

$$\frac{dx_c}{dt} = -\left(w_c + 2\pi f_\Delta x_m(t)\right)A_c \sin\left[w_c t + 2\pi f_\Delta \int^t x(\lambda)d\lambda\right]$$

By performing envelope detection, which can be accomplished using a lowpass filter, we can separate the instantaneous frequency from the carrier. Since the instantaneous frequency is proportional to the message we only need amplify the signal as necessary and we have demodulated the FM stream and recovered the message signal. Overall, slope detection is readily realized in DSP technology and is computationally cheap. However it is highly susceptible to noise and works best if the signal can be preconditioned to remove as much noise as possible prior to demodulating it.[12][13]

Once the message has been extracted from the frequency modulation we are left with a signal that contains the monaural audio, a 19kHz pilot tone, the stereo audio difference information, and the RDS data. The RDS data must be filtered to isolate if from the audio signals. The frequency spectrum of the FM radio information is shown in Figure 3. In the figure, an idealized bandpass filter is centered on 57kHz; this represents the RDS data filter.

If a particular channel is to be listened to, a separate parallel process may be considered to perform the FM demodulation since the quality of demodulation needed for the RDS data may not be satisfactory for high fidelity audio reproduction.

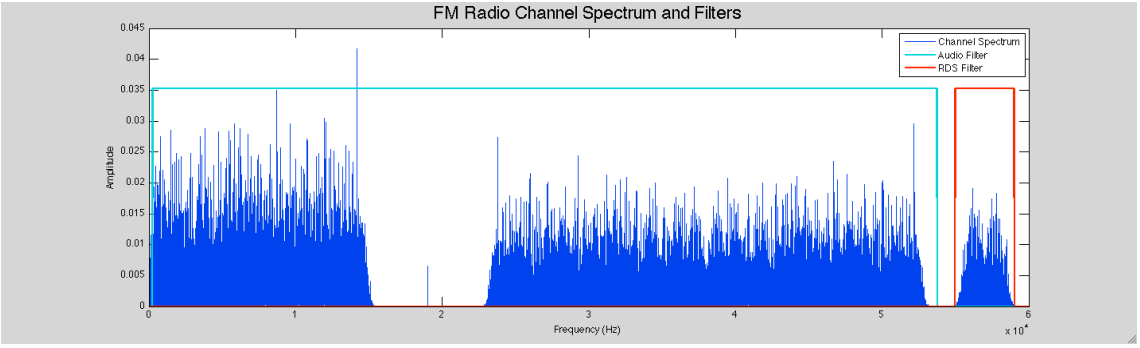

Figure 3 - Frequency Spectrum of FM Radio

RDS Extraction

The filter to isolate the RDS signal should be centered at 57kHz with a passband width of 4812Hz. This width is derived from a 2Hz uncertainty in the 19kHz pilot tone, which is tripled to achieve the 57kHz tone on which the RDS signal is modulated. The tripling of the frequency also triples the uncertainty or deviation in the center frequency. This 6Hz deviation is added to the 2.4kHz bandwidth on either side of the carrier, which is modulated with the RDS data.

The data we are interested in consists of only two states rather than a continuously changing signal. This can assist in simplifying the demodulation process. If we look at the mathematical description of the carrier, Equation 6, modulated with the message, Equation 7, we can see that when the digital information, a_i changes from 1 to -1, $x_c(t)$ changes from cos(t) to –cos(t), which is equivalent to cos(t+π) or a phase inversion. This is very similar to binary phase shift keying which changes the phase of the carrier based on the binary message signal.

Equation 6 - Amplitude modulated signal with message signal m(t)

$$x_c(t) = m(t)\cos(2\pi f_c t)$$

Equation 7 - Digital message signal

$$m(t) = \sum_{i=-\infty}^{\infty} a_i \quad \| \{a_i\} = \pm 1$$

For a signal to change instantly would require an infinite bandwidth which is not practical or realizable. To smooth out the digital signal a pulse shaping function h(t) is applied to the binary message signal as shown in Equation 8.

Equation 8 - Digital message signal with shaping function h(t)

$$m(t) = \sum_{i=-\infty}^{\infty} a_i h(t - iT_s) \quad \| \{a_i\} = \pm 1$$

The RDS standard specifies that the frequency response, or transfer function, of the pulse shaping function H(f) should be that of Equation 9, where td = 1/1187.5 seconds.

Equation 9 - Pulse shaping filter to limit message spectrum

$$H_T(f) = \cos\left(\frac{\pi f t_d}{4}\right) \quad | 0 < f < \frac{2}{t_d}$$

This is a matched filter that the transmitter and the receiver both apply to the pulse. The combination is called a raised cosine filter and is ideal for digital transmission and helps in reducing inter symbol interference (ISI).

After applying the pulse shaping function to the RDS data, the FM radio station modulates the third octave of the FM pilot tone (19kHz) with the shaped message signal using amplitude modulation dual sideband with a suppressed carrier (AMDSB-SC). This means that the 57kHz tone is canceled out before transmission. This allows for transmission energy to be used for the message and not for the supporting signals such as the carrier tone. This also means that on the receiving end, the 57kHz carrier must be recreated for demodulation.

To create a sinusoid at a particular frequency in digital processing a trigonometric lookup table is used with a clock scaled to an appropriate rate. The values are sequentially read from the table to generate a sampled sinusoid at whatever clock frequency is needed. Jumping ahead or lagging behind can also adjust phase slightly if needed to match the phase of the incoming signal. This process must be done to regenerate the 57kHz carrier tone. Once generated, this is multiplied with the received AMDSB-SC signal to restore the suppressed carrier. The result of this product generates sinusoids at the sum and differences of the 57kHz tone and the frequency of the received signal. This also modulates the tone such that we can now extract the message by examining the envelope. Both the sum and difference terms contain the modulated data and either can be used to extract the message signal. This is done by envelope detection using a lowpass filter.

To reduce the DC power on the 57kHz carrier by not having the signal stay at any value for too much time, each RDS data bit is represented by a bi-phase symbol as expressed in Equation 10 for a logic 1, and Equation 11 for a logic 0.

Equation 10 - Logic 1 biphase symbol

$$e(t) = \delta(t) - \delta(t - \tfrac{t_d}{2})$$

Equation 11 - Logic 0 biphase symbol

$$e(t) = +\delta(t) + \delta(t - \tfrac{t_d}{2})$$

This means that for each bit transmitted, the signal will change polarity. Because of the raised cosine pulse-forming filter, the data does not appear as a repeating square wave, but more like a series of rolling hills. The digital information is extracted from this by locking onto the zero crossings and then examining the data between. Every t_d = 1/1187.5 seconds, a new binary value is sent. By comparing the signal over a time period t_d we are able to determine if it matches Equation 10, or Equation 11. This matching can be done with a correlation block, or can look at the derivative of the message to determine the direction of change. The output of the matching will generate a continuous stream of binary values, zeros and ones known as a bitstream.

The binary RDS data is differentially encoded at the radio station according to the rules in Table 1. This is the truth table for a logical XOR.

Table 1 - RDS Differential Encoding Rules

Previous Output	New Input	New Output
0	0	0
0	1	1
1	0	1
1	1	0

On the receiving side of the channel the bitstream is again sent through a differential decoder according to the rules in Table 2.

Table 2 - RDS Differential Decoding Rules

Previous Input	New Input	New Output
0	0	0
0	1	1
1	0	1
1	1	0

This encoding/decoding process is done so that the data is correct regardless of polarity.

At this point the RDS bitstream has been extracted and the data rate is quite slow such that the composite of all the stations being processed could be quite easily transferred for further processing elsewhere. At the same time, integrating a RDS interpreter into the same parallel pipeline would not require significant extra processing or resources.

RDS Interpreter

The RDS standard specifies the bit sequence and format of the bitstream. In general, the stream contains consecutive groups of 104 bits. Each group is made up of four 26 bit blocks. Each block is then made up of 16 bits of information and 10 checkword bits for error correction.

Groups are continuously transmitted one after another. One group at 104 bits with a bit rate of 1187.5 bits/s takes 87.6ms to transmit. The RDS standard specifies the information format for the groups and blocks and will not be repeated here.[14]

There are many ways that the RDS data can be processed. One method, would be to use pattern matching circuits and the parallel nature of DSP hardware to gather the correct bits at the right times. This would work by passing the bitstream through a shift register that is compared to a certain bit sequence every clock cycle. The correlation of the two sets of data would be taken and if they match a flag is set triggering a secondary process that extracts the needed data. The RDS data from all the parallel processes can be stored in memory such that it can be updated in real time and accessed by a display or other algorithms for further processing if desired.

Conclusion

The method for the simultaneous demodulation and extraction of RDS data from multiple FM stations is both novel and realizable. While conducting our literature search we have found several papers on using varying degrees of digital processing to access the RDS data inside of a single FM channel, but found none that were

looking at multiple RDS streams at once. While all the individual components used in this method are sound and industry proven, to this point, we have not seen a system such as ours discussed or proposed.

While we have not implemented the proposed system for simultaneous RDS extraction in hardware such as an FPGA, we have done research into the limitations and capabilities of current hardware and are confident that hardware exists that can easily do all that this method would require. We have verified that the MATLAB Filter Toolkit is not only capable of generating coefficients but will also export the filter to Verilog, which is a Hardware Descriptive Language (HDL) used to configure DSP hardware such as FPGAs. This capability of MATLAB should greatly assist anyone planning on implementing filters in DSP applications.

The RDS standard is quite flexible and is capable of much more than it is currently used for. Implementation of a method such as ours in standard radio units would allow whole new opportunities for FM radio usage. Allowing FM radios to do more than just play music, the user would now have access to information such as weather alerts, traffic warnings, news bulletins and programming information from all stations simultaneously in real time.

APPENDIX 1

```
%-----------------------------------------------------------------

% Zero Crossing FM detection Algorithm

% Garret Catron 3-31-11

% Generates a test FM signal and demodulates it using the zero crossing FM
% demodulation technique. Also plots extracted signal verses original
% message signal for comparison.

%-----------------------------------------------------------------

StepSize = .0001;           % Set step size in seconds for test data
Duration = 4;               % Set duration in seconds for test data
t= 0:StepSize:Duration;     % Initialize time vector
CarrierFrequency = 50;      % Set carrier frequency in Hz
MessageFrequency = 1;       % Set message frequency in Hz

MessagePeriod = 1/MessageFrequency;
SampleWidthMessagePeriod = floor(MessagePeriod/StepSize);

MessageSignal = sin(MessageFrequency*2*pi*t);       % Create test message signal
y = cos(2*pi*CarrierFrequency*t+2*MessageSignal);   % Modulate carrier with test message


PreviousSample = 0;
FrequencyDeviation = 0;
PreviousSignChange = 1;                 % Initializing values to minimize startup noise

ExtractedSignal = zeros(1,1+Duration/StepSize); % Initialize vector to hold extracted
signal

% Begin looping through samples
for k=1:Duration/StepSize;

    % Get current sample from array
    CurrentSample = y(k);

    % Determine sign of current sample true for positive
    CurrentSampleSign = CurrentSample > 0;

    % Determine sign of previous sample
    PreviosSampleSign = PreviousSample > 0;

    % Determine if a sign change has occurred
    SignChange = xor(PreviosSampleSign,CurrentSampleSign);

    if (SignChange)

        % Use time vector to determine time between zero crossings
        TimeBetween = (t(k) - t(PreviousSignChange));

        % Calculate Frequency Deviation
        FrequencyDeviation = 1/(2*TimeBetween)-CarrierFrequency;

        % Set value for next iteration
        PreviousSignChange = k;
    end

    ExtractedSignal(k) = FrequencyDeviation;        % Build extracted signal array
    PreviousSample = y(k);      % Set Value for next iteration

end

% Plot signal and extracted signal on same graph
plot(t,ExtractedSignal,t,MessageSignal);
```

REFERENCES

[1] D. Kopitz, J. Beerling and B. Marks "FM Radio With RDS Will Have A Long Future", *RADIOWORLD*, Apr. 2010; http://www.rwonline.com/article/98568 (Accessed 2010-10-11)

[2] A. B. Carlson, P. B. Crilly and J. C. Rutledge, Communications Systems: An Introduction to Signals and Noise in Electrical Communication. 4th edition. Boston, MA: McGraw Hill 2002

[3] A. B. Cook and A.A. Liff, Frequency Modulation Receivers. Englewood Cliffs, N.J.: Prentice-Hall 1968

[4] http://www.silabs.com/Support%20Documents/TechnicalDocs/Si4702-03-C19.pdf

[5] A. A. Smith Jr., Radio Frequency Principles And Applications: The Generation, Propagation, And Reception Of Signals And Noise**.** New York, NY: IEEE Press, 1998

[6] "Bores Signal Processing – Training in DSP and Media Processing Home Page", Oct. 2010; http://www.bores.com/

[7] R. G. Lyons, Understanding Digital Signal Processing 2nd Edition. Upper Saddle River, NJ: Prentice Hall, 2004

[8] K.G. Beauchamp, Signal Processing Using Analog And Digital Techniques**.** New York, NY: Wiley 1973

[9] "UC Berkeley Develops Communications Lab with NI Tools", Oct. 2010; http://zone.ni.com/devzone/cda/pub/p/id/40

[10] A. D. Poularikas, Signals And Systems Primer With MATLAB. Boca Raton, FL: CRC Press/Taylor & Francis, 2007

[11] G. M. Russell, Modulation And Coding In Information Systems. Englewood Cliffs, NJ: Prentice-Hall, 1962

[12] V. K. Madisetti and D. B. Williams, The Digital Signal Processing Handbook. Boca Raton, FL: CRC Press New York, NY: IEEE Press, 1998

[13] W. R. Bennet and J. R. Davey, Data Transmission. New York, NY: McGraw-Hill 1965

[14] "EBU UECP Specification", Oct. 2010; http://www.rds.org.uk/rds98/ebuuecpspecification.htm