Scanning Digital Radar Receiver
Final Report

by

Ryan Hamor

Project Advisor:
Dr. Brian Huggins

Bradley University
Department of Electrical and Computer Engineering
May 8, 2006
Table of Contents

Abstract ........................................................................................................................ 3

Function Description and System Block Diagram .............................................. 4-9

  Functional Description .................................................................................. 4-5

  Mixer Subsystem ......................................................................................... 5

  Data Acquisition Subsystem ....................................................................... 5-6

  Frequency Synthesizer Subsystem .............................................................. 6

    DDS Theory of Operation ......................................................................... 6-7

  Signal Processing & Control Subsystem ....................................................... 7-9

    DSP Flow Chart ......................................................................................... 8

    Signal Detection ....................................................................................... 9

Mathematical Verification .................................................................................. 10-11

Design of IF Filter and Amplifier ..................................................................... 12-13

Simulation Results ............................................................................................ 13-14

Experimental Results ....................................................................................... 15-16

Analysis ............................................................................................................. 16-17

Conclusion ......................................................................................................... 17

Datasheet ........................................................................................................... 18

Related Standards and Patents ......................................................................... 19

References ......................................................................................................... 20

Appendix A ......................................................................................................... 21-26

Appendix B ......................................................................................................... 27
Abstract

An important aspect of Electronic Warfare is the development of Radar Warning Receivers (RWR). With the EM spectrum being extremely volatile in a combat situation, the RWR allows for classification and identification of enemy radar signals. Once these signals have been identified they can be used to indicate possible threats, and as a precursor to jamming devices. The Scanning Digital Radar Receiver is a scaled down version of a RWR. It identifies Pulse Radar signals and determines there characteristics. This paper describes the receiver front end design, as well as the software algorithms used to determine signal characteristics.
I. Introduction

The Scanning Digital Radar Receiver is a project supported by Northrop Grumman Corporation. The function of the system is to detect a pulsed radar signal over a large frequency and power range. The frequency range is from 50 to 120 MHz and the power level can be as low as -20 dBm. After a pulsed radar signal has been identified, the system will output the characteristics of the signal to the user.

II. High Level Block Diagram

The high level system block diagram is shown in figure 1. The user input is strictly a software input, through this input the LO scan range can be set. The pulse radar signal is mixed with the very stable frequency synthesizer, which acts like a local oscillator.

The internal signal, denoted as IF, is the down converted pulse radar signal. It is fed into the A/D board which is connected to the PC. The signal is then processed using DSP methods, and if a radar signal is present the signals characteristics are sent as outputs.

(Figure 1 High Level System Block Diagram)
The illustration of a pulse radar signal is shown in Figure 2. By processing the signal in the frequency domain, the system can determine time domain characteristics. The following is the list of outputs and how they relate to Figure 2:

- The Pulse Width (PW) corresponds to $\tau$.
- The Pulse Repetition Rate (PRR) is the inverse of the period, $T$, of the time domain signal.
- The Freq RF is the frequency of the sinusoid under the pulse envelope denoted as $F_{RF}$.

![Figure 2 Time Domain Representation of the Pulsed Radar Signal]

**III. High Level Block Diagram Subsystems**

*Mixer*

Figure 3 is the mixer subsystem. The mixer multiplies the RF signal with LO signal and then outputs the IF signal. The intermediate signal contains frequencies $F_{RF} - F_{LO}$ and $F_{RF} + F_{LO}$, where $F_{RF}$ was previously defined and $F_{LO}$ is the frequency of the LO signal. The purpose of this subsystem is to frequency translate the RF signal to a lower frequency so that it can be sampled by the data acquisition subsystem. During the conversion process there is an average power loss of 5.24 dB. The frequency of the LO signal is limited to 120 MHz to prevent aliasing. This limit is also set on the frequency of the RF signal.

![Figure 3 Mixer Subsystem]
**Data Acquisition**

Figure 4 shows the Data Acquisition subsystem. It is composed of an analog filter, amplifier, and the data acquisition board. The IF signal is filtered using a low-pass filter with a 200 KHz cutoff, which passes the F\text{IF} frequencies with little attenuation, but significantly attenuates signals with frequencies above 200 KHz. After filtering, the signal is amplified using a 741 op-amp in a non-inverting configuration. The amplifier adds about 20 dB of gain to compensate for the losses due to the filter. The amplifier is also used to set the voltage scale of the IF signal for the best data acquisition.

The data acquisition card is the I/O tech 3005USB; it has a maximum sampling rate of 1MS/sec and 16 bit resolution. The 1 MHz sampling rate applies for only one channel. While the card has 16 channel, every additional channel in use cuts the sampling rate by two. In addition to the analog inputs, the 3005 USB contains 32-bit counters and times for automated sampling.

![Diagram](Figure 4 Data Acquisition Subsystem)

**Frequency Synthesizer**

Figure 5 shows the Frequency Synthesizer subsystem. The control word is sent to the frequency synthesizer, through the parallel port, which determines the frequency of the synthesizer output. To boost power to the mixer a 10 dB amplifier is used. The amplifier is needed to get the amplitude from the DDS frequency synthesizer up to the 7 dBm level needed for the mixer.

The Analog Digital DDS 9854 is used as the frequency synthesizer. It uses direct digital synthesis to create a very stable sinusoid wave form. The maximum frequency of the synthesizer is 120 MHz to prevent aliasing.

![Diagram](Figure 5 Frequency Synthesizer Subsystem)
**DDS Theory of Operation**

DDS or Direct Digital Synthesis is the ability to use a digital system to create an analog output. Figure 6 is a block diagram of the core of a generic DDS system. The frequency tuning word is loaded into the Phase Accumulator. The values in the Phase Accumulator correspond to output points on the cycle of the output sinewave. Resolution of the Phase Accumulator is determined by N and can range from 24 to 48 bits. The Phase Accumulator functions as a counter and every pulse from the reference clock increase the count value. By increasing the frequency tuning word the counter takes bigger steps and the sinewave cycle is completed faster, thus increasing the output frequency.

(Figure 6 Block Diagram of DDS Core)

The output of the phase accumulator is linear and cannot directly be used to generate a sinewave or any other waveform except a ramp. Therefore, a phase-to-amplitude lookup table is used to convert a truncated version of the phase accumulator’s instantaneous output value into the sinewave amplitude information that is presented to the D/A converter.

**Signal Processing & Control Subsystem**

Figure 7 shows the components contained in the signal processing and control block. Visual Basic is used to communicate with the frequency synthesizer, configure the system, and display results. A server version of Matlab is used to process the radar signal. Through a COM interface Visual Basic can run m-files, as well as extract results from the Matlab server.

(Figure 7 Signal Processing and Control Block Diagram)
**DSP Flow Chart**

Figure 8 shows how the radar signal is processed in Matlab. The signal processing begins by checking if a signal was detected. If no signal is detected the frequency synthesizer is incremented and the program exits. However, if a signal is detected, then its FFT is taken.

The FFT consists of an array whose points correspond to various magnitudes. To get frequency information, a frequency array is constructed based on the number of points in the FFT and the sampling rate. To find the IF frequency the point with the maximum magnitude is found in the FFT array and is correlated with the same point in the frequency array. To get the PRR the point with the next largest magnitude is found. The frequency of that point is subtracted from the IF frequency to get the PRR. Finally, to get the pulse width the local minimum is found and its frequency is subtracted from the IF frequency.

(Figure 8 DSP Flow Chart)

All code is written in Matlab in the form of m-files and are included in Appendix A.
Signal Detection

The system has two levels of detection before the signal processing begins. Figure 9 illustrates this detection process. The signal must first pass Level 1 Detection, which determines if the signal’s amplitude is above a preset threshold level. For this system the threshold level was set to 200 mV. This threshold corresponds to about -20 dBm signal level on the input. Once Level 1 Detection has been passed, the signal must pass Level 2 Detection. During Level 2 Detection the system takes an FFT with a small number of points and then searches for the spikes on either side of the main spike. If these spikes are detected then a pulse radar signal has been detected and the signal can continue to be processed.

(Figure 9 Detection Process)

Both Level 1 and Level 2 detection work in unison to decide if a radar signal has been detected. The goal is to provide multiple detection process to speed the system up by not wasting time performing full signal processing to determine if the signal is a pulse radar signal.
IV. Mathematical Verification

The purpose of this section is to derive the relationships between the radar characteristics in the time domain and frequency domain. Figure 11 is the time domain representation of a pulsed radar signal. To obtain the FFT the signal is broken up into two components, which are then multiplied together.

(Figure 11 Time Domain Representation of Pulsed Radar Signal)

The two time domain components are:

\[
\begin{align*}
    x_1(t) &= \cos(2\pi f_{RF} T) \quad (1) \\
    x_2(t) &= A \sum \prod \left( -\frac{nT}{\tau} \right) \quad (2)
\end{align*}
\]

Then;

\[
g(t) = x_1(t)x_2(t) \quad (3)
\]

Use Multiplication Theorem (table 2-1, Couch)

\[
x_1(t)x_2(t) = X_1(f) \ast X_2(f) \quad (4)
\]

Using Fourier Transform (table 2-2, Couch) and Time Delay Theorem (table 2-1, Couch)

\[
\begin{align*}
    X_1(f) &= \frac{1}{2} \delta(f - f_{RF}) + \frac{1}{2} \delta(f + f_{RF}) \quad (5) \\
    X_2(f) &= A \sum \pi \text{sinc}(\pi f) e^{j - 2\pi f n T} \quad (6)
\end{align*}
\]

\[
G(f) = X_1(f) \ast X_2(f) \quad (7)
\]
\[ G(f) = \frac{1}{2} \delta(f - f_{RF}) + \frac{1}{2} \delta(f + f_{RF}) * A \sum_{n} \tau \text{sinc}(\tau f) e^{j2\pi fnT} \quad (8) \]

\[ G(f) = \frac{A\tau}{2} \sum_{n} [\delta(f - f_{RF}) \text{sinc}(\tau(f - f_{RF})) e^{j2\pi fnT} + \delta(f + f_{RF}) \text{sinc}(\tau(f + f_{RF})) e^{j2\pi fnT}] \quad (9) \]

Using Frequency Shifting Property. (Haykin, pg.38-39)

\[ G(f) = \frac{A\tau}{2} \sum_{n} [\text{sinc}(\tau(f - f_{RF})) e^{j2\pi fnT} + \text{sinc}(\tau(f + f_{RF})) e^{j2\pi fnT}] \]

\[ G(f) = \frac{A\tau}{2} \text{sinc}(\tau(f - f_{RF})) \sum_{n} e^{j2\pi fnT} + \frac{A\tau}{2} \text{sinc}(\tau(f + f_{RF})) \sum_{n} e^{j2\pi fnT} \]

Using Poisson Sum Formula. (Eqn 3-38, Couch)

\[ \sum_{n=-\infty}^{\infty} e^{\pm j2\pi fnT} = \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T}) \]

\[ G(f) = \frac{A\tau}{2} \text{sinc}(\tau(f - f_{RF})) \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta((f - f_{RF}) - \frac{n}{T}) + \frac{A\tau}{2} \text{sinc}(\tau(f + f_{RF})) \frac{1}{T} \sum_{n=-\infty}^{\infty} \delta((f + f_{RF}) - \frac{n}{T}) \quad (12) \]

Figure 12 below is a plot of the above equation. It shows the doubled sided sinc functions and how they are made up of the delta functions.
V. Design of IF Filter and Amplifier

The following section discusses the design of the IF filter and amplifier. All of the other system components are off the self; therefore, there is no design involved.

**IF Filter**

For the IF filter a low-pass, fifth order, Butterworth design was chosen. Other active filter designs were rejected due to the stray capacitances associated with the bread board. This filter was implemented on vectorboard which better isolates components. Figure 13 is the schematic for the IF filter. The design comes from *Radio Frequency Electronics* by Jon Hagen. The filter design consists of two cascaded pi networks whose values were designed for a cutoff of 1 rad/sec and then unnormalized for a cutoff of 200 KHz.

![Figure 13 Low Pass IF Filter Schematic](image)

Figure 14 is the experimental verification of the IF filter. The 3 dB point is at the designed for cutoff frequency of 200 KHz. However, the steady gain is about -5dB due to the 50 Ohm resistors, this problem is corrected when the filter is inserted into the system. The problem arises because the input and output loads are not matched.

![Figure 14 Experimental Verification of IF Filter](image)
**IF Amplifier**

Figure 15 is the circuit schematic for the IF amplifier. It is a LM741 op-amp in a non-inverting configuration. The purpose of the amplifier is to make up for signal loss that is encountered in various parts of the system. The design gain was 21 V/V; however, at the 100 KHz operating frequency it was 12 V/V. This drop in gain was due to the 1.5MHz unity gain bandwidth associated with the LM741.

![IF Amplifier Circuit Schematic](image)

**VI. Simulated Results**

The Simulink block diagram for the system is shown in Figure 16. It includes the RF front end, local oscillator, and the IF filter. The filter is set for 200 KHz and the signal coming from the RF_signal block is the pulsed radar signal.

![Simulink Block Diagram](image)
For the simulation the RF signal is a 105 MHz sinusoid mixed with a 2 ms period, 20% duty cycle pulse train. The local oscillator signal is at 104.9 MHz, the output signal will be at $F_{IF} = F_{RF} - F_{LO}$. Figure 17 is the time domain plot of the filtered IF signal.

(Figure 17 Time Domain Plot of Simulated IF Signal)

The signal processing scheme previously described is used to extract time domain information mathematically from the frequency domain. By taking an FFT of the signal, the time domain characteristics can be obtained as shown in Figure 10. The code to perform the calculations is in Appendix A. To find the original RF signal the LO frequency is added to the IF frequency. So 104.9 MHz added to 100 KHz is a 105 MHz.

In Figure 18 the FFT of the IF signal is shown. The plot shows the frequency points from which the time domain characteristics are calculated. The corresponding variable locations are the same that were derived in section IV. It should be noted that this plot is a plot of the magnitude of the FFT. Since the FFT is has a limited number of points, about 40,000, there is not an exact zero point to calculate pulse width from. In this case an average value for that local minimum is determined.

(Figure 18 Frequency Domain Plot of the IF Signal)
VII. Experimental Results

The test setup for the system is shown in Figure 19. All of the components are given and the signals are labeled. The oscilloscope was used at the input to the A/D to make sure that there was not a problem with the signal coming in. The function generator used for the radar signal has a frequency range from 250 KHz to 4 GHz. The test results can be found in appendix B.

![Figure 19 Setup for System Testing](image1)

In Figure 20 the time domain of a sampled signal can be seen. The frequency of the RF signal is 100 MHz and the Local Oscillator frequency is 99.9 MHz. Pulse repetition rate is 500 Hz and the pulse width is 400 uS (2500 Hz).

![Figure 20 Sampled Pulse Radar Signal in Matlab](image2)
Figure 21 is the results for the signal in the frequency domain from the FFT, which is used to determine the characteristics of the radar signal. The IF frequency is center peak at 100 KHz as expected. Thus, \( \text{LO} + \text{IF} = 100 \text{ MHz} \) which is correct. Also the PRR is determined to be 500 Hz and the pulse width is at 2500 Hz. All of the values match up and correlate with simulation results.

![Experimental Results Frequency Domain](image)

(Figure 21 Experimental Results Frequency Domain)

VIII. Analysis of Results

The system specifications established in the initial phase of the project are listed below:

1. RF frequency range from 50 to 120 MHz
2. PRR from 500 to 10000 Hz (2 to 0.1 mS)
3. Pulse width up to 40%

The system had no problems in analyzing RF frequencies in the specified range. The system could go even higher; however, it was limited by the output of the frequency synthesizer. If the synthesizer would have been able to go above 120 MHz then I could have scanned all the way up to 500 MHz, which was the limiting frequency of the mixer and LO amplifier.

The system was able to detect PRR’s from 500 to 10000 Hz. The upper limit for PRR was 10000 Hz. For PRR’s greater than 10000 Hz a higher resolution FFT would need to be used. PRR’s lower than 500 Hz could be detected if the number of samples is increased to make sure that twice the period of the pulse was captured.

The system was only able to detect pulse widths up to 25%. This was because of the resolution of the FFT vector. If the number of points was increased the systems scan time would be greatly increased as well.

One of the areas where system performance was lacking was in the scan time which is the time it takes the system to scan the full 50 to 120 MHz range. It took 1 second to scan a 10 MHz
bandwidth. Therefore, to scan the full bandwidth it would take approximately 70 seconds. This is very slow and would not be suitable for a production system.

IX. Conclusions

The Project was an overall success. The specifications were not met exactly; however, the main objective was to make sure the system was able to automatically scan. The system was able to successfully scan the entire frequency range and detect and characterize a pulse radar signal with the range. Also there was no error when determining the PRR of the signal. Only when the signal reached 10000 Hz PRR was the system unable to detect it.

The system could detect pulse widths up to 25%. Pulse widths greater than 25% require an increase in the number of points in the FFT vector. Even though it could not detect such large pulse widths that does not present very much of a problem, in radar systems is not practical to use large pulse widths because it wastes power and decreases possible chances of finding targets.

One of the main reasons for the slow scanning rate was using Matlab to do my real time signal processing. Matlab is slow to take FFTs and scanning the large FFT vectors was also extremely slow. Another factor that slowed the system down is the fact that control of the frequency synthesizer was done with visual basic through the parallel port.

The systems performance could be greatly increased if a DSP board and microcontroller replaced the PC. The DSP board would allow C code that could be generated by Simulink, to efficiently process the signal. Furthermore, the microcontroller could be used to control the AD 9854 Evaluation Board.
X. Datasheet

Table 1 is the datasheet for the Scanning Digital Radar Receiver. It describes input requirements and individual subsystem requirements.

(Table 1 Scanning Digital Radar Receiver Datasheet)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Min</th>
<th>Max</th>
<th>Typical</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYSTEM SPECIFICATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel Scanning Time</td>
<td>0.1</td>
<td>70</td>
<td>-</td>
<td>[Sec]</td>
</tr>
<tr>
<td>RF Power</td>
<td>-20</td>
<td>-10</td>
<td></td>
<td>[dBm]</td>
</tr>
<tr>
<td><strong>INPUT SPECIFICATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carrier Frequency (f_{RF})</td>
<td>50</td>
<td>120</td>
<td>-</td>
<td>[MHz]</td>
</tr>
<tr>
<td>Pulse Repetition Rate (PRR)</td>
<td>500</td>
<td>10000</td>
<td>500</td>
<td>[Hz]</td>
</tr>
<tr>
<td>Pulse Width (PW)</td>
<td>50</td>
<td>500</td>
<td>-</td>
<td>[us]</td>
</tr>
<tr>
<td>Pulse Width (PW) Error</td>
<td>5</td>
<td>25</td>
<td>20</td>
<td>[% of PRR]</td>
</tr>
<tr>
<td><strong>MIXER (ZAD-1)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency Range</td>
<td>D.C.</td>
<td>500</td>
<td>-</td>
<td>[MHz]</td>
</tr>
<tr>
<td>Average Conversion Loss</td>
<td></td>
<td></td>
<td>5.28</td>
<td>[dB]</td>
</tr>
<tr>
<td><strong>LOCAL OSSCILATOR (AD9854)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Frequency</td>
<td>0</td>
<td>300</td>
<td>120</td>
<td>[MHz]</td>
</tr>
<tr>
<td>Step Size</td>
<td>0.1</td>
<td>-</td>
<td>1.00E+05</td>
<td>[Hz]</td>
</tr>
<tr>
<td>Clock Jitter (100 MHz)</td>
<td></td>
<td></td>
<td>7</td>
<td>[Ps rms]</td>
</tr>
<tr>
<td><strong>DATA ACQUISITION (IOTech 3005USB)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sampling Rate</td>
<td>1</td>
<td>1</td>
<td></td>
<td>[MHz]</td>
</tr>
<tr>
<td>Number of Bits</td>
<td>16</td>
<td>16</td>
<td></td>
<td>[bits]</td>
</tr>
</tbody>
</table>
XI. Standards Relating to Project

The following is a list of standards documents that pertain to this project. Since the standards must be purchased, only their basic information is listed. Information was gathered from www.nssn.org.

IEEE Standard Radar Definitions
Author: IEEE
Published: 4/27/1998
Document #: 686-1997

Early Warning Systems
Author: Department of Defense (DoD)
Published: 8/31/1995
Document #: QAP-124 ED.1

Electronic Counter-countermeasures in Radar Systems Acquisition
Author: Department of Defense (DoD)
Published: 6/05/1987
Document #: MIL-HDBK-293

XII. Patents Relating to Project

The following is a list of patents with abstracts that pertain to this project. Information was gathered from www.uspto.gov.

Patents

5,010,341 High Pulse Repetition Frequency Early Warning Receiver
4,860,013 Automatic Threshold Multi-channel Digital Radar Early Warning Sys.
6,879,281 Pulse Radar Detection System
6,587,072 Pulse Radar Detection System
4,504,830 Synthetic Pulse Radar System and Method
6,693,578 Mixer Optimization for Active Radar Warning Receiver

Patent Applications

20050179582 Radar Detection Method and Apparatus
20050134502 Pulse Wave Radar Device
20030193430 Pulse Radar Detection System
20040212358 Scanning RF Receiver
References


Appendix A (Software)

The following software listings are m-files used during signal processing.

% This m-file initializes the A/D board using the Data Acquisition Toolbox.
% Created by Ryan Hamor 03/6/2006

ai=analoginput('iotdaq',1,'setup_0.xml');  % create A/D object
addchannel(ai,16);  % setup channel and input info
ai.channel.inputrange=[-1;1];
ai.samplespertrigger=4000;
ai.samplerate=1e6;

% m-file to take the sample data
% created by Ryan Hamor 3/6/06

start(ai)  % sample the data
data=getdata(ai);  % extract the sample from the engine
sig=data-mean(data);  % Take out DC
maxamp=max(sig);  % find max amplitude for threshold testing

% this m-file is responsible for detecting the difference between a pulse
% waveform using a low resolution fft. The number of points is 2^12.
% Using a power of 2 allows for faster fft's.

fftamt=4096;  % set the fft length for
sigfft=abs(fft(sig,fftamt));  % take fft
nyq=fftamt/2;  % nyquist sampling point so no oversample
fc_mag=max(sigfft(1:nyq));  % only search for the first peak
ptm=1;  % initialize maximum pointer
while sigfft(ptm)~=fc_mag
    ptm=ptm+1;
end

% Find the deltaR peak.
pl=ptm;  % initialize pointers
p2=pl+1;

while sigfft(pl)>sigfft(p2)
    pl=pl+1;
    p2=p2+1;
end
while sigfft(p2)>sigfft(pl)
    pl=pl+1;
    p2=p2+1;
end
mag2=sigfft(pl);

threshold = fc_mag/2  % Set the threshold level at 50% of max mag
if mag2 > threshold
    det = 1
else
    det = 0
end
%M-file to take the fft of a signal that meets intial threshold
%requirements. It then finds it center frequency and the first delta
%spikes on either side if present. If they are not 50% of the amplitude
%of the center spike, then no signal is detected.
%created by: Ryan Hamor 3/8/06.

cond = 0; %Variable to set if a peak is ligitemate
ts=length(sig); %set the fft length for 10 * number of
fftamt=10*ts; %points.
sigfft=abs(fft(sig,fftamt)); %take fft
%setup the freq scale
freq=1e6*[0:fftamt-1]/fftamt;
nyq=fftamt/2; %nyquist sampling point so no oversample
fc_mag=max(sigfft(1:nyq)); %only search for the first peak
ptm=1; %intialize maximum pointer
while sigfft(ptm)~=fc_mag %intialize maximum pointer
    ptm=ptm+1;
end
fc=freq(ptm); %save the carrier frequency

%Find the deltaL and deltaR peaks.
pl=ptm; %Intialize pointers
p2=pl+1;
while cond == 0
    while sigfft(pl)>sigfft(p2)
        pl=pl+1;
        p2=p2+1;
    end
    while sigfft(p2)>sigfft(pl)
        pl=pl+1;
        p2=p2+1;
    end
    mag2=sigfft(pl);
    if mag2 > 0.4*fc_mag
        magdeltaL=sigfft(pl); %Save the location of the magnitude.
        fdeltaL=freq(pl); %Get the frequency of the delta on left
        cond = 1; %Set condition to get out of loop
    else
        pl=pl+1;
        p2=p2+1;
    end
end

%search for right side delta
cond = 0;
pl=ptm;
p2=ptm-1;
while cond == 0
    while sigfft(pl)>sigfft(p2)
        pl=pl-1;
        p2=p2-1;
    end
    while sigfft(p2)>sigfft(pl)
        pl=pl-1;
    end
end

22
p2=p2-1;
end

mag2=sigfft(p1);
if mag2 > 0.4*fc_mag
    magdeltaL=sigfft(p1);
    fdeltaL=freq(p1);
    cond = 1;  %Set condition to get out of loop
else
    p1=p1-1;
    p2=p2-1;
end
end

%Figure out if the signal is detected.
avg_fdel=((fdeltaR-fc)+(fc-fdeltaL))/2;  %computer average delta freq
tPRR=1/avg_fdel;
tPRR=tPRR*1e4;  %Round the PRR to nearest 10,000th
tPRR=round(tPRR);
PRR=tPRR*1e-4;

%m-file to find the Pulse Width of a detected Pulse Radar Waveform.
The technique used is to count the delta spikes and figure the Pulse
Width from there. The code can only find Pulse Width that are equal or
less than 20%.
%Created By: Ryan Hamor
%3/20/06

%Intialize Variables
p1 = ptm;
p2 = p1+1;
freq1=fc;
mag1=fc_mag;
cond=0;  %this variable stays 0 until a PW is found
while cond == 0
    while sigfft(p1)>sigfft(p2)  %find peaks
        p1=p1+1;
        p2=p2+1;
    end
    while sigfft(p2)>sigfft(p1)  %
        p1=p1+1;
        p2=p2+1;
    end
    freq2=freq(p1);
mag2=sigfft(p1);
    if freq2-freq1 < avg_fdel + 30
        if mag2 > 0.45*mag1  %Check if legitimate peak
            mag1 = mag2;
            freq1 = freq2;
            p1=p1+1;
            p2=p2+1;
        else
            p1=p1+1;
        end
    end
The following is the code for the visual basic front end.

Option Explicit 'Declare all variables.
Option Base 1 'Lower array index = 1.
Public Matlab As Object
Public Result As String
Public fc As Double, maxamp As Double, PRP As Double
Public pulse_width As Double
Public sigdet As Integer, per_PW As Integer, pulse_width_fin As Integer
'Created by Ryan Hamor 1/16/06
'Form version 1.0 added ability to connect and disconnect from Automation Server
'Form version 1.1 added Subs, cmdLoadData and cmdSig_Proc to test m-files 01/19/06
'Form version 1.2 Removed testing sub routines and added the cmdIntAD and cmdScan_Click()
'this version has provisions to automatically scan once the LO setting are entered.
'03/07/06
'From version 1.3 Added code to run second detection process 3/8/06
'Form version 1.4 Broke down code into subroutines for final testing and adding modes of operation. 4/19/06

Private Sub cmdIntAD_Click()
'Intializes the 3005USB A/D card through matlab

'Set working directory of matlab
Result = Matlab.Execute(˝cd c:\RadarRec˝) 'Set the working directory
Result = Matlab.Execute(˝intad˝) 'setup AD

End Sub

Private Sub cmdQuit_Click()

'Quit out of matlab server
Matlab.Quit
Private Sub cmdScan_Click()

' This Subroutine starts the automatic scanning process and does not stop until the it is done scanning
Scan:
Call frmLO.IncLO
Call Detect_Level1
If sigdet = 1 Then
    Call Detect_Level2
    If sigdet = 2 Then
        Call Get_Char
        GoTo Done
    Else
        GoTo Scan
    End If
Else
    GoTo Scan
End If

Done:

End Sub

Private Sub cmdsetup_Click()

' Create Matlab Server Shared
Set Matlab = CreateObject("Matlab.Application")

End Sub

Private Sub Detect_Level1()

' This will sample the data and determine if the there is a signal with appropriate level present.
Result = Matlab.Execute("sample") ' sample signal
maxamp = Matlab.GetVariable("maxamp", "base") ' get maximum value of signal from matlab

If maxamp >= 0.225 Then ' Check that there is a signal present
    txtstatus.Text = "Signal Found Determining if Pulse Waverform..."
    sigdet = 1
Else
    sigdet = 0
End If
Private Sub Detect_Level2()
    Dim det As Integer

    Result = Matlab.Execute("det_level2")
    det = Matlab.GetVariable("det", "base")
    If det = 1 Then
        sigdet = 2
    Else
        sigdet = 0
    End If
End Sub

Private Sub Get_Char()
    Dim curfreq As Double
    'This subroutine gets characteristics of the waveform and displays them.
    Result = Matlab.Execute("detect") 'M-file to see if real pulse waveform and find PRR

    txtstatus.Text = "Pulse Waveform Found Please Wait for Characteristics..."
    Result = Matlab.Execute("PW") 'Executes m-file to find signal Pulse Width
    fc = Matlab.GetVariable("fc", "base")
    PRP = Matlab.GetVariable("PRR", "base")
    pulse_width = Matlab.GetVariable("PWt", "base")
    per_PW = pulse_width / PRP * 100
    pulse_width_fin = pulse_width * 1000000 'convert to micro seconds
    PRP = PRP * 1000 'convert to milliseconds
    curfreq = CSng(frmLO.txtFWLO_MHZ.Text) + (0.000001 * fc)

    txtPW.Text = pulse_width_fin 'Update text boxes
    txtPW_percent.Text = per_PW
    txtfc.Text = fc
    txtPRR.Text = PRP
    txtRFfreq.Text = curfreq
End Sub
Private Sub Form_Load()

    Height = 7200
    Width = 9700
    Left = 100
    Top = 100

End Sub
### Appendix B (Test Results)

<table>
<thead>
<tr>
<th><strong>Scan</strong></th>
<th><strong>Digital Radar</strong></th>
<th><strong>Test Sheet</strong></th>
<th><strong>Ryan Hauer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Conditions</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LO Amp</td>
<td>AAA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF Amp</td>
<td>6:10 dBi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LO Step Size</td>
<td>200 kHz</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actual Signal Characteristics</th>
<th>Detected Signal Characteristics</th>
<th>Computed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF Freq (MHz)</td>
<td>PRR (μs)</td>
<td>PW (mS)</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>70</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>90</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>120</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>140</td>
<td>1.9</td>
<td>200</td>
</tr>
<tr>
<td>160</td>
<td>1.8</td>
<td>160</td>
</tr>
<tr>
<td>180</td>
<td>1.7</td>
<td>160</td>
</tr>
<tr>
<td>200</td>
<td>1.6</td>
<td>180</td>
</tr>
<tr>
<td>220</td>
<td>1.5</td>
<td>150</td>
</tr>
<tr>
<td>240</td>
<td>1.4</td>
<td>150</td>
</tr>
<tr>
<td>260</td>
<td>1.3</td>
<td>150</td>
</tr>
<tr>
<td>280</td>
<td>1.2</td>
<td>150</td>
</tr>
<tr>
<td>300</td>
<td>1.1</td>
<td>150</td>
</tr>
<tr>
<td>320</td>
<td>1.0</td>
<td>150</td>
</tr>
<tr>
<td>340</td>
<td>0.9</td>
<td>150</td>
</tr>
<tr>
<td>360</td>
<td>0.8</td>
<td>150</td>
</tr>
<tr>
<td>380</td>
<td>0.7</td>
<td>150</td>
</tr>
<tr>
<td>400</td>
<td>0.6</td>
<td>150</td>
</tr>
<tr>
<td>420</td>
<td>0.5</td>
<td>150</td>
</tr>
<tr>
<td>440</td>
<td>0.4</td>
<td>150</td>
</tr>
<tr>
<td>460</td>
<td>0.3</td>
<td>52</td>
</tr>
<tr>
<td>480</td>
<td>0.2</td>
<td>52</td>
</tr>
<tr>
<td>500</td>
<td>0.1</td>
<td>24</td>
</tr>
<tr>
<td>520</td>
<td>0.8</td>
<td>112</td>
</tr>
<tr>
<td>540</td>
<td>1.7</td>
<td>420</td>
</tr>
<tr>
<td>560</td>
<td>1.4</td>
<td>246</td>
</tr>
<tr>
<td>580</td>
<td>0.5</td>
<td>76</td>
</tr>
<tr>
<td>600</td>
<td>1.1</td>
<td>128</td>
</tr>
<tr>
<td>620</td>
<td>1.8</td>
<td>312</td>
</tr>
<tr>
<td>Max PW Testing</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>70</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>90</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>110</td>
<td>2</td>
<td>550</td>
</tr>
<tr>
<td>130</td>
<td>2</td>
<td>524</td>
</tr>
<tr>
<td>150</td>
<td>1</td>
<td>400</td>
</tr>
</tbody>
</table>

PRR too low to call!