```
% PulseMeasV31.m
% Version 3.1,  31 Aug. 2009
% J.R.Andrews, KH6HTV, Picosecond Pulse Labs, Boulder, Colorado, USA
% inputs waveform from disc A:  plots it, determines pulse parameters
% Vmax, Vmin, 0%bottom,100%top, Vptp, amplitude, Over/UnderShoot,
% polarity, type, delay, duration, risetime, & falltime
% The Histogram method is used to determine the 0% & 100% levels.
% Version 1.0 was written by Martin VanPelt, PSPL in 1989
% It consisted of three, seperate, special MatLab 'functions' called
% TOPBOT.m, FWHM.m & RISETIME.m
% Version 2.0 written by J.R.Andrews, WA0NHD, 7/20/04, 9/10/04
% Consolidated VanPelt's functions, added pulse type & polarity selection,
% and waveform plotting with measurement notation.  Added optional, ext.
% manual input of 0% & 100% levels.  Also optional 20%-80% rise/falltime
% Version 3.0 written by J.R.Andrews, WA0NHD, Dec. 2004
% added capability to measure all pulses in a multiple pulse train.
% added display of histogram used to determine 0% & 100% levels.
% Version 3.1 written by Jim Andrews, KH6HTV, 31 Aug. 2009
% corrected bug which caused program to fail for large data N > 2K
disp('  ')
disp('PulseMeasV31.m --- MatLab Program')
disp('Version 3.1, J.R.Andrews, KH6HTV, 31 Aug 2009')
disp('Picosecond Pulse Labs, Boulder, Colorado')
disp('Input waveform comes from an oscilloscope, another MatLab program,')
disp('or other source as a *.txt file via disc drive A: or C:')
disp('Program makes pulse measurements on step waveform, single pulse')
disp('or multiple pulse train.')
disp('  ')
clear
disp('Please use the keyboard to respond to the following questions.')
drive = input('Data in via drive A: or C:?  (1=A: 3=C:)   ');
if drive == 1
    disp('Data in will be as *.txt file via floppy disc in drive A:')
else
    disp('Data in will be as *.txt file in current C: drive directory')
end
disp(' ')
fname = input('Enter Data File Name =   ','s');
if drive == 1
    dname = ['A:', fname, '.txt'];
else
    dname = [fname, '.txt'];
end
v = load(dname);
Tw = input('Enter Time Window in NanoSeconds =   ');
N = length(v); % = # of data points
```

```
disp('Number of data points =')
disp(N)
dt=Tw/N; % (in ns)
disp('sample spacing, dt, in NanoSeconds = ')
disp(dt)
for i=1:N
    t(i)=(i-1)*dt;
end
plot(t,v)
grid
xlabel('time in ns')
ylabel('Volts')
title('Input Pulse')
disp('plot of input pulse --- press any key to continue')
pause
disp('Program normally determines rise/falltimes between the 10% & 90%
levels')
T2080 = input('Would you prefer to measure 20%/80% rise/falltimes? (1=yes,
0=no) ');
if T2080 == 1
    Trf1 = 0.2;
    Trf9 = 0.8;
else
    Trf1 = 0.1;
    Trf9 = 0.9;
end
disp('Program automatically determines 0% & 100% levels using Histogram
method')
levman = input('Do you want to manually enter 0% & 100% levels? (1=yes,
0=no)  ');
if levman == 1
    top = input('Topline  100% level in Volts = ');
    bot = input('Bottomline 0% level in Volts = ');
    maxpoint = max(v);
    minpoint = min(v);
else
% Martin VanPelt's TOPBOT.m program, PSPL, 1989
% modified slightly by jra,7/20/04
% TOPBOT  This subroutine returns the top & bottom values of a voltage
%         waveform.  It does this by histograming a waveform.  It divides
%         the whole waveform in "numbuckets" buckets.  It assigns each
point
%         into its appropriate voltage bucket.  These are counted in the
%         matrix "hits".  Looking at the top & bottom 50% only, the 2
buckets
%         with the most hits are the top & bottom as long as they have at
```

```
%         least "hitlimit" of the hits.  If not, the max and/or the min
%         are assigned to the top or bottom.
numbuckets = 2000;
hitlimit = 0.005;  % Martin used 0.01 (1%), jra changed to 0.005 (0.5%)
9/10/04
maxpoint = max(v);
minpoint = min(v);
bucketsize = (maxpoint-minpoint)/(numbuckets-1);
lowend = minpoint - (bucketsize/2);
hits = zeros(1,numbuckets); % zeros initializes hits arrary to all 0s
for i = 1:length(v)
        bucket = ceil((v(i)-lowend)/bucketsize); %ceil rounds up to integer
        hits(bucket) = hits(bucket) + 1;
end
% index50 = fix(0.5*length(v)); % fix rounds down to integer
% note: Martin used top & bottom 30% only. jra changed to 50%
% note index50 = fix(0.5*length(v)); lead to problems with large arrays
% 8/31/09, jra - modified the index to not be a function of N, but instead
% restrict search to top 45% & bottom 45%
index45 = fix(0.45*numbuckets);
maxarray = hits(numbuckets:-1:index45);
minarray = hits(1:index45);
[tophits,topindex] = max(maxarray); %max stores indicies of max. value in
array
[bothits,botindex] = max(minarray);
if tophits >= (hitlimit*length(v))
        top = maxpoint - (topindex-1)*bucketsize;
else
        top = maxpoint;
end
if bothits >= (hitlimit*length(v))
        bot = minpoint + (botindex-1)*bucketsize;
else
        bot = minpoint;
end
for i = 1:numbuckets
    vh(i) = minpoint + (i-1)*bucketsize;
end
plot(hits,vh)
xlabel('Histogram  # of hits')
ylabel('Volts')
title(['# of Buckets= ',num2str(numbuckets),'  BucketSize= ',...
        num2str(bucketsize),'V  BotHits= ',num2str(bothits),...
        '  TopHits= ',num2str(tophits)])
pause
Hitsmax = bothits;
```

```
if tophits > bothits
    Hitsmax = tophits;
end
for i = 1:numbuckets
    Hitsplot(i) = 0.2*Tw*(-1.1+hits(i)/Hitsmax);
end
plot(Hitsplot,vh,'g',t,bot,'r',t,top,'r',t,v,'b')
grid
xlabel('time in ns')
ylabel('Volts')
title('waveform (blue), histogram (green), 0% & 100% (red)')
pause
end % end for levman
%end of Martin's TOPBOT.m subroutine
% calc. voltages
Vmax = maxpoint;
Vmin = minpoint;
Vptp = maxpoint - minpoint;
Amp = top - bot;
Oshoot = (maxpoint-top)*100/Amp; % in %
Ushoot = (bot-minpoint)*100/Amp; % in %
% start sub-routine program to find 50% crossing points
lev50 = bot + 0.5*Amp;  % 50% level
disp(' ')
disp('Pulse polarity is determined relative to initial point v(t=0).')
if v(1) < lev50
    Pol = +1;
    disp('Pulse Polarity = Positive')
else
    Pol = -1;
    disp('Pulse Polarity = Negative')
end
k = 1;
T50I(1) = 1;  % closest integer after crossing 50% level
S50I(1) = Pol;  % corresponding slope at T50I, +1=pos(rise), -1=neg(fall)
for i=2:N
    if sign(v(i)-lev50) ~= sign(v(i-1)-lev50)
        T50I(k) = i;
        S50I(k) = sign(v(i)-lev50);
        k = k+1;
    end
end
N50 = length(T50I); % total # of 50% crossings
for i=1:N50
    k = T50I(i);
    interp = (lev50-v(k-1)) / (v(k)-v(k-1));
```

```
    t50I(i) = t(k-1) + interp*(t(k)-t(k-1)); % actual 50% crossing times
end
delay = t50I(1);
% initialize pulse time parameters to NaN
tr = NaN;
tf = NaN;
duration = NaN;
dur = NaN;
period = NaN;
per = NaN;
freq = NaN;
dutycycle = NaN;
% determine type of pulse
if N50 == 1
    Ptype = 1;
    disp('Step-Like Pulse, no second edge')
    duration = NaN;
end
if N50 == 2
    Ptype = 2;
    disp('Pulse is Impulse or Rectangular with 2 edges')
    duration = t50I(2) - t50I(1);
    period = NaN;
    freq = NaN;
end
if N50 > 2
    Ptype = 3;
    disp('Waveform is pulse train with multiple edges')
    duration = t50I(2) - t50I(1); % units are ns
    period = t50I(3) - t50I(1);  % units are ns
    freq = 1/period;  % units are GHz
    dutycycle = (duration/period)*100; % units are %
end
if N50 > 3
    for i = 1:(N50-2)
        per(i) = t50I(i+2) - t50I(i);
    end
    permin = min(per);
    PRR = 1/permin;  % pulse rep. rate
    for i = 1:fix(N50/2)
        dur(i) = t50I(2*i) - t50I(2*i-1);
    end
end
% end 50% edge sub-routine
% start Trise & Tfall sub-routine
% major modification of Martin VanPelt's RISETIME.m program, PSPL 1989
```

```
% now start search for 10% & 90% crossings from the 50% points.
lev10 = bot + Trf1*Amp;  % i.e. 10% level
lev90 = bot + Trf9*Amp;  % i.e. 90% level
tr(1:N50) = NaN;
tf(1:N50) = NaN;
for k=N50:-1:1 % examine each edge
    if S50I(k) == 1 % i.e. pos. slope, determine risetime
        % find 90% crossing
        i = T50I(k);
        while (sign(v(i)-lev90) == sign(v(i-1)-lev90)) & (i < N)
            i = i+1;
        end
        interp = (lev90-v(i-1)) / (v(i)-v(i-1));
        t90r = t(i-1) + interp*(t(i)-t(i-1));
        % find 10% crossing
        j = T50I(k);
        while (sign(v(j-1)-lev10) == sign(v(j)-lev10)) & (j > 2)
            j = j-1;
        end
        interp = (lev10-v(j-1)) / (v(j)-v(j-1));
        t10r = t(j-1) + interp*(t(j)-t(j-1));
        if (i<N)&(j>2)
            tr(k) = t90r - t10r;
        end
    end
    if S50I(k) == -1 % i.e. neg. slope, determine falltime
        % find 10% crossing
        i = T50I(k);
        while (sign(v(i)-lev10) == sign(v(i-1)-lev10)) & (i < N)
            i = i+1;
        end
        interp = (lev10-v(i-1)) / (v(i)-v(i-1));
        t10f = t(i-1) + interp*(t(i)-t(i-1));
        % find 90% crossing
        j = T50I(k);
        while (sign(v(j-1)-lev90) == sign(v(j)-lev90)) & (j > 2)
            j = j-1;
        end
        interp = (lev90-v(j-1)) / (v(j)-v(j-1));
        t90f = t(j-1) + interp*(t(j)-t(j-1));
        if (i<N)&(j>2)
            tf(k) = t10f - t90f;
        end
    end
end
tr = tr(~isnan(tr)); % used to remove NaN from risetime file
```

```
tf = tf(~isnan(tf));
% end of Trise & Tfall sub-routine
disp('  ')
disp('    PULSE MEASUREMENT RESULTS')
disp('  ')
Head1 = ['    Bottom(0%)','Top(100%)',' Amplitude',' Vptp',...
        '      Vmax','      Vmin'];
Head2 = ['    OverShoot & UnderShoot (in %)'];
Head3 = ['    Delay   ','  Risetime'];
Head4 = ['    Delay   ','  Falltime'];
Head5 = ['    Delay   ','  Duration','  Risetime','  Falltime'];
Head6 = ['    Delay   ','  Duration','  Risetime','  Falltime',...
        '  Period',' Rep.Rate(GHz)',' DutyCycle(%)'];
Data1 = [bot top Amp Vptp Vmax Vmin];
Data2 = [Oshoot Ushoot];
disp('    Pulse Voltages (in Volts)')
disp(Head1)
disp(Data1)
disp(Head2)
disp(Data2)
disp('    Inital Pulse Times (in ns)')
if (Ptype == 1) & (Pol == 1)  % i.e. pos. step pulse
    disp(Head3)
    Data3 = [delay tr(1)];
    disp(Data3)
end
if (Ptype == 1) & (Pol == -1)  % i.e. neg step pulse
    disp(Head4)
    Data4 = [delay tf(1)];
    disp(Data4)
end
if Ptype == 2  % i.e. single rect. pulse
    disp(Head5)
    Data5 = [delay duration tr(1) tf(1)];
    disp(Data5)
end
if Ptype == 3  % i.e. multiple pulses
    disp(Head6)
    Data6 = [delay duration tr(1) tf(1) period freq dutycycle];
    disp(Data6)
end
if N50 > 3
    disp('    Multiple Pulse Data Train')
    disp('    data from left to right as time increases')
    disp('    Durations of Pulses (in ns)')
    disp(dur)
```

```
    disp('    Risetimes of Pulses (in ns)')
    disp(tr)
    disp('    Falltimes of Pulses (in ns)')
    disp(tf)
    disp('    Periods of Pulses   (in ns)')
    disp(per)
    disp('    max. Rep. Rate (in GHz)')
    disp(PRR)
end
% plot measurement results
if (Ptype == 1) & (Pol == 1)  % i.e. pos. step pulse
    plot(t,bot,'r',t,top,'r',t,v,'b',t10r,lev10,'+r',t50I(1),lev50,'+r',...
        t90r,lev90,'+r')
    title(['Amplitude = ',num2str(Amp),'V    Risetime = ',num2str(tr
(1)),'ns'])
end
if (Ptype == 1) & (Pol == -1)  % i.e. neg step pulse
    plot(t,bot,'r',t,top,'r',t,v,'b',t10f,lev10,'+r',t50I(1),lev50,'+r',...
        t90f,lev90,'+r')
    title(['Amplitude = ',num2str(Amp),'V    Falltime = ',num2str(tf
(1)),'ns'])
end
if Ptype == 2  % i.e. single rect. pulse
    plot(t,bot,'r',t,top,'r',t,v,'b',t10r,lev10,'+r',t50I(1),lev50,'+r',...
        t90r,lev90,'+r',t90f,lev90,'+r',t50I(2),lev50,'+r',t10f,lev10,'+r')
    title(['Amp = ',num2str(Amp),'V  Tr = ',num2str(tr(1)),'ns  Tf = ',...
        num2str(tf(1)),'ns  Duration = ',num2str(duration),'ns'])
end
if Ptype == 3  % i.e. multiple pulses
    plot(t,bot,'r',t,top,'r',t,v,'b',t10r,lev10,'+r',t50I(1),lev50,'+r',...
        t90r,lev90,'+r',t90f,lev90,'+r',t50I
(2),lev50,'+r',t10f,lev10,'+r',...
        t50I(3),lev50,'+r')
    title(['Amp = ',num2str(Amp),'V  Tr = ',num2str(tr(1)),'ns  Tf = ',...
        num2str(tf(1)),'ns  Dur = ',num2str(duration),'ns  Period = ',...
        num2str(period),'ns'])
end
grid
xlabel('time in ns')
ylabel('Volts')
disp('Waveform plotted with measurement results added')
disp('Use "Edit -- Axes Properties" to alter plot scales, titles, etc')
disp('  ')
disp('end of PulseMeasV31.m program')
```