Decoding NRZ Data

Using XDEV Customization to convert to digital waveform

Non-Return to Zero (NRZ) signals use two voltage levels (high and low) to represent logic one and logic zero values. Traditional oscilloscopes can capture and view these waveforms but do not have the capability to decode waveforms into their digital equivalents. Oscilloscopes with XDEV custom functionality can now decode these waveforms.

In Figure 1, a block of NRZ data is acquired as Function F1 and the parameter TIE@lv (Time Interval Error) is applied as parameter P1. The Virtual Clock capability of TIE reports the underlying bitrate as 2.488 Gb/s, which corresponds to this OC-48 datastream. Function F2 allows for a user-defined Matlab script to automatically decode each waveform as shown in Figure 2. The algorithm reads the base frequency from the TIE parameter to locate boundaries of each unit interval. As the algorithm processes each waveform edge, it appends the logic value into a buffer of datastream values. When the loop completes, the digitally-decoded waveform and its statistics are displayed as well as recorded to a file. This process will repeat for each acquisition allowing rapid in-line waveform conversion.

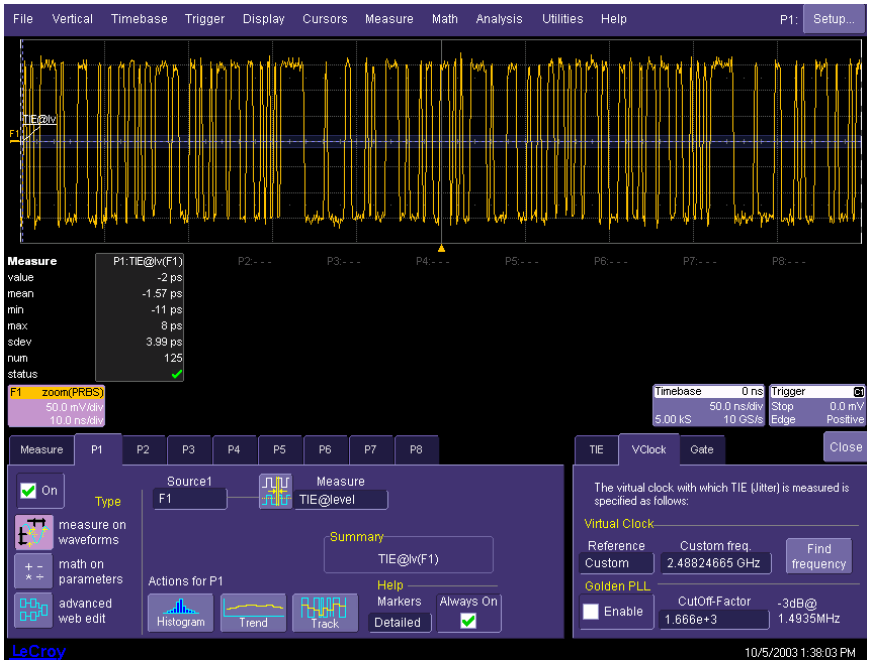

Figure 1 The Virtual Clock tab of the TIE@lv parameter is used to identify the data rate of the pseudo-random bit stream.

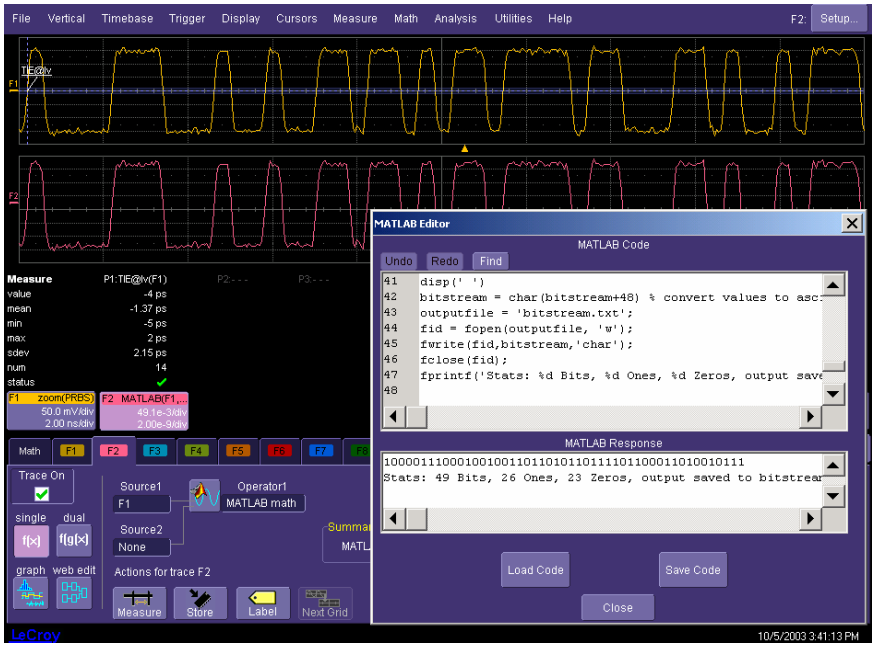

Figure 2: XDEV custom function reports digitally decoded waveform.

```
clear bitstream; % erase previous bitstream values
WformOut = WformIn1 - mean(WformIn1);  % offset the waveform to place the mean value
at zero
comclient off % needed for Matlab rev 6.5 - delete this line if using any other Mat-
lab revision
h = actxserver('LeCroy.WaveMasterApplication'); % establish ActiveX control between
the scope and Matlab
xincr = h.Math.F1.Out.Result.HorizontalPerStep; % sample resolution
basefreq = double(get(h.Measure.P1.Operator.BaseFrequency,'Value')); % TIE frequency
determination
baseper = 1/basefreq;
samplesperbit = ceil(baseper/xincr);

n = find(diff(WformOut>0)); % locate position of edges on the NRZ data
numberofedges = length(n);

bitcounter = 0; % initialize a bit counter
virtualclockposition = n(1); %
numones = 0; % keep track of total number of ones identified
numzeros = 0; % keep track of total number of zeros identified
for i=1:(numberofedges-1)
   virtualclockposition = n(i); % set virtual clock to coincide with NRZ edge tran-
sitions
   bitswide = round((n(i+1)-n(i))/samplesperbit);  % determine how many bits occur
between edges

   for j=1:bitswide % for example if there are two zeros in a row between edges,
execute this loop twice to check both bits
      bitcounter = bitcounter + 1; % keep track of how many bits have been checked
so far

      if (WformOut(virtualclockposition + ceil(samplesperbit/2)) > 0) % the ampli-
tude is high, this bit is a One
         bitstream(bitcounter) = 1; % log the value of this bit as a zero or one for
future reference
         numones = numones + 1; % increment the total number of identified One val-
ues
      else
         bitstream(bitcounter) = 0; % the amplitude low, this bit is a Zero
         numzeros = numzeros + 1; % increment the total number of identified Zero
values
      end
   end
end

%% SAVE BITSTREAM TO A FILE
disp(' ')
bitstream = char(bitstream+48) % convert values to ascii
outputfile = 'bitstream.txt';
fid = fopen(outputfile, 'w');
fwrite(fid,bitstream,'char');
fclose(fid);
fprintf('Stats: %d Bits, %d Ones, %d Zeros, output saved to %s\n',
length(bitstream), numones, numzeros, outputfile);
```

Figure 3: Matlab algorithm decodes NRZ datastream