In[25]:=
```
SetDirectory["~seb/Desktop/agilent/"];
Namelist = Drop[FileNames[], -1];
Namelist // Length;
nfiles = %
```

Out[28]= 2

In[29]:=
```
fileno = 2
```

Out[29]= 2

In[30]:=
```
Print[Namelist]
```

```
{Agilent-2013-05-24-003-0.10.txt, Agilent.dat}
```

In[31]:=
```
Print [Namelist[[fileno]]]
```

```
Agilent.dat
```

In[32]:=
```
Timing[scopedata = Import[Namelist[[fileno]], "tsv"];]
```

Out[32]= {781.402011, Null}

In[33]:=
```
Dimensions[scopedata]
```

Out[33]= {29 200, 4005}

Name the output file

In[34]:=
```
outfile = StringReplace[Namelist[[fileno]],
  {".dat" → ".csv", "Agilent" → "Agilent_compress"}]
```

Out[34]= Agilent_compress.csv

Pick 1 event for displaying.

In[222]:=
```
eventno = 4006;
index = (eventno - 1) * 4;
firstrec = scopedata[[1 + index]]; secndrec = scopedata[[2 + index]];
thrdrec = scopedata[[3 + index]]; frthrec = scopedata[[4 + index]];
```

In[225]:=
```
filestruct = Dimensions[scopedata];
records = Dimensions[scopedata][[1]];
nevents = Dimensions[scopedata][[1]] / 4;
reclen = Dimensions[scopedata][[2]];
Print[{"There are   ", nevents,
  " events and the data dimensionality is  ", filestruct}]
```

```
{There are   , 7300,  events and the data dimensionality is  , {29 200, 4005}}
```

Channel 1 input is Scintillator (negative pulse). Channel 2 and 3 are 64 mm^2 APDs. Channel 4 is 4 mm^2 APD (negative).

In[230]:=
```
Print[{"channels are read in the following order:",
  scopedata[[1, 1]], scopedata[[2, 1]], scopedata[[3, 1]],
  scopedata[[4, 1]], scopedata[[5, 1]], scopedata[[6, 1]], ",etc"}]
```

```
{channels are read in the following order:, 1, 2, 3, 4, 1, 2, ,etc}
```

Set up to plot scope traces for a particular event (eventno).

In[231]:=
```
chan1 = firstrec[[1]]; chan2 = secndrec[[1]];
chan3 = thrdrec[[1]]; chan4 = frthrec[[1]];
dt = firstrec[[2]]
```

Out[232]= $5. \times 10^{-11}$

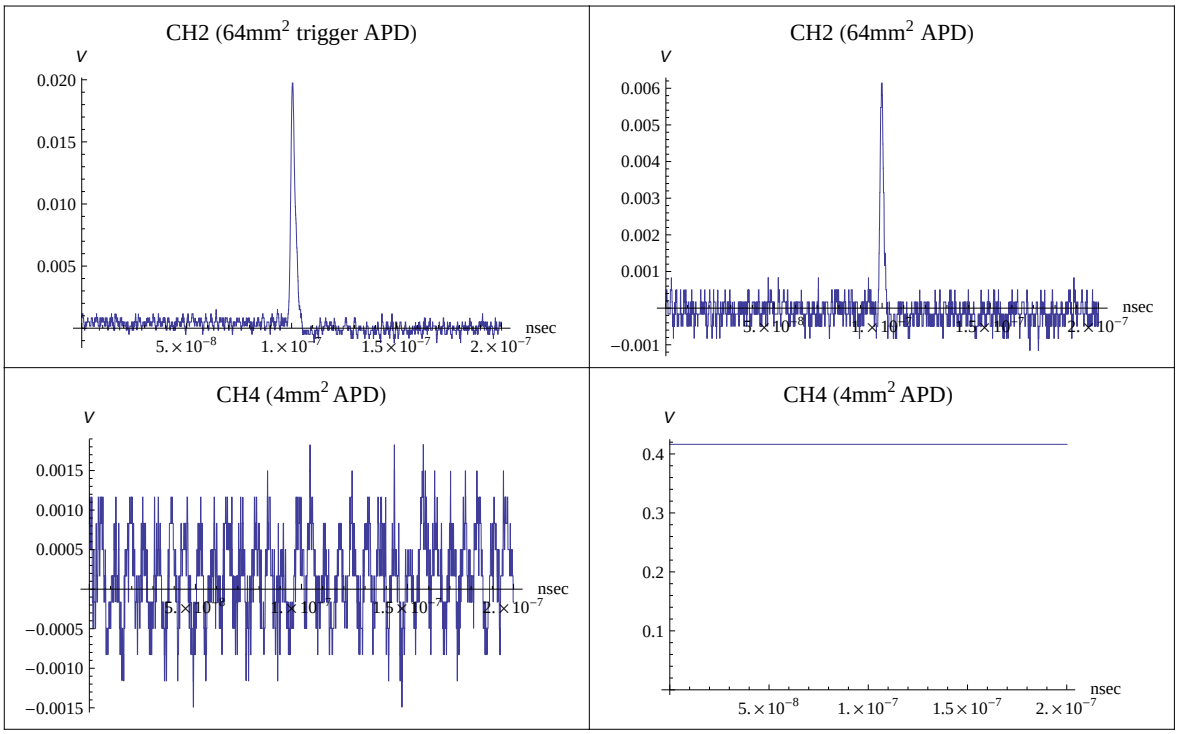
In[233]:=
```
v1 = Drop[firstrec, 2]; v2 = Drop[secndrec, 2];
v3 = Drop[thrdrec, 2]; v4 = Drop[frthrec, 2];
npts = Dimensions[v1][[1]]
Print[Style[{" 4 Channels read out in ", dt, " nanosecond bins from ",
   tstart, " to ", tstart + npts * dt " nanoseconds "}, 18, Red]]
```

Out[234]= 4003

$\{$ 4 Channels read out in , $5. \times 10^{-11}$,
nanosecond bins from , $5. \times 10^{-11}$, to ,
$5. \times 10^{-11} + 2.0015 \times 10^{-7}$ nanoseconds $\}$

In[236]:=
```
time = Table[(i - 1) * dt, {i, npts}];
GraphicsGrid[
 {{ListPlot[Transpose[{time, v1}], Joined → True, AxesLabel → {nsec, V},
    PlotRange → Full, PlotLabel → "CH2 (64mm^2 trigger APD)"],
   ListPlot[Transpose[{time, v2}], Joined → True, AxesLabel → {nsec, V},
    PlotRange -> Full, PlotLabel → "CH2 (64mm^2 APD)"]},
  {ListPlot[Transpose[{time, v3}], Joined → True, AxesLabel → {nsec, V},
    PlotRange → Full, PlotLabel → "CH4 (4mm^2 APD)"], ListPlot[
    Transpose[{time, v4}], Joined → True, AxesLabel → {nsec, V}, PlotRange → Full,
    PlotLabel → "CH4 (4mm^2 APD)"]}}, Frame → All, ImageSize → Full]
```

Out[237]=



In[101]:=
```
timefun[tbin_] := (tbin - 1) * dt * 10^9
bin[t_] := (t) / dt * 10^(-9) + 1
```

For all channels go through and generate histograms of pulse maximum, baseline offset, noise, etc.

In[103]:=
```
(*chmax=ConstantArray[0,{4,1000}];choff=ConstantArray[0,{4,1000}];
chnoise=ConstantArray[0,{4,1000}];*)
chmax1 = ConstantArray[0, 7300]; chmax2 = ConstantArray[0, 7300];
chmax3 = ConstantArray[0, 7300]; chmax4 = ConstantArray[0, 7300];
choff1 = ConstantArray[0, 7300]; choff2 = ConstantArray[0, 7300];
choff3 = ConstantArray[0, 7300]; choff4 = ConstantArray[0, 7300];
chnoise1 = ConstantArray[0, 7300]; chnoise2 = ConstantArray[0, 7300];
chnoise3 = ConstantArray[0, 7300]; chnoise4 = ConstantArray[0, 7300];
hvect = ConstantArray[0, 7300];
ln = bin[60]
hn = bin[90]
sigmax = bin[120]
```

Out[107]= 1201.

Out[108]= 1801.

Out[109]= 2401.

In[110]:=
```
scintmin = bin[160]
```

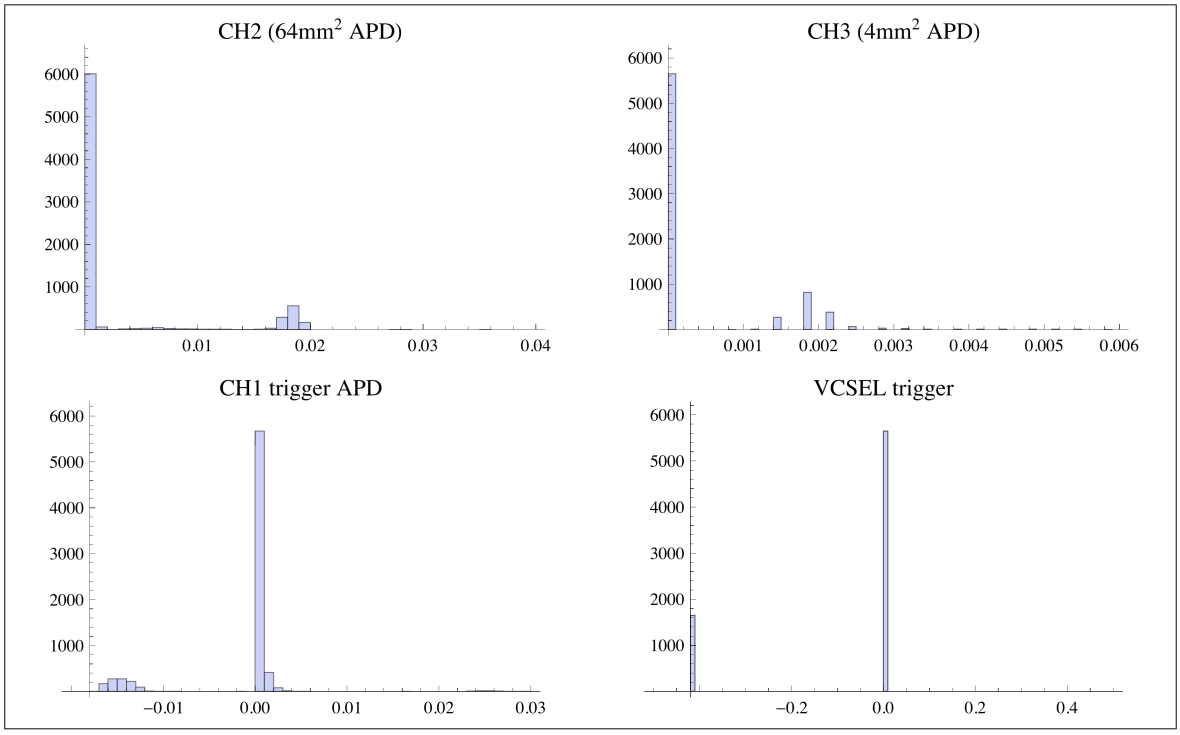
Out[110]= 3201.

In[111]:=
```
records = 4000;
Do[
 vsig = Drop[scopedata[[i]], 3];
 ichan = Mod[i, 4];
 If[ichan == 0, ichan = 4];
 iev = IntegerPart[(i - 1) / 4] + 1;
 If[ichan == 1, Goto[ch1]];
 If[ichan == 2, Goto[ch2]];
 If[ichan == 3, Goto[ch3]];
 If[ichan == 4, Goto[ch4], Goto[end]];
 Label[ch1];
 chmax1[[iev]] = Max[-Drop[vsig, 1800]];
 choff1[[iev]] = Mean[Take[vsig, 1800]];
 chnoise1[[iev]] = RootMeanSquare[Take[vsig, 1800]];
 Goto[end];
 Label[ch2];
 chmax2[[iev]] = Max[Drop[vsig, 1800]];
 choff2[[iev]] = Mean[Take[vsig, 1800]];
 chnoise2[[iev]] = RootMeanSquare[Take[vsig, 1800]];
 Goto[end];
 Label[ch3];
 chmax3[[iev]] = Max[Drop[vsig, 1800]];
 choff3[[iev]] = Mean[Take[vsig, 1800]];
 chnoise3[[iev]] = RootMeanSquare[Take[vsig, 1800]];
 Goto[end];
 Label[ch4];
 chmax4[[iev]] = Max[-Drop[vsig, 1800]];
 choff4[[iev]] = Mean[Take[vsig, 1800]];
 chnoise4[[iev]] = RootMeanSquare[Take[vsig, 1800]];
 Label[end];
 , {i, 400, 7000}]
```

Distribution of Pulse Peak Amplitude

In[115]:=
```
GraphicsGrid[{{Histogram[chmax2, {0.0, .04, .001}, PlotLabel → "CH2 (64mm^2 APD)"],
   Histogram[chmax3, {0.0, .006, .0001}, PlotLabel → "CH3 (4mm^2 APD)"]},
  {Histogram[chmax1, {-.02, .03, .001}, PlotLabel → "CH1 trigger APD"],
   Histogram[chmax4, {-.5, .5, .01}, PlotLabel → "VCSEL trigger"]}},
 ImageSize → Full, Frame → True]
```

Out[115]=



Distribution of Noise

In[118]:=
```
GraphicsGrid[{{Histogram[chnoise2, PlotLabel → "Ch2"],
    Histogram[chnoise3, "LogCount", PlotLabel → "Ch3"]},
  {Histogram[chnoise1, PlotLabel → "Ch1"], Histogram[chnoise4,
    "LogCount", PlotLabel → "Ch4"]}}, ImageSize → Large, Frame → True]
```

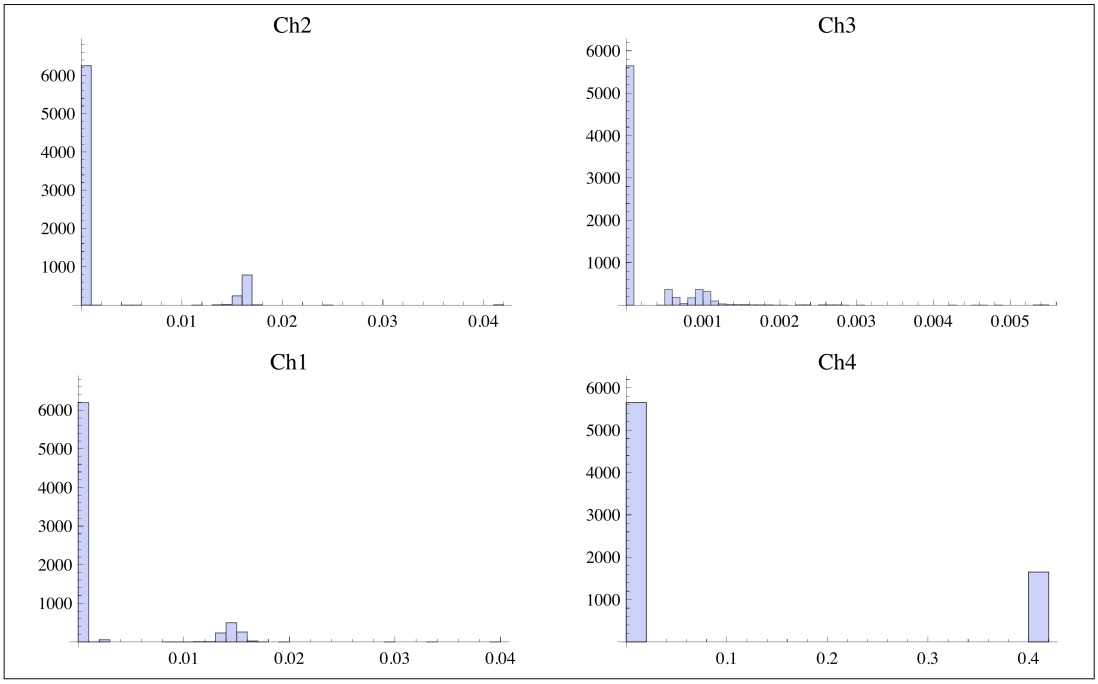
Histogram::hbins :

The bin specification LogCount cannot be used to determine either how many or which bins to use. ≫

Histogram::hbins :

The bin specification LogCount cannot be used to determine either how many or which bins to use. ≫

Out[118]=



Average Noise (mV)

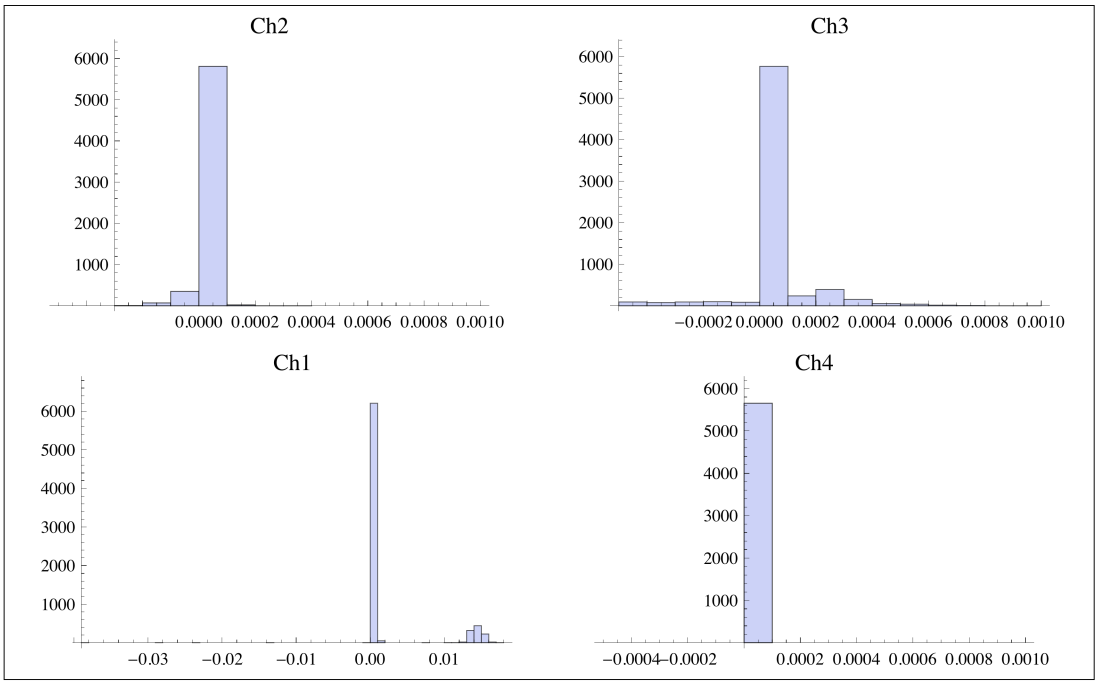
In[119]:=
```
Print[Column[{1000 * Mean[chnoise2],
    1000 * Mean[chnoise3], 1000 * Mean[chnoise1], 1000 * Mean[chnoise4]}]]
```

```
2.36896
0.206671
2.15625
94.1562
```

Baseline Offsets(mV)

In[120]:=
```
GraphicsGrid[{{Histogram[choff2, {-.0005, .001, .0001}, PlotLabel → "Ch2"],
    Histogram[choff3, {-.0005, .001, .0001}, PlotLabel → "Ch3"]},
  {Histogram[choff1, PlotLabel → "Ch1"], Histogram[choff4,
    {-.0005, .001, .0001}, PlotLabel → "Ch4"]}}, ImageSize → Large, Frame → True]
```

Out[120]=



Average Baseline Offset (mV)

In[121]:=
```
Print[Column[{1000 * Mean[choff2],
    1000 * Mean[choff3], 1000 * Mean[choff1], 1000 * Mean[choff4]}]]
```

```
2.30804
0.0041325
2.07933
94.1559
```

RMS of Baseline Offset (mV)

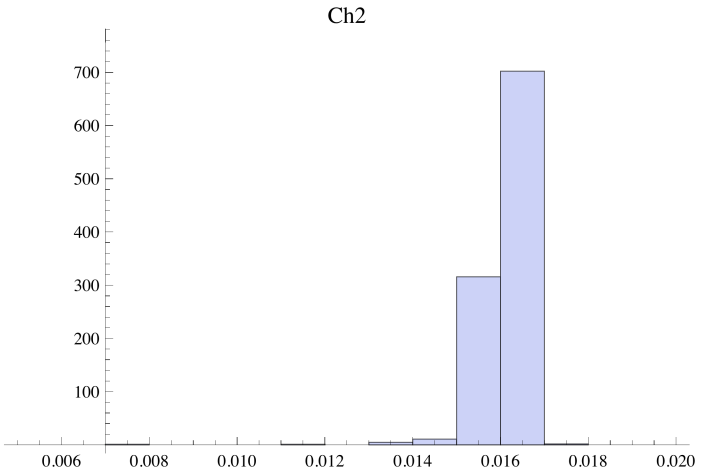
In[122]:=
```
Print[Column[{1000 * RootMeanSquare[choff2], 1000 * RootMeanSquare[choff3],
    1000 * RootMeanSquare[choff1], 1000 * RootMeanSquare[choff4]}]]
```

```
6.18323
0.168553
5.47715
197.987
```

Events with offset close to the trigger threshold. These are due to ambient switching noise in the North Area.

In[123]:=
```
Histogram[choff2, {.005, .02, .001}, PlotLabel → "Ch2"]
```

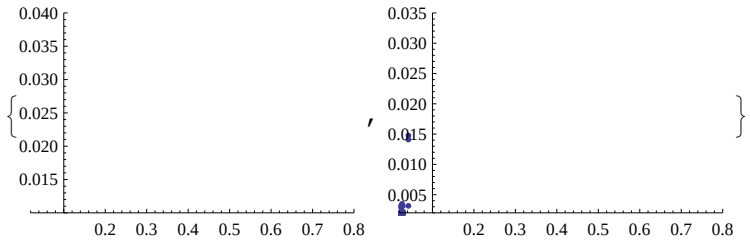
Out[123]=



Some Correlation Plots

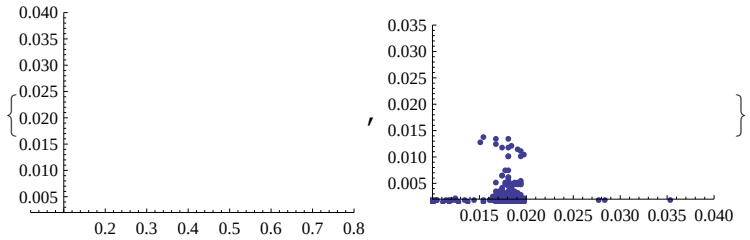
In[124]:=
```
{ListPlot[Transpose[{chmax1, chmax2}], PlotRange → {{0.02, 0.8}, {0.01, 0.04}}],
 ListPlot[Transpose[{chmax1, chmax3}], PlotRange → {{0.02, 0.8}, {0.002, 0.035}}]}
```

Out[124]=



In[125]:=
```
{ListPlot[Transpose[{chmax1, chmax4}], PlotRange → {{0.02, 0.8}, {0.002, 0.04}}],
 ListPlot[Transpose[{chmax2, chmax3}],
  PlotRange → {{0.01, 0.04}, {0.002, 0.035}}]}
```

Out[125]=



Set up a Status word Useful for selecting events for output.

```
statusw = ConstantArray[0, 1000]; goodevents = 0;
Do[
 If[chmax1[[iev]] < 0.3 && chmax1[[iev]] > 0.02,
   statusw[[iev]] = 0, statusw[[iev]] = 5];
 If[chmax3[[iev]] > 0.003, , statusw[[iev]] = statusw[[iev]] + 2];
 If[chmax2[[iev]] > 0.039 || chmax3[[iev]] > 0.039,
   statusw[[iev]] = statusw[[iev]] + 10];
 If[choff2[[iev]] < 0.006, , statusw[[iev]] = statusw[[iev]] + 1];
 If[statusw[[iev]] == 0, goodevents++]
 , {iev, 1000}]
```

Cuts used to select 1 Mip candidates to be written for output :
-Scint ph consistent with Mip
-a hit in the second 64 mm^2 APD
-upper limit on both 64 mm^2 APD pulse eights for consistency with Mip
- a cut on the baseline to remove obvious low frequency noise

```
Histogram[statusw, LabelingFunction -> Above]
```