Table of Contents

Testing the NI Board

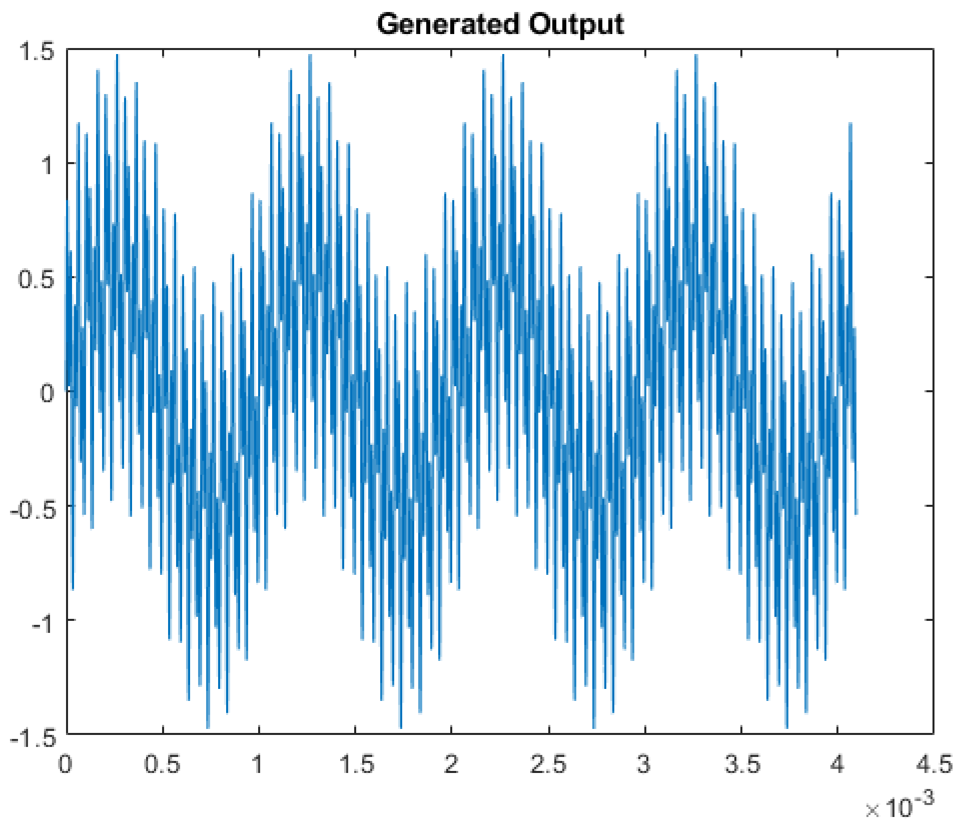
```
clearvars; close all; clc;
```

Vars

```
N = 1024 * 4;
```

Add Output

```
d = daq("ni");
addoutput(d,"Dev1","ao0","Voltage");
d.Rate = 1e+06;

amplitudePeakToPeak_ch1 = 1;

sineFrequency = 1e3; % 1e3 Hz
sineFrequency2 = 2e4;
sineFrequency3 = 5e4;
duration = 0.5;

outputSignal = [];
outputSignal(:,1) = createSine(amplitudePeakToPeak_ch1/2, sineFrequency,
sineFrequency2, sineFrequency3, d.Rate, "bipolar", duration);
t = (0:(N-1)) / d.Rate;
figure;
plot(t, outputSignal(1:N))
title('Generated Output')
source = outputSignal(1:N);

preload(d,outputSignal);
start(d,"repeatoutput")
```



Add Inputs

```
dd = daq("ni");
dd.Rate = 1000000;
%{
ch1 = addinput(dd,"Dev1","ai0","Voltage");
ch1.TerminalConfig = "SingleEnded";
ch1.Range = [-1 1];
%}
ch2 = addinput(dd,"Dev1","ai1","Voltage");
ch2.TerminalConfig = "SingleEnded";
ch2.Range = [-1 1];
%{
ch3 = addinput(dd,"Dev1","ai2","Voltage");
ch3.TerminalConfig = "SingleEnded";
ch3.Range = [-1 1];
ch4 = addinput(dd,"Dev1","ai3","Voltage");
ch4.TerminalConfig = "SingleEnded";
ch4.Range = [-1 1];
ch5 = addinput(dd,"Dev1","ai4","Voltage");
ch5.TerminalConfig = "SingleEnded";
ch5.Range = [-1 1];
ch6 = addinput(dd,"Dev1","ai5","Voltage");
ch6.TerminalConfig = "SingleEnded";
ch6.Range = [-1 1];
ch7 = addinput(dd,"Dev1","ai6","Voltage");
ch7.TerminalConfig = "SingleEnded";
ch7.Range = [-1 1];
ch8 = addinput(dd,"Dev1","ai7","Voltage");
ch8.TerminalConfig = "SingleEnded";
ch8.Range = [-1 1];
%}
ch9 = addinput(dd,"Dev2","ai0","Voltage");
ch9.TerminalConfig = "SingleEnded";
ch9.Range = [-1 1];
%{
ch10 = addinput(dd,"Dev2","ai1","Voltage");
ch10.TerminalConfig = "SingleEnded";
ch10.Range = [-1 1];
ch11 = addinput(dd,"Dev2","ai2","Voltage");
ch11.TerminalConfig = "SingleEnded";
ch11.Range = [-1 1];
ch12 = addinput(dd,"Dev2","ai3","Voltage");
ch12.TerminalConfig = "SingleEnded";
ch12.Range = [-1 1];
ch13 = addinput(dd,"Dev2","ai4","Voltage");
ch13.TerminalConfig = "SingleEnded";
ch13.Range = [-1 1];
ch14 = addinput(dd,"Dev2","ai5","Voltage");
ch14.TerminalConfig = "SingleEnded";
ch14.Range = [-1 1];
ch15 = addinput(dd,"Dev2","ai6","Voltage");
ch15.TerminalConfig = "SingleEnded";
ch15.Range = [-1 1];
ch16 = addinput(dd,"Dev2","ai7 ","Voltage");
ch16.TerminalConfig = "SingleEnded";
ch16.Range = [-1 1];
%}
```

Add Triggers

Use `addtrigger` to add a digital start trigger from (`'RTSI0/PFI3'` (source) to `'RTSI0/Dev4'` (destination)

```
addtrigger(dd,"Digital","StartTrigger","Dev1/RTSI0","Dev2/RTSI0");
```

Add Scan Clock

Use `addclock` to share a scan clock using the `RTSI1` terminal connection.

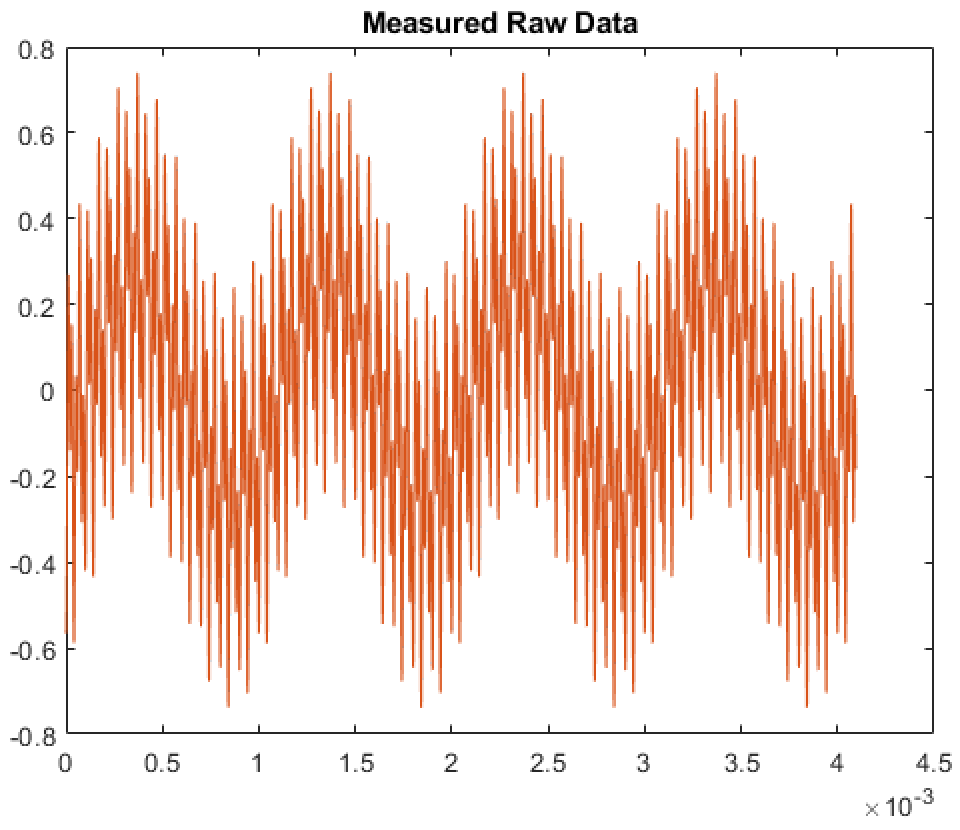
```
addclock(dd, "ScanClock", "Dev1/RTSI1","Dev2/RTSI1");
```

Acquire Data with Synchronization

Use `read` to acquire data.

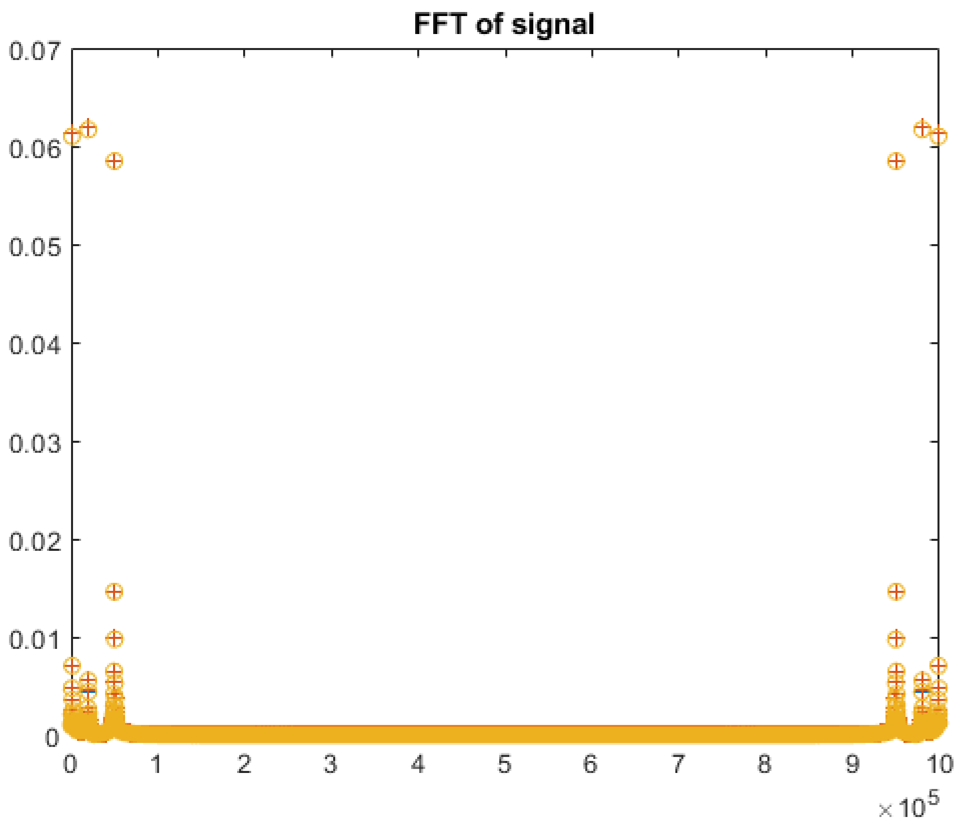
```
[signal,time] = read(dd, N, "OutputFormat", "Matrix");
figure();
```

```
plot(time,signal)
title('Measured Raw Data')

stop(d)
```



Verify with FFT

```
FFT_signal = fft(signal, N);
FFT_source = fft(source, N);
f_HZ = dd.Rate * (0:N-1)/N;
figure;
plot(f_HZ, abs(FFT_signal)/N/2, '+', f_HZ, abs(FFT_source)/N/4, 'o')
title('FFT of signal')
```



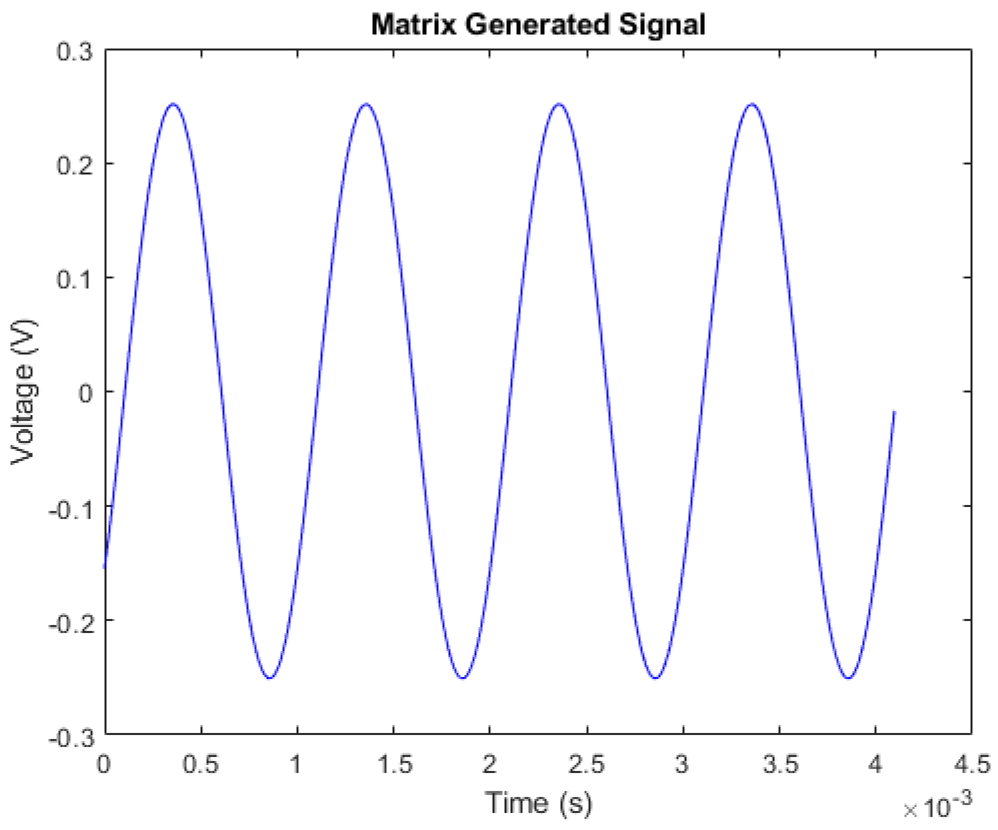
Sampling

declare more varibles

```
k = 0:N-1;              % vector of n points
f_samp = dd.Rate;       % Hz
tk = k/f_samp;          % xAxis

freq = 1000;          % desired frequency in Hz
w0 = 2*pi*freq;         % omega???

% matrix containing the desired frequency component of the system
E = [sin(w0*tk)', cos(w0*tk)', ones(1, length(tk))'];

% inverse of E * s^t (pinv)
phi =  E\signal; % gives alpha, beta and offset

% seperate phi characteristics
alpha = phi(1);
beta = phi(2);
Cout = phi(3);

% create the signal @wo
f1Signal = E * phi;
```

plot new signal

```
figure
plot(time, f1Signal, 'color', 'b')
title("Matrix Generated Signal")
xlabel("Time (s)");
ylabel("Voltage (V)");
```



Find/verify PHI componets

offset

```
offset = mean(f1Signal);
% amplitude in volt peak
ampVp = sqrt(alpha^2 + beta^2);
% phase
phase = asind(beta/ampVp);
```

Create Sine Function

```
function sine = createSine(A, f1, f2, f3, sampleRate, type, duration)

numSamplesPerCycle = floor(sampleRate/f1);
T = 1/f1;
```

```
timestep = T/numSamplesPerCycle;
t = (0 : timestep : T-timestep)';

if type == "bipolar"
    y = A*sin(2*pi*f1*t) + A*sin(2*pi*f2*t)+ A*sin(2*pi*f3*t);
elseif type == "unipolar"
    y = A*sin(2*pi*f1*t) + A*sin(2*pi*f2*t) + A*sin(2*pi*f3*t) + A;
end

numCycles = round(f1*duration);
sine = repmat(y,numCycles,1);
end
```

Published with MATLAB® R2023b