Analog Input Examples

On this page…

Basic Steps for Acquiring Data

This section illustrates how to perform basic data acquisition tasks using analog input subsystems and Data Acquisition Toolbox™ software. For most data acquisition applications, you must follow these basic steps:

1. Install and connect the components of your data acquisition hardware. At a minimum, this involves connecting a sensor to a plug-in or external data acquisition device.
2. Configure your data acquisition session. This involves creating a device object, adding channels, setting property values, and using specific functions to acquire data.
3. Analyze the acquired data using MATLAB®.

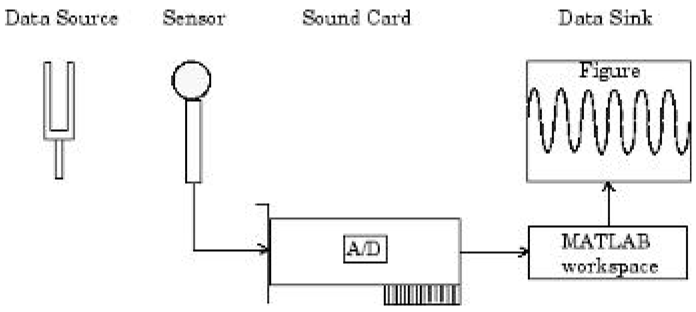
Simple data acquisition applications using a sound card and a National Instruments® board are given below.

To see how to set up continuous analog input acquisitions, refer to the Continuous Acquisitions Using Analog Input example.

Acquire Data with a Sound Card

Note: You cannot use the legacy interface on 64–bit MATLAB. See Session-Based Interface to acquire and generate data.

Suppose you must verify that the fundamental (lowest) frequency of a tuning fork is 440 Hz. To perform this task, you will use a microphone and a sound card to collect sound level data. You will then perform a fast Fourier transform (FFT) on the acquired data to find the frequency components of the tuning fork. The setup for this task is shown below.



Configure Data Acquisition Session

For this example, you will acquire 1 second of sound level data on one sound card channel. Because the tuning fork vibrates at a nominal frequency of 440 Hz, you can configure the sound card to its lowest sampling rate of 8000 Hz. Even at this lowest rate, you should not experience any aliasing effects because the tuning fork will not have significant spectral content above 4000 Hz, which is the Nyquist frequency. After you set the tuning fork vibrating and place it near the microphone, you will trigger the acquisition one time using a manual trigger.

You can run this example by typing `daqdoc4_1` at the MATLAB Command Window.

1. **Create a device object** — Create the analog input object `AI` for a sound card. The installed adaptors and hardware IDs are found with `daqhwinfo`.

   ```
   AI = analoginput('winsound');
   ```

2. **Add channels** — Add one channel to `AI`.

   ```
   chan = addchannel(AI,1);
   ```

3. **Configure property values** — Assign values to the basic setup properties, and create the variables `blocksize` and `Fs`, which are used for subsequent analysis. The actual sampling rate is retrieved because it might be set by the engine to a value that differs

from the specified value.

```
duration = 1; %1 second acquisition
set(AI,'SampleRate',8000)
ActualRate = get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',duration*ActualRate)
set(AI,'TriggerType','Manual')
blocksize = get(AI,'SamplesPerTrigger');
Fs = ActualRate;
```

See "The Sampling Rate" for more information.

4. **Acquire data** — Start `AI`, issue a manual trigger, and extract all data from the engine. Before `trigger` is issued, you should begin inputting data from the tuning fork to the sound card.

   ```
   start(AI)
   trigger(AI)
   wait(AI,duration + 1)
   ```

   The `wait` function pauses MATLAB until either the acquisition completes or the time-out elapses (whichever comes first). If the time-out elapses, an error occurs. Adding 1 second to the duration allows some margin for the time-out.

   ```
   data = getdata(AI);
   ```

5. **Clean up** — When you no longer need `AI`, you should remove it from memory and from the MATLAB workspace.

   ```
   delete(AI)
   clear AI
   ```

Analyze Data

Note: You cannot use the legacy interface on 64–bit MATLAB. See Session-Based Interface to acquire and generate data.

For this example, analysis consists of finding the frequency components of the tuning fork and plotting the results. To do so, the function `daqdocfft` was created. This function calculates the FFT of `data`, and requires the values of `SampleRate` and `SamplesPerTrigger` as well as `data` as inputs.

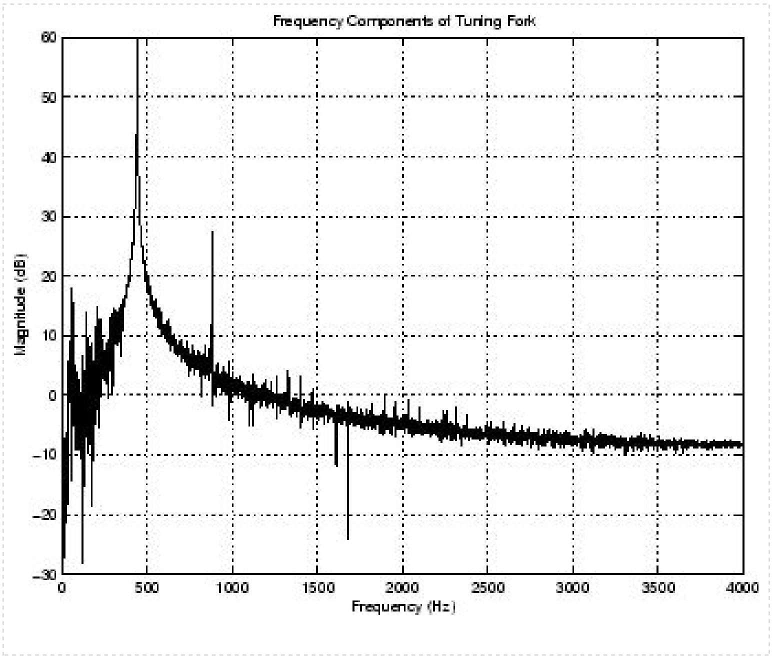
```
[f,mag] = daqdocfft(data,Fs,blocksize);
```

`daqdocfft` outputs the frequency and magnitude of `data`, which you can then plot. `daqdocfft` is shown below.

```
function [f,mag] = daqdocfft(data,Fs,blocksize)
%    [F,MAG]=DAQDOCFFT(X,FS,BLOCKSIZE) calculates the FFT of X
%    using sampling frequency FS and the SamplesPerTrigger
%    provided in BLOCKSIZE

xfft = abs(fft(data));

% Avoid taking the log of 0.
index = find(xfft == 0);
xfft(index) = 1e-17;

mag = 20*log10(xfft);
mag = mag(1:floor(blocksize/2));
f = (0:length(mag)-1)*Fs/blocksize;
f = f(:);
```

The results are given below.

```
plot(f,mag)
grid on
ylabel('Magnitude (dB)')
xlabel('Frequency (Hz)')
title('Frequency Components of Tuning Fork')
```



The plot shows the fundamental frequency around 440 Hz and the first overtone around 880 Hz. A simple way to find actual fundamental frequency is

```
[ymax,maxindex]= max(mag);
maxfreq = f(maxindex)
maxfreq =
   441
```

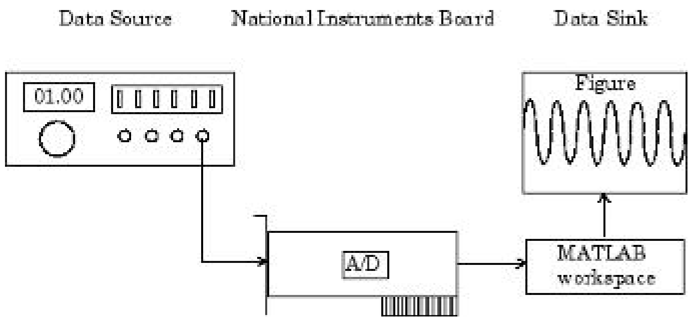
The answer is 441 Hz.

Note The fundamental frequency is not always the frequency component with the largest amplitude. A more sophisticated approach involves fitting the observed frequencies to a harmonic series to find the fundamental frequency.

Acquire Data with a National Instruments Board

Note: You cannot use the legacy interface on 64–bit MATLAB. See Session-Based Interface to acquire and generate data.

Suppose you must verify that the nominal frequency of a sine wave generated by a function generator is 1.00 kHz. To perform this task, you will input the function generator signal into a National Instruments board. You will then perform a fast Fourier transform (FFT) on the acquired data to find the nominal frequency of the generated sine wave. The setup for this task is shown below.



Configure Data Acquisition

For this example, you will acquire 1 second of data on one input channel. The board is set to a sampling rate of 10 kHz, which is well above the frequency of interest. After you connect the input signal to the board, you will trigger the acquisition one time using a manual trigger.

> **Note:** You cannot use the legacy interface on 64–bit MATLAB. See Session-Based Interface to acquire and generate data.

You can run this example by typing `daqdoc4_2` at the MATLAB Command Window.

1. **Create a device object** — Create the analog input object `AI` for a National Instruments board. The installed adaptors and hardware IDs are found with `daqhwinfo`.

   ```
   AI = analoginput('nidaq','Dev1');
   ```

2. **Add channels** — Add one channel to `AI`.

   ```
   chan = addchannel(AI,0);
   ```

3. **Configure property values** — Assign values to the basic setup properties, and create the variables `blocksize` and `Fs`, which are used for subsequent analysis. The actual sampling rate is retrieved because it might be set by the engine to a value that differs from the specified value.

   ```
   duration = 1; %1 second acquisition
   set(AI,'SampleRate',10000)
   ActualRate = get(AI,'SampleRate');
   set(AI,'SamplesPerTrigger',duration*ActualRate)
   set(AI,'TriggerType','Manual')
   blocksize = get(AI,'SamplesPerTrigger');
   Fs = ActualRate;
   ```

   See "The Sampling Rate" for more information.

4. **Acquire data** — Start `AI`, issue a manual trigger, and extract all data from the engine. Before `trigger` is issued, you should begin inputting data from the function generator into the data acquisition board.

   ```
   start(AI)
   trigger(AI)
   wait(AI,duration + 1)
   ```

   The `wait` function pauses MATLAB until either the acquisition completes or the time-out elapses (whichever comes first). If the time-out elapses, an error occurs. Adding 1 second to the duration allows some margin for the time-out.

   ```
   data = getdata(AI);
   ```

5. **Clean up** — When you no longer need `AI`, you should remove it from memory and from the MATLAB workspace.

   ```
   delete(AI)
   clear AI
   ```

Analyze Data

> **Note:** You cannot use the legacy interface on 64–bit MATLAB. See Session-Based Interface to acquire and generate data.

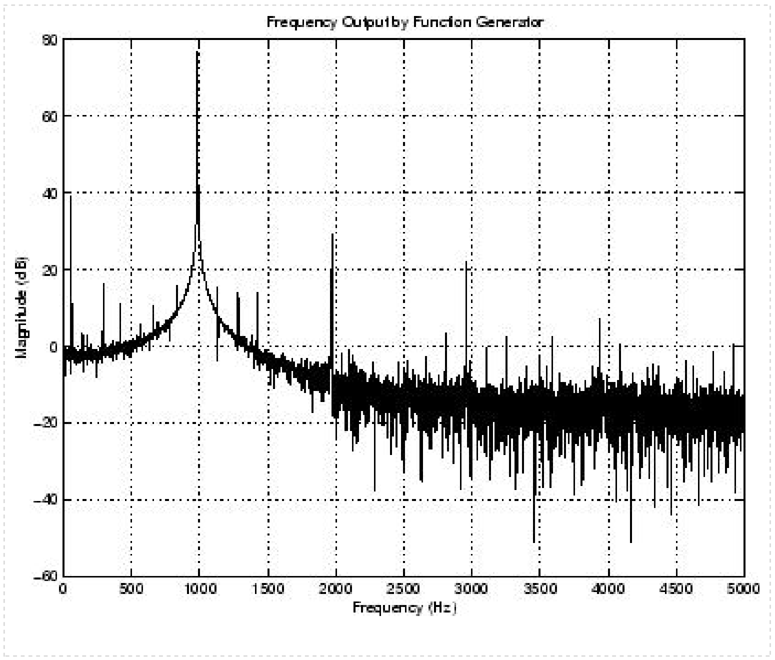
For this experiment, analysis consists of finding the frequency of the input signal and plotting the results. You can find the signal frequency with `daqdocfft`.

```
[f,mag] = daqdocfft(data,Fs,blocksize);
```

This function, which is shown in Analyze Data, calculates the FFT of `data`, and requires the values of `SampleRate` and `SamplesPerTrigger` as well as `data` as inputs. `daqdocfft` outputs the frequency and magnitude of `data`, which you can then plot.

The results are given below.

```
plot(f,mag)
grid on
ylabel('Magnitude (dB)')
xlabel('Frequency (Hz)')
title('Frequency Output by Function Generator')
```



This plot shows the nominal frequency around 1000 Hz. A simple way to find actual frequency is shown below.

```
[ymax,maxindex]= max(mag);
maxindex
maxindex =
   994
```

The answer is 994 Hz.