Overflows

The AD9363 transceiver in the PlutoSDR is capable of operating at 61.4 MHz of complex bandwidth, which is suitable for many modern radio standards. However, downstream bottlenecks will usually limit data transfer speed. For example, USB2.0 will limit data transfer to 25 MB/s, or~6.5MS/s of data rate (6.5 MHz of complex bandwidth). A detailed outline of the data path from the transceiver to MATLAB is outlined in Figure 4. You should keep this in mind when streaming data from the transceiver as there will be latency introduced in the USB transmission due to the added overhead. Once data gets to the PC it must be processed, and even high-end PCs, performing calculations (especially double precision ones) will be difficult at this speed.

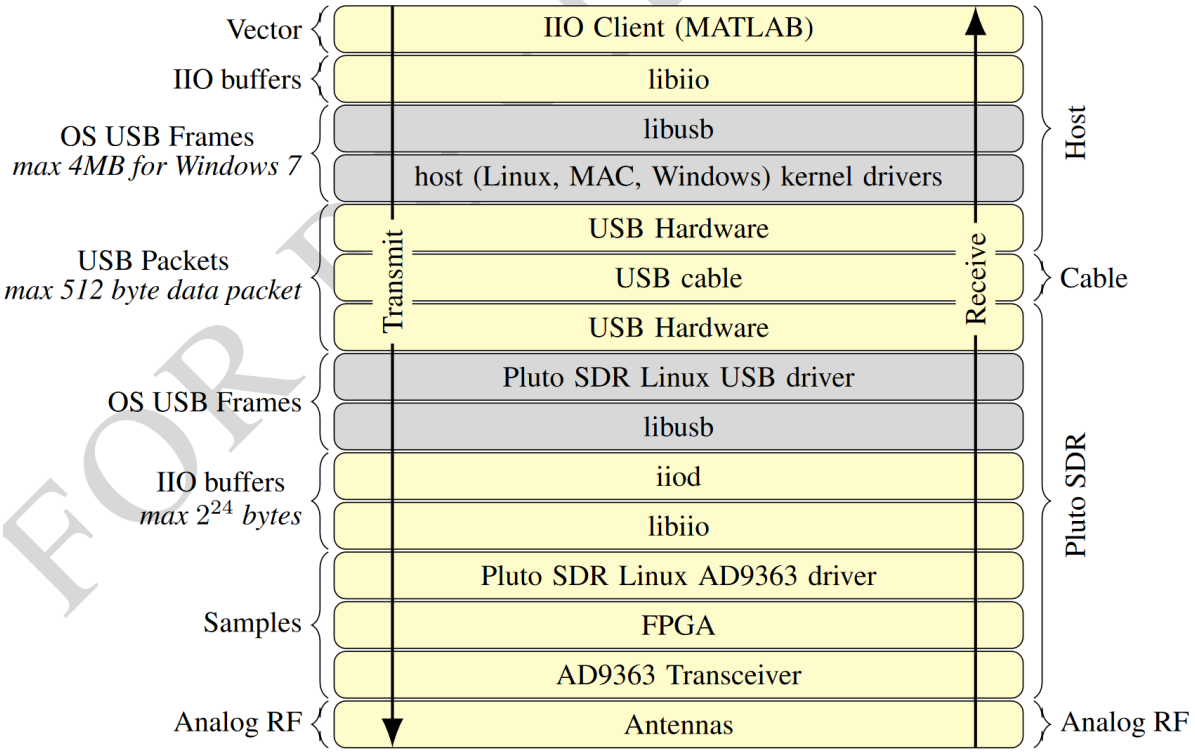


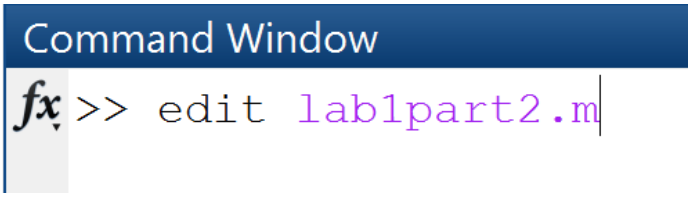
Figure 4 Data path from AD9363 to MATLAB

For applications that are relatively low in bandwidth, or those that require less complex processing, it may be possible to operate in "real-time" in MATLAB and Simulink, and the Host PC can process data faster than it can be produced by the device. Our goal here, however, is to develop a deployable system, so optimizing MATLAB code for simulation speed is not a priority. Therefore, when performing data "streaming", we are actually operating on bursts of data, and not a continuous stream, and need to be mindful of data overflows and ensure that sufficient darta is captured in each burst to provide a meaningful assessment of our algorithm.

Data overflow is a condition where data is lost because the step method of the radio object is not called fast enough. Checking the output of the step-call to the radio will tell us if this condition occurred. To reduce this possibility, we utilize coding styles, or templates, that avoid upfront processing, capture a

large chunk of contiguous data, and then perform processing on that data. This ensures that enough contiguous data is captured by the host PC to test the algorithms.

We will start by opening the script lab1part2.m from the command line:



This script has two main parameters to configure on lines 12 and 13:

```
%% CHANGE ME HERE

% Overflow only in SA
SampleRate = 4.5e6;
SamplesPerRXFrame = 2^16;
```

- **SampleRate** is the rate the device is configured to collect data
- **SamplesPerRXFrame** is the size of the buffer transfer from PlutoSDR to MATLAB.

The remaining part of the script will check for overflow conditions while data is processed in the capture loop, which drivesa Spectrum Analyzer scope visualization. We also check for overflow conditions when this visualization is done after all the data is collected.

Run the script lab1part2 and view the information displayed on the console:

```
>> lab1part2
(OUTLOOP) Overflow events: 0 of 100
(INLOOP) Overflow events: 59 of 100
```

The displayed numbers relate to the number of overflows that occurred. This is a great example of why we might not want to do processing within the capture loop. We can try to reduce these overflow events by either increasing the SamplesPerRXFrame value, which reduces the overhead per sample of data pull from PlutoSDR, or we can reduce the SampleRate if our application allows.

Edit line 13 to reduce the SamplesPerRXFrame to 2^15 as follows:

```
11      % Overflow only in SA
12 -    SampleRate = 4.5e6;
13 -    SamplesPerRXFrame = 2^15;
14 -    FramesToCollect = 1e2;
```

Now rerun the script and observe overflow conditions that occur in both cases:

```
>> lab1part2
(OUTLOOP) Overflow events: 78 of 100
(INLOOP) Overflow events: 99 of 100
fx >>
```

Now try reducing the sample rate. ***Edit line 12 to reduce the SampleRate as follows:***

```
11    % Overflow only in SA
12 -  SampleRate = 1e6;
13 -  SamplesPerRXFrame = 2^15;
14 -  FramesToCollect = 1e2;
```

Rerun the script and observe overflow conditions that occur in both cases:

```
>> lab1part2
(OUTLOOP) Overflow events: 0 of 100
(INLOOP) Overflow events: 0 of 100
```

Question: The maximum SamplePerFrame possible is 2^20. At this size, what is the maximum rate you can collect data without overflow?