COMPSYS301

Hardware Task 1 – Measurment of Analogue Signals[1]

Task 1 – ADC /DAC

This task will require you to –

- Simulate a circuit on LTSPICE
- Build the circuit on a breadboard
- Create a PSoC schematic for a reading and producing analogue signals
- Write C code to digitize an analogue signal.
- Verify your solution by visualizing the on Matlab and a oscilloscope

The Schematic

Simulate the circuit shown below in LTSPICE before you come to the lab.

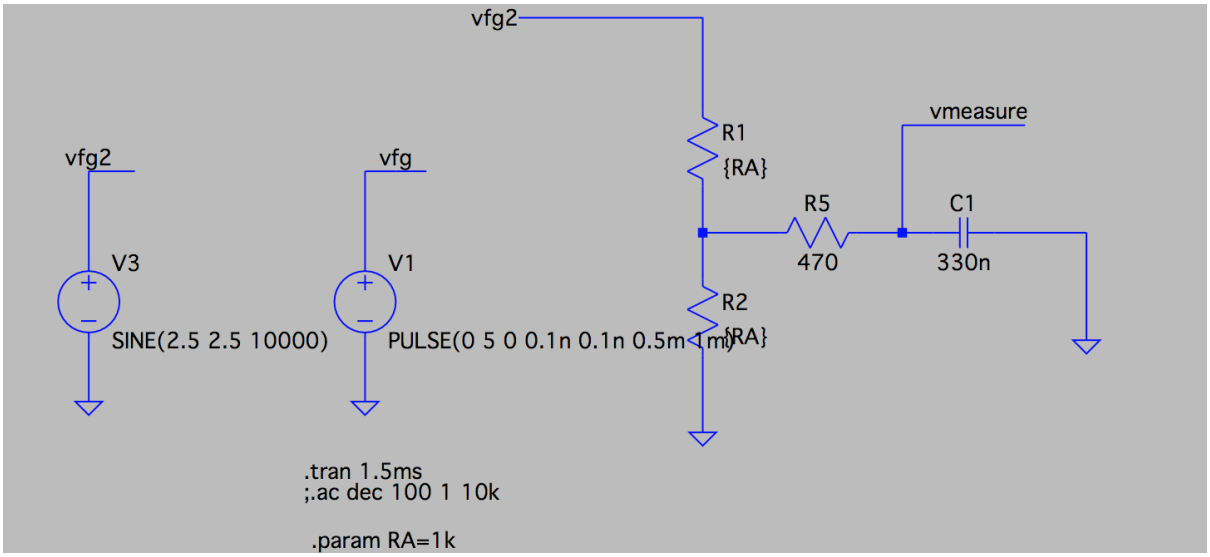

Figure 1 - LTSPICE Schematic

The schematic in Figure 1 shows a RC circuit with component values of R1 and R2 parametrised using the .PARAM spice directive and the matching '{value}' in the component's value field. The schematic also shows to voltage generators V1 and V3.

R1 is shown to be connected to source V3 using the net label 'vfg2' and can be edited to excite the RC circuit with either of the signals. Note the simulation directive .TRAN is active while the .AC directive has been 'commented out'.

The figures below in Figure 2 and Figure 3 shows the results of simulation.

[1] A partial solution may be provided.

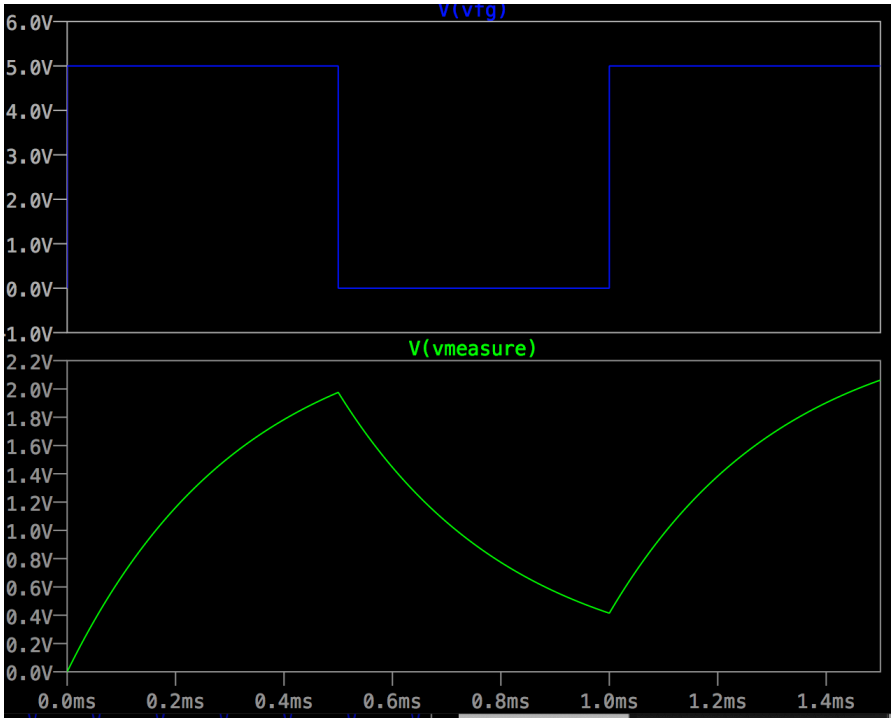

Figure 2 - Response to a square wave

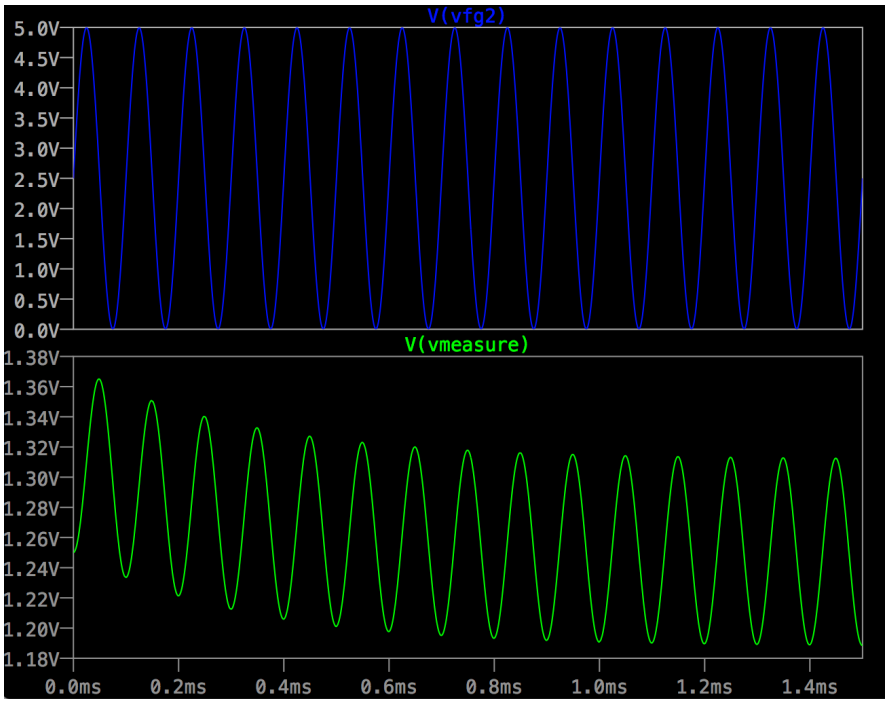

Figure 3 - Response to a Sinewave

Breadboard

Execute the following tasks. The PSoC on the Kit059 will be damaged if you connect signals beyond 5V. Hence you will verify the signal levels on a breadboard first.

1. Mount the PSoC on the breadboard. Do not connect it to anything
2. Build the RC circuit (R1, R2, R5 and C1)
3. Setup the function generator to produce 5Vpp sinewave with an offset of 2.5V[2]. Verify on the scope and using the benchtop DMM
4. Setup the function generator to produce 5Vpp square with an offset of 2.5V. Verify on the scope and using the benchtop DMM

ADC and DAC

Build up a PSoC schematic that includes a SAR ADC and a DAC. You will digitize the output of the RC circuit – at the junction of R5 and C1. You should convert the ADC value back into an analogue signal. Since the frequency of the sinewave is 10KHz, your ADC should convert the analogue signals (sampling rate) at a rate faster than the Nyquist limit (20KHz i.e. 20K samples per sec). You will need to match the resolution of the ADC output to the DAC input.

- You will need a SAR ADC and an input pin, a DAC and an output pin, an interrupt, a timer and a USBUART.
- Configure the ADC for a suitable sampling rate. Note: You may use a free-running configuration but you need to fix the sampling rate if you intend to visualize the results in Matlab. Configure the pins suitablly for analogue input and output.
- You will need one interrupt triggered on an ADC's EOC event (end of conversion).
- One solution is to use a Timer that asserts its TC (terminal count) signal at the desired sampling rate. The TC pin of the TIMER could be connnected to the ADS's SOC input (start of conversion). The EOC will be triggered after a fixed number of clocks and hence you will have ADC values at the designed sampling rate.
- In C, the ADC values will/may need to be resized before applying it to the DAC.
- Depending on the sampling rate, the transfer of data via the USBUART may experience bottle-necks. Perhaps you can store a ADC results into a suitably sized buffer. To do this you will need to define the size of this buffer, and in C arrange to digitize and store a fixed number of samples and then send this buffer via the USBUART.

Verification

To ensure the ADC is operating as expected, it is best to verify the behaviour. The use of a DAC permits a very quick verification. Since the DAC is limited in resolution, you will also verify the measurements in Matlab.

- Probe the junction of the R5/C1 and the DAC output using scope.
- Switch between a sine and square wave.

---

[2] Ensure that the output of the function generator is setup to drive a high impedance load and not 50 ohms. If its configured for 50ohms, the generator will scale the output to 2x and hence a 5Vpp entry will effectively be 10Vpp.

- You may need tweak the sampling rate to improve the smoothness of the DAC output.
- Read the output of the USBUART into Matlab. Matlab can import text files[3] easily if formated correctly. The simplest is a CSV (comma separated values) – `index, ADCValue, DACValue`. The use of an index is important if your Matlab plot is to display actual time. It also can use used to detect values lost in transfer.
- An inaccurate clock on the PSoC will imply a measurement of the frequency in Matlab will also be inaccurate. Perhaps you can probe the 'TC' value to accurately measure the sampling rate. This is not required but if you have some time available you may wish to explore this too.

Questions

1. Why was the SAR selected? Does a scope use a SAR?
2. What is a single-ended and a differential ADC measurment?
3. Why is sampling rate important?
4. The lower limit of the sampling rate is crisply defined. Is the upper limit just as crisp?

Conclusions

On completion, you should have learnt or refreshed your knowledge on the following -

- Simulation using LTSPICE
- Building a circuit on a breadboard
- Digitised an analogue signal at a required sampling rate
- Become aware of various kinds ADC on the PSoC and their operating modes
- … and Matlab is pretty cool

[3] >> `[i, adcvalue, dacvalue] = textread('filename','%d, %f, %f')`
This matlab command will read the text data, assuming it formated as "index, value, value".
You can aslo skip some lines incase you are sending a line with column names.
>> `[i, adcvalue, dacvalue] = textread('filename','%d, %f, %f', 'headerlines', 1)`
Here the first line is interpreted as a 'header' and will not be parsed.