Feedback Control

Introduction to digital Low-Level Radio Frequency Controls in Accelerators

Lab 11

Qiang Du

US Particle Accelerator School
Janurary 23 – 27, 2023

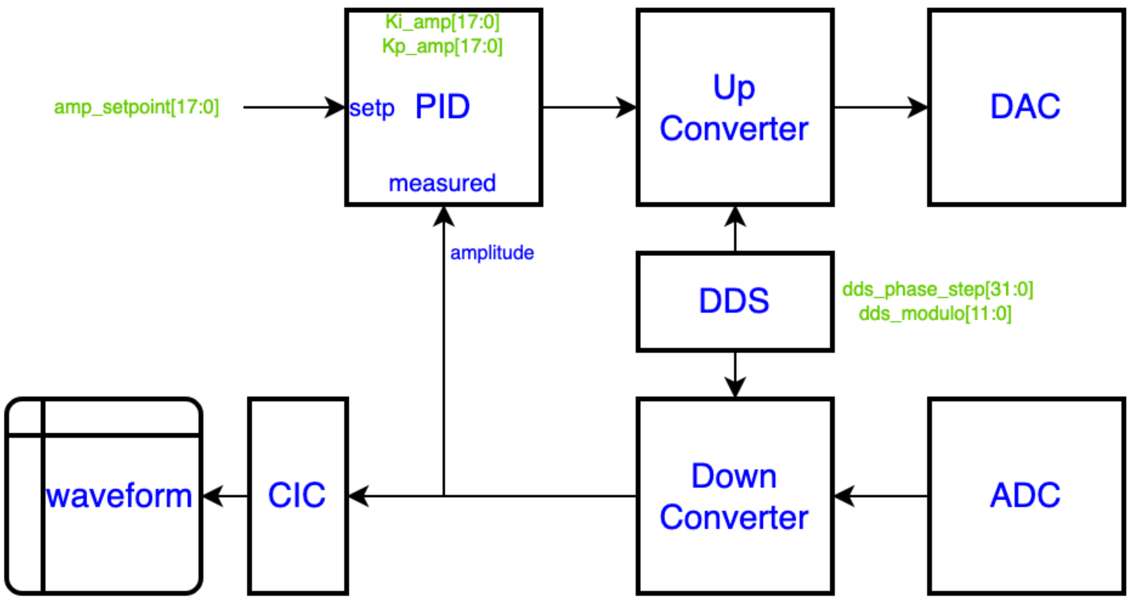
Contents

1 Introduction

In this lab we will learn how to use a digital LLRF system as a vector network analyzer. We will also leverage the experience from previous lab exercise on ADC, DAC and DDS, and the 20 MHz crystal characterizations.

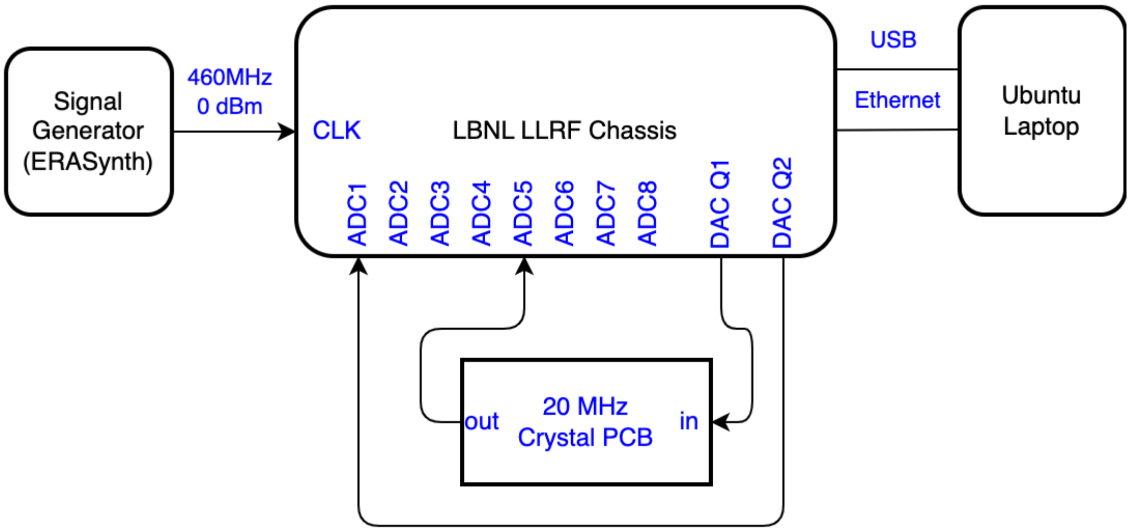
We will operate the feedback loops using the 20 MHz crystal as a resonator.

1.1 Firmware details



Frequency	Derivation	Value	Unit
f_MO		480	MHz
f_IF	f_MO/24	20	MHz
f_LO	f_MO - f_IF	460	MHz
f_CLK	f_LO / 4	115	MHz
f_IF / f_CLK		4 / 23	

1.2 Hardware setup



Connect the system as shown in the diagram, so we have both direct loop back on `ADC1` and with 20 MHz crystal in the system on `ADC5`. The firmware has identical `DAC Q1` and `DAC Q2` output, as we previously measured on the DAC characterization lab.

1.3 Firmware and software setup

- Configure FPGA chassis using the provided `marble_zest_top_uspas.bit`;
- Start EPICS IOC on the connected laptop computer;
- Run Phoebus GUI on the connected laptop computer;

2 Exercises

2.1 Run EPICS Phoebus interface

See `README.md` for docker command:

```
xhost local:root
docker run --rm -it --name phoebus_alsu --network="host" -e DISPLAY=${DISPLAY} -v /tmp/.
```

2.2 Run pulse mode

- Set Amp Loop setpoint to 30000.
- Toggle "Pulse Mode" button to "ON".
- Adjust pulse width to 60000.
- Set waveform `Decimation` to 50.

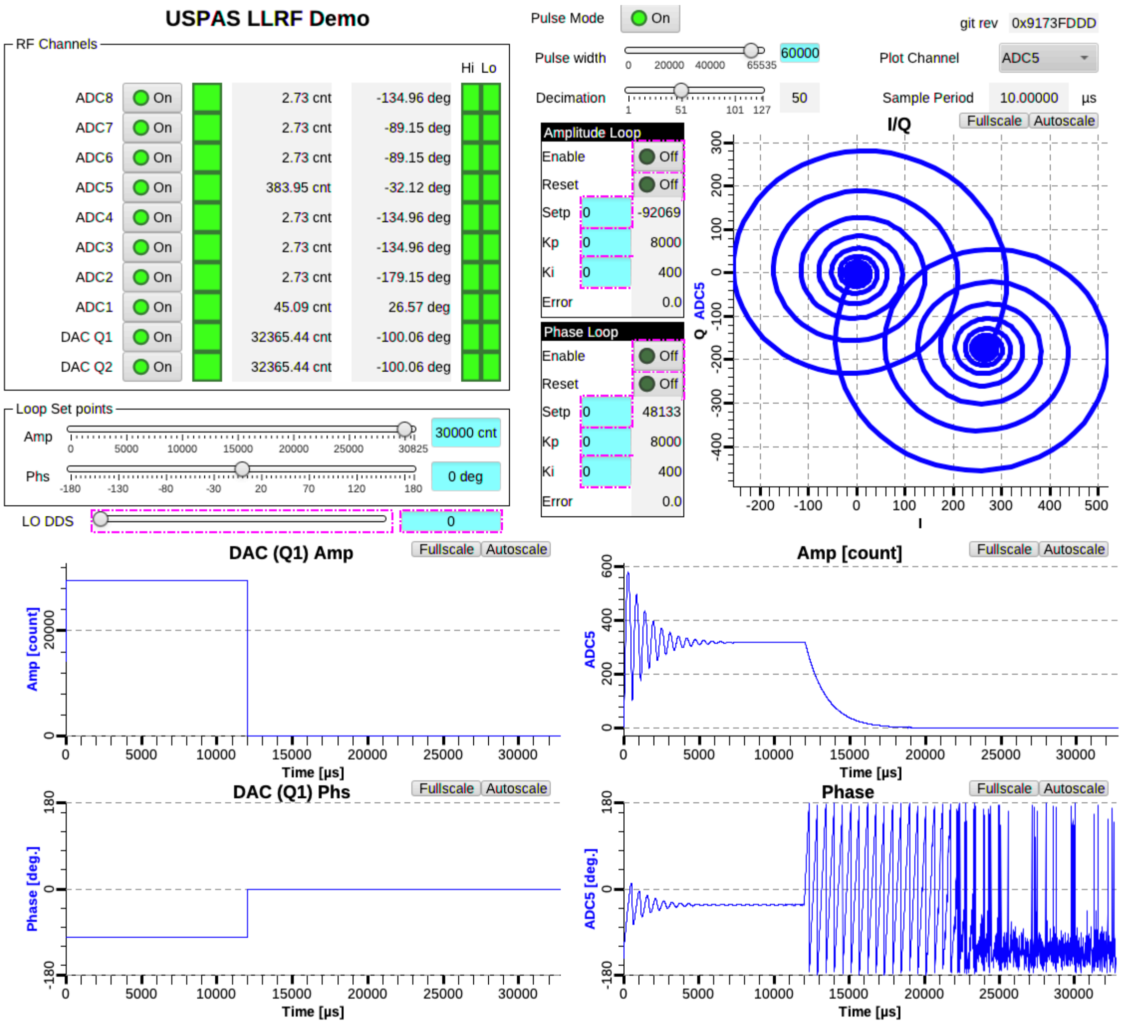
The pulse length is 23 * 60000 / 115e6 = 12.0 µs. The CIC filter before the waveform will average 23 * Decimation samples, so the waveform sample interval is 1/115e6 * 50 * 23 = 10.0 µs. This will give us enough length of waveforms to capture about 32 ms.

[7]:
```python
%matplotlib inline
from matplotlib import pyplot as plt
from scipy import signal
import pandas as pd
import numpy as np
from epics import PV, caget, caput

import os
os.environ['EPICS_CA_ADDR_LIST'] = 'localhost'
os.environ['EPICS_CA_AUTO_ADDR_LIST'] = 'NO'

plt.rcParams['figure.figsize'] = [6, 4]
plt.rcParams['axes.grid'] = True
plt.rcParams['axes.grid.which'] = "both"
plt.rcParams['grid.linewidth'] = 0.5
plt.rcParams['grid.alpha'] = 0.5
plt.rcParams['font.size'] = 8
```

[8]:
```python
caput('USPAS:LLRF:Loop:AmpSetp', 30000)
caput('USPAS:LLRF:reg_pulse_mode', 1)
caput('USPAS:LLRF:reg_pulse_high_len', 60000)
caput('USPAS:LLRF:ACQ_DECIM', 50)
```

[8]: 1

The ADC5 waveform will show the step response of the crystal. It is expected to have a result like the following screenshot.



Explain the reason of ringing of the ADC5 amplitude and phase waveforms. Explain the spirals in IQ waveform plot.

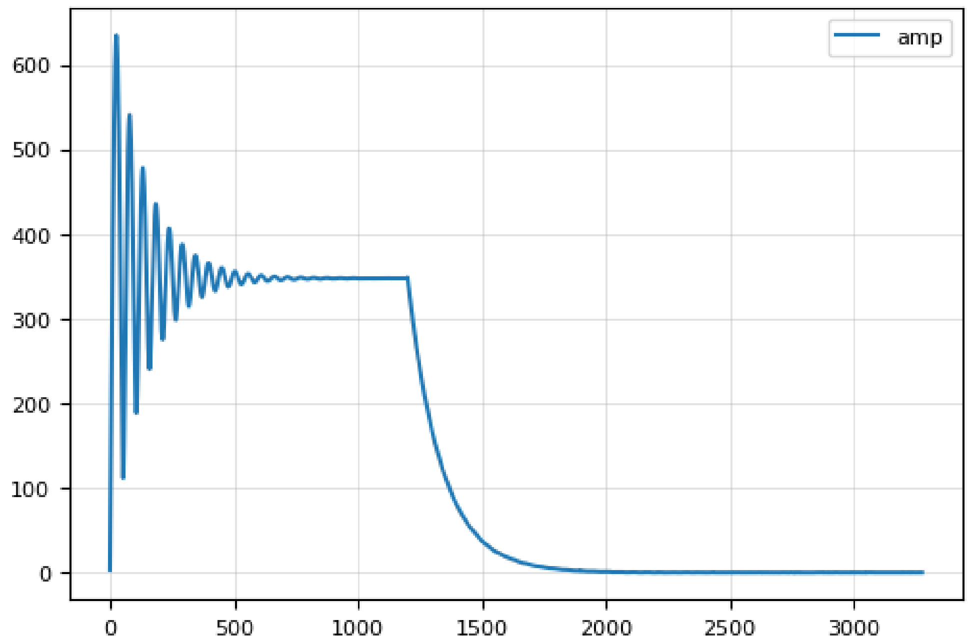
2.3 Find cavity detune through falling edge analysis

[9]:
```
fs = 1e6 / PV('USPAS:LLRF:ACQ_SAMP_PERIOD').value
pv_twf = PV('USPAS:LLRF:CavCel:TWF')
pv_awf = PV('USPAS:LLRF:CavCel:AWF')
pv_pwf = PV('USPAS:LLRF:CavCel:PWF')
pv_iwf = PV('USPAS:LLRF:CavCel:IWF')
pv_qwf = PV('USPAS:LLRF:CavCel:QWF')

df = pd.DataFrame({
    'T [µs]': pv_twf.value,
    'amp': pv_awf.value,
    'phs': pv_pwf.value,
```

```
    'i': pv_iwf.value, 'q': pv_qwf.value})
df.set_index('T [µs]');
```

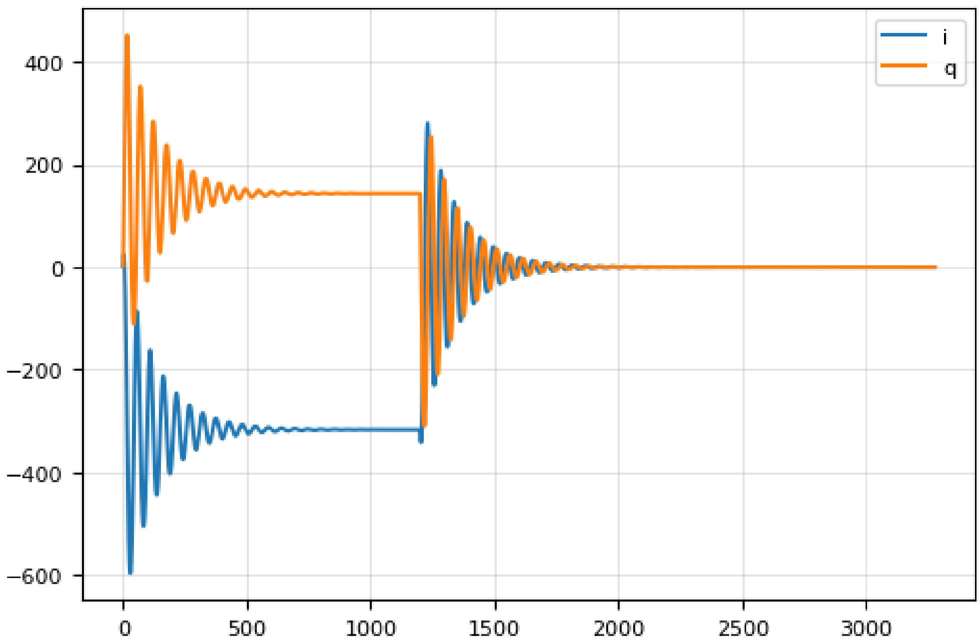
[10]:
```
df[['amp']].plot()
```

[10]: <AxesSubplot: >



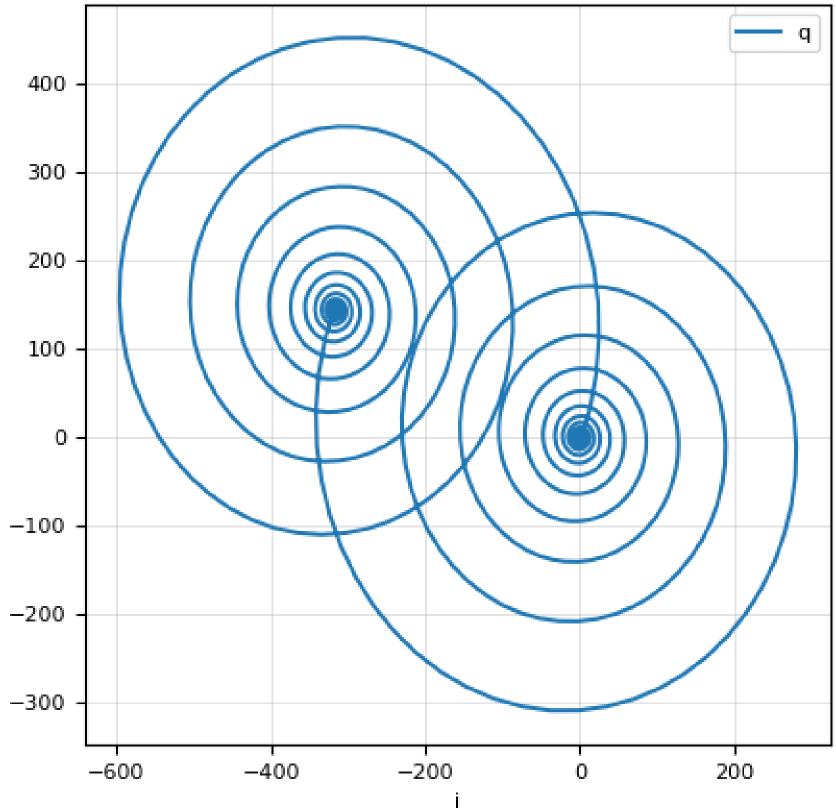
[11]:
```
df[['i', 'q']].plot()
```

[11]: <AxesSubplot: >



[12]:
```
df[['i', 'q']].plot(x='i', figsize=(5,5))
```

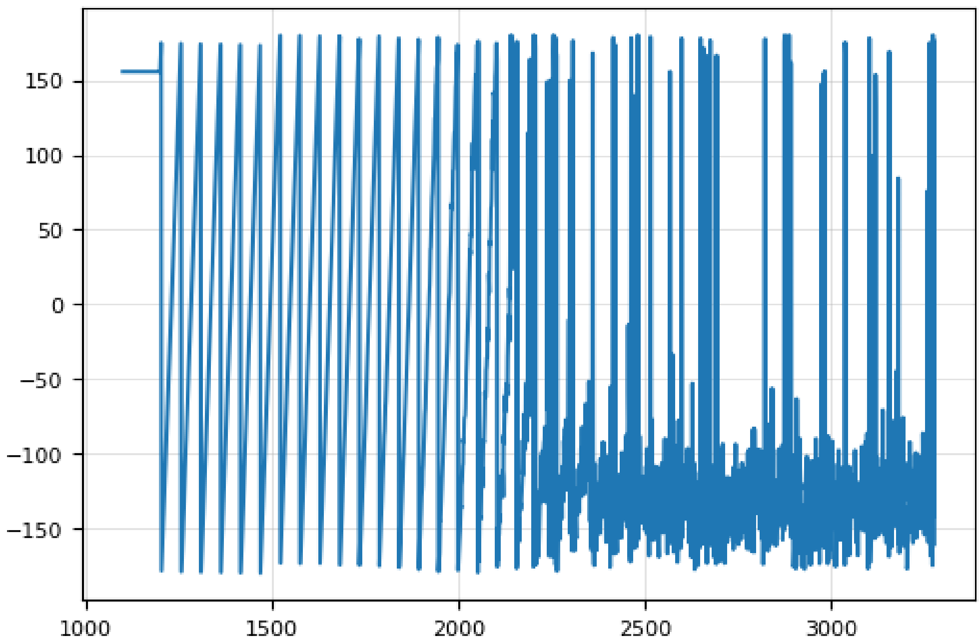
[12]: <AxesSubplot: xlabel='i'>



2.4 Extract detune frequency from falling edge

Confirm falling edge starts from 1200 µs:

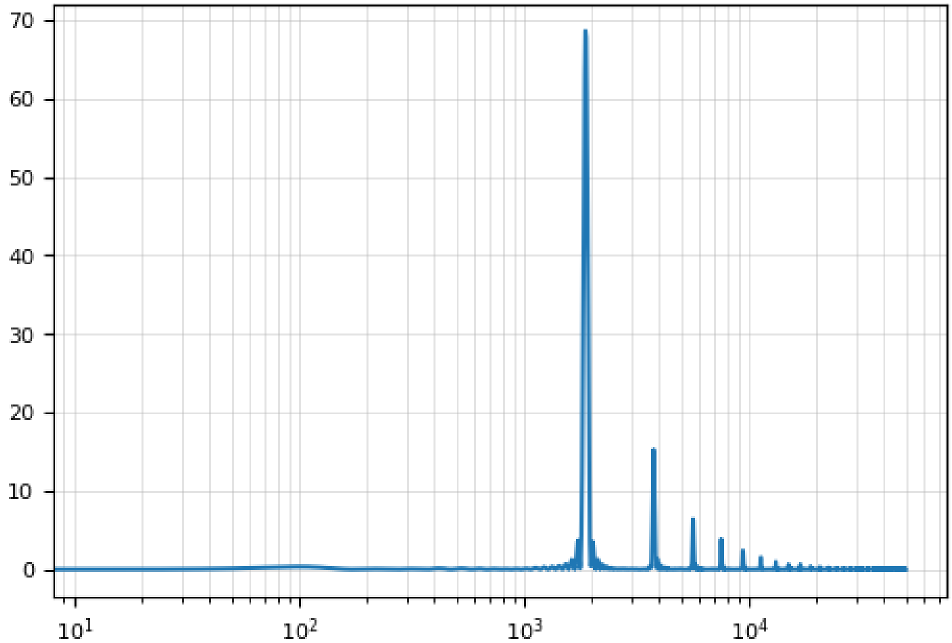
[13]:
```
df[1100:]['phs'].plot();
```



[14]:
```
fall_wf = df[1200:2200]['phs']
f, pxx_den = signal.periodogram(fall_wf, fs, nfft=8192)
f_peak = f[np.argmax(pxx_den)]
```

[15]:
```
plt.semilogx(f, pxx_den)
print(f'Peak freq: {f_peak:.2f} Hz')
```

```
Peak freq: 1879.88 Hz
```



Therefore cavity detune is at 1.8 kHz.

From crystal characterization lab, we know the resoance frequency is around 19.998 MHz, so this measurement confirms that the exact frequency is at 19.9982 MHz.

2.5 Adjust LLRF LO DDS to *tune* cavity

[16]:
```python
import sys
sys.path.append('..')
from dds.dds import calc_dds, reg2freq
```

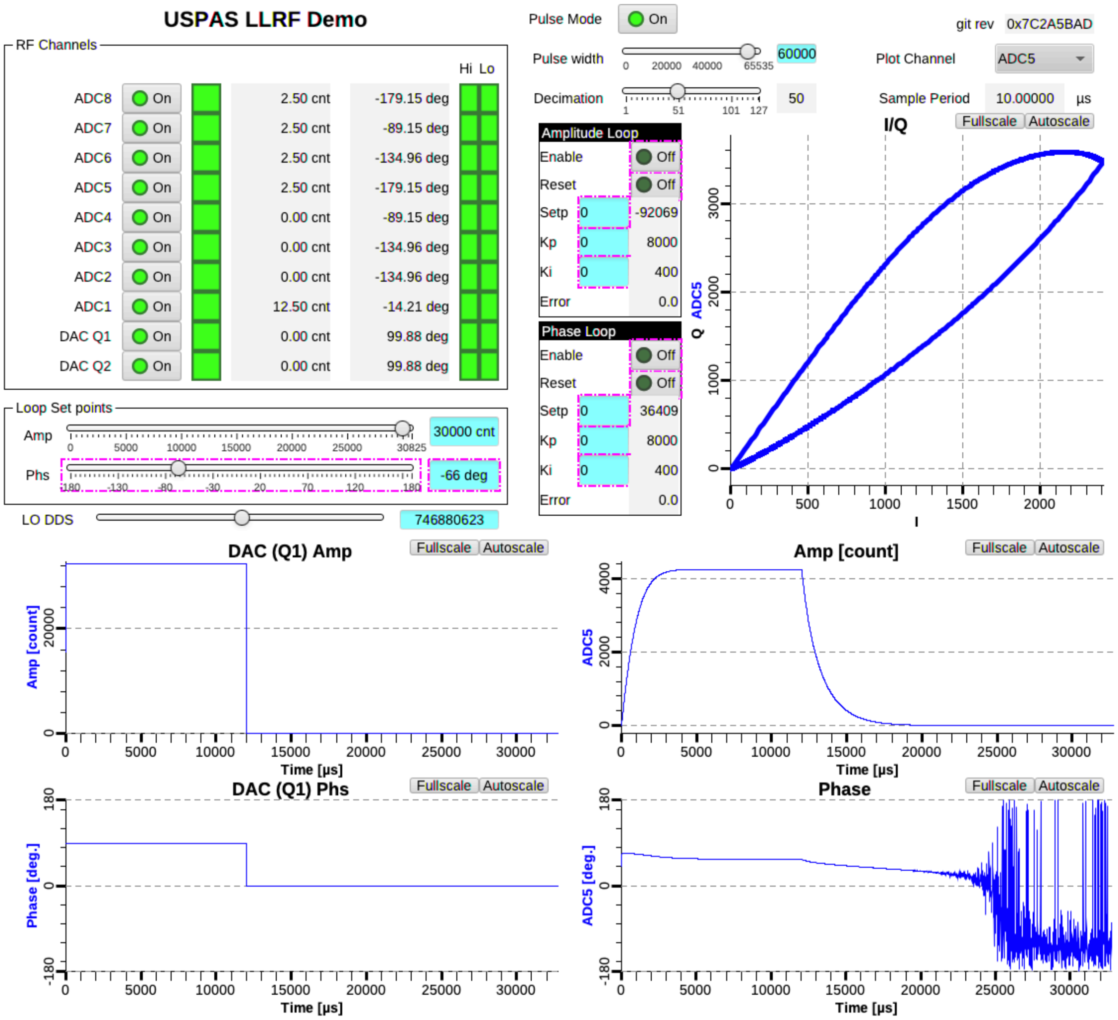
[17]:
```python
num = 4 * (1 - f_peak / 20e6)  # offset from 4
ph, pl, modulo = calc_dds(num, 23)
fdds = reg2freq(ph, pl, modulo, 115e6)
print(f'New DDS freq: {fdds/1e6:.3f} MHz')
```

```
major resolution:  109.673 Hz
minor resolution:  0.027 Hz
modulo resolution: 0.024 Hz
New DDS freq: 19.998 MHz
```

[18]:
```python
new_ph_step = (ph << 12) | pl
```

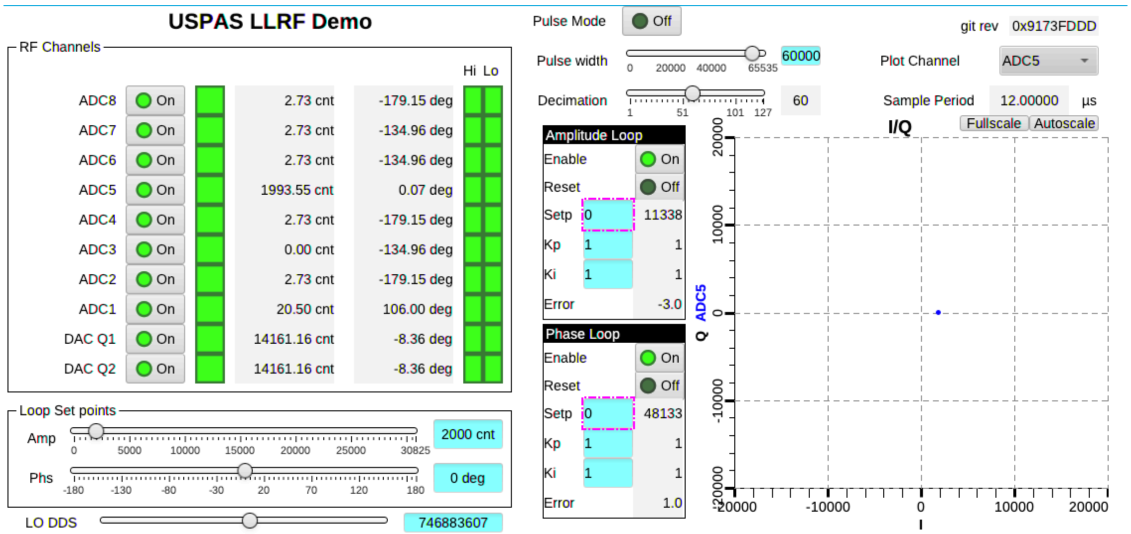
Adjust DDS frequency!

[19]:
```python
phase_step_pv = PV('USPAS:LLRF:reg_dds_phase_step')
phase_step_pv.value = new_ph_step
```



2.6 Close amplitude and phase loop

1. Disable pulse mode
2. Adjust amplitude setpoint to 2000
3. Set both loops Ki, Kp gain to 1
4. Reset both loops



[20]:
```python
caput('USPAS:LLRF:reg_pulse_mode', 0)
caput('USPAS:LLRF:Loop:AmpSetp', 2000)
caput('USPAS:LLRF:reg_Kp_amp', 100)
caput('USPAS:LLRF:reg_Ki_amp', 1)
caput('USPAS:LLRF:reg_amp_loop_reset', 1)
caput('USPAS:LLRF:reg_amp_loop_reset', 0)
caput('USPAS:LLRF:reg_amp_loop_enable', 1)

caput('USPAS:LLRF:Loop:PhsSetp', 1)
caput('USPAS:LLRF:reg_Kp_phs', 100)
caput('USPAS:LLRF:reg_Ki_phs', 1)
caput('USPAS:LLRF:reg_phs_loop_reset', 1)
caput('USPAS:LLRF:reg_phs_loop_reset', 0)
caput('USPAS:LLRF:reg_phs_loop_enable', 1)
```

[20]: 1

Increase integral and proportional gain and observe loop stability.

Tap on the crystal board and see what happens.