DEPARTMENT OF PHYSICS
Institute for Beam Physics and Technology

Stability Improvements at FLUTE
(Verbesserung der Stabilität von FLUTE)

Master thesis
of

Marvin-Dennis Noll

at the Institute for Beam Physics and Technology

Reviewer:	Prof. Dr.-Ing. John Jelonnek (IHM)
Second Reviewer:	Prof. Dr. Anke-Susanne Müller (IBPT)
Advisor:	Dr. Nigel Smale (IBPT)

15.11.2020 – 01.07.2021

Erklärung zur Selbstständigkeit

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde und dass ich die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung vom 24.05.2018 beachtet habe.

Karlsruhe, den 01.07.2021, _____

Marvin-Dennis Noll

Als Prüfungsexemplar genehmigt von

Karlsruhe, den 01.07.2021, _____

Prof. Dr.-Ing. John Jelonnek (IHM)

Contents

List of Figures

List of Tables

1. Interfacing FLUTE

This chapter covers methods on interfacing the FLUTE accelerator, that is how to read diagnostic measurements into the control system from FLUTE and how to influence the electron acceleration appropriately to achieve stabilization.
In this chapter *input* and *output* refer to the view from the control system.

1.1 Inputs

FLUTE uses the *Experimental Physics and Industrial Control System (EPICS)*[1] for control of various parts of the accelerator, to send real time data to be archived and to build user interfaces via *Control System Studio (CSS)*[2], a JAVA based integrated development environment (IDE).

EPICS offers client/server and publish/subscribe paradigms to access data in so called process variables (PV) through channels. Modules are usually written in the C programming language. To ease the access to EPICS channels in programs written with the Python language, the package *PyEpics*[3] can be used. Since all data of interest as input for the control system can be extracted through an EPICS channel, the next section deals with using PyEpics to obtain the data.

1.1.1 Accessing EPICS channels in Python with PyEpics

Before usage PyEpics needs to be installed, e.g. with `pip3 install pyepics` from the *PyPi* repository. If the computer running the Python code can reach the EPICS CA repeater on the machine network, the connection is established automatically in the background. To get a channel value asynchronously, i.e. at an arbitrary time, the function `caget(pvname)` can be used with the name of the desired process value, see **??**.

Listing 1.1: Using `caget()` to get the value of an EPICS process value

```
1 from epics import caget
2 print(f"Cavity RF power: {caget('F:RF:LLRF:01:GunCav1:Power:Out')}")
```

Another way is to setup a channel object and create a subscription with an user defined callback function that is executed each time the process variable changes. This implements synchronous access to the PV and can be compared to an interrupt rather than polling the variable as in Listing 1.1.
For a non trivial example see Listing 1.2. In this program, the time differences between new values and their statistics are printed to the console.

Listing 1.2: Using a user defined callback function to access an EPICS process value

```
1 from epics import ca
2 import time
3 import numpy as np
4
```

```
5  dts=np.array([])
6  lastCalled=time.time()
7
8  def call(pvname, value, **kwargs):
9    global lastCalled, dts
10   now=time.time()
11   dt=now-lastCalled
12   lastCalled=now
13   dts=np.append(dts, dt)
14
15 chid=ca.create_channel("F:RF:LLRF:01:GunCav1:Power:Out")
16 _ , _ , eventID=ca.create_subscription(chid, callback=call, use_time=True)
17
18 while(True):
19   time.sleep(2)
20   print(f"N:{len(dts)},mean:{np.mean(dts)},min:{np.min(dts)},max:{np.max(dts)},std:{np.std(dts)}")
```

1.1.2 Properties of the Available Process Variables

In this section some process variables that may be used as inputs for the control system are analyzed. These are:

- `F:RF:LLRF:01:GunCav1:Power:Out Value`: The RF power measured immediately before the cavity
- `F:AX:DAQDT:01:1:Wave:05:Sample Value`: The charge measured with a Faraday cup (RadiaBeam Technologies FARC-04 [4]) and amplified with a charge sensitive amplifier (PCB 421A25 [5])
- `F:INJ-1:Gun:01:Temperature:Body Value`: The body temperature of the cavity

In Figure 1.1 all three are plotted for a duration of 15 min without any interference to the system and the system being in steady state operation.

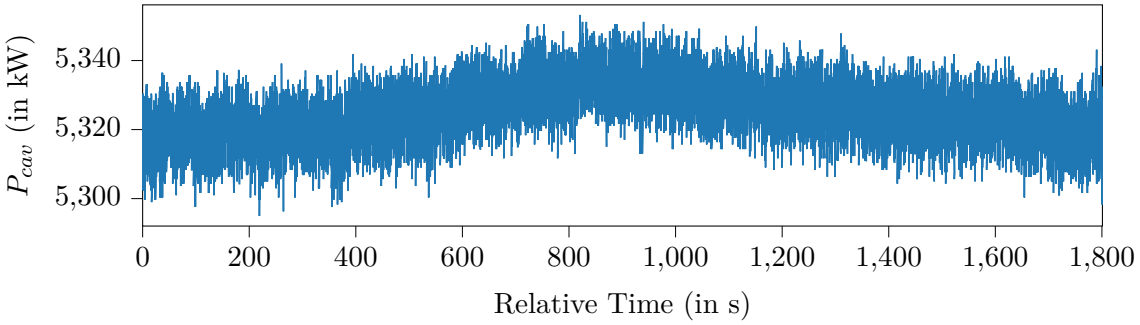
In addition the sample times of the process variables is examined if the method with a custom callback is used (see Listing 1.2) or the data is extracted from the archive. The differences in the sample times are calculated according to

$$\Delta = t_{n+1} - t_n. \tag{1.1}$$

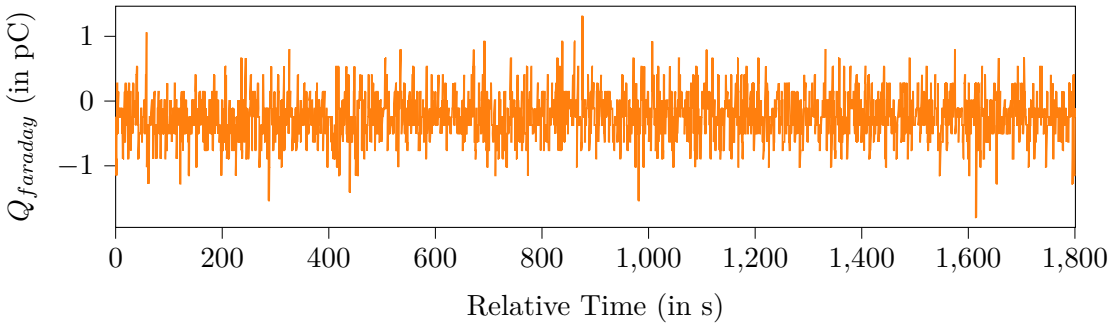
Then a histogram with the relative frequency on the y axis is used as an estimator for the probability density function of the sample time intervals (see Figure 1.2). The histogram shows that the time series resulting from recording process variables out of the EPICS system are highly unevenly spaced. Thus the data needs to be converted to posses evenly spaced sample times to use common signal processing methods like the DFT or digital LTI filters. For offline analysis of prerecorded data, it can easily be resampled to a fixed time grid. But for online operation of a filter or a whole control system it is not possible to use an arbitrary resample method because they are often non causal or introduce significant group delay when made causal. Instead calling `caget()` with a (software-) timer can be used to get evenly spaced samples online.

Table 1.1: Comparing quantization steps

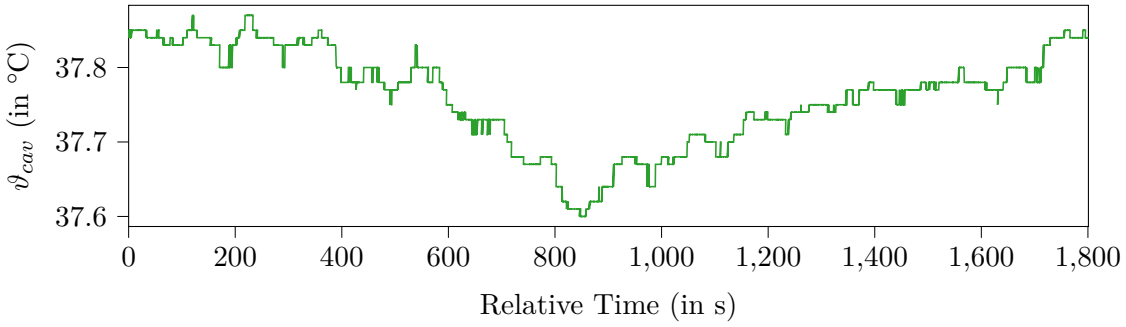
PV	N_{unique}	q_{avg}	q_{norm}
F:RF:LLRF:01:GunCav1:Power:Out Value	84	0.6935	0.011 904
F:AX:DAQDT:01:1:Wave:05:Sample Value	22	0.015	0.045 45
F:INJ-1:Gun:01:Temperature:Body Value	18	0.1294	0.055 55



(a) Cavity RF power `F:RF:LLRF:01:GunCav1:Power:Out Value`



(b) Faraday cup charge `F:AX:DAQDT:01:1:Wave:05:Sample Value`



(c) Cavity body temperature `F:INJ-1:Gun:01:Temperature:Body Value`

Figure 1.1: Comparing the quantization noise of three process variables

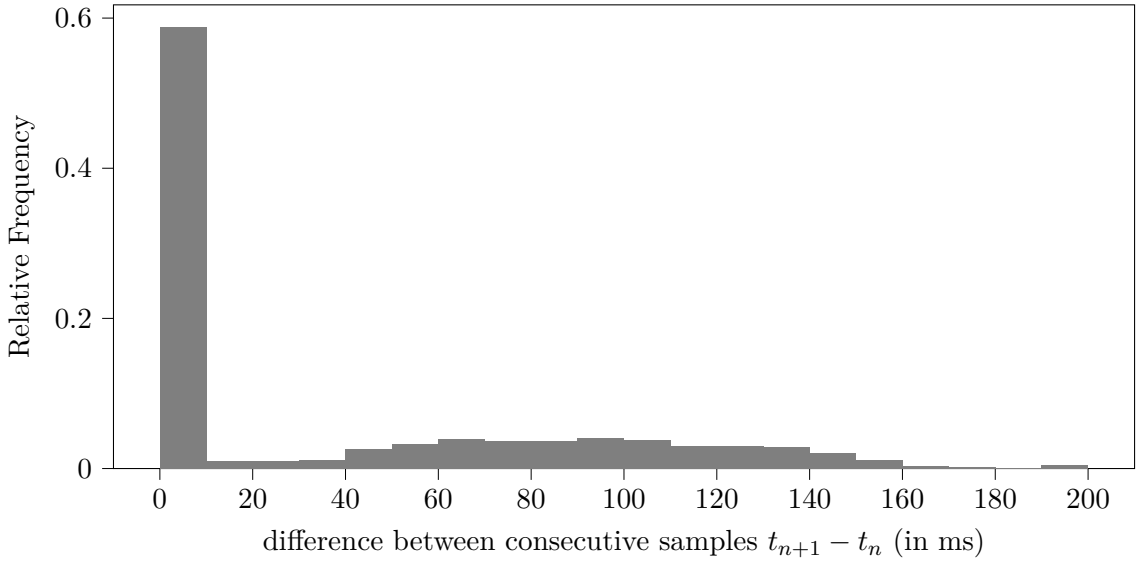


Figure 1.2: Histogram of the sample time intervals Δ of the plots in Figure 1.1

1.2 Output: Controllable RF Attenuator

The output signal computed by the control systems has to have a way of influencing the RF power send to the cavity. This could be done over an EPICS channel (e.g. with the PyEpics function `caput()` to set the value of a process variable via a channel access). However to make it possible to move the control system from a general purpose personal computer to a dedicated digital signal processor, FPGA or similar in the future, using a physical device in the signal path is preferred.

1.2.1 Defining Requirements

Table 1.2: Requirements for the controllable RF attenuator

Requirement	Value/Range
minimum attenuation	
maximum attenuation	
attenuation resolution	
setup time	
operating temperature range	
supply voltage	
control voltage	

1.2.2 Evaluation of the ZX73-2500-S+ Controllable RF Attenuator

1.2.3 Conclusion

Appendix

A Lab Test and Measurement Equipment

A.1 Benchtop multimeters

A.1.1 Agilent 34411A

Table A.3: Agilent 34411A specifications

Specification	Value
	DC volt
Digits	6 1/2
Measurement method	cont integrating multi-slope IV A/D converter
Accuracy (10 V range, 24 hours)	0.0015 %+0.0004 % (% of reading + % of range)
Bandwidth	15 kHz (typ.)

Table A.4: Agilent 34411A some SCPI commands

Description	Example command	Example return
Read current measurement	`READ?`	`+2.84829881E+00` (2.848 V)

A.1.2 Keysight 34470A

Table A.5: Keysight 34470A specifications

Specification	Value
	DC volt
Digits	7 1/2
Measurement method	cont integrating multi-slope IV A/D converter
Accuracy (10 V range, 24 hours)	0.0008 %+0.0002 % (% of reading + % of range)
Bandwidth (10 V range)	15 kHz (typ.)

Table A.6: Keysight 34470A some SCPI commands

Description	Example command	Example return
Read current measurement	`READ?`	`+9.99710196E+00` (9.997 V)

A.2 Data Acquisition/Switch Unit

A.2.1 Keysight 34972A

Table A.7: Keysight 34972A specifications

Specification	Value
34907A (Multifunction module)	
DAC range	±12 V
DAC resolution	16 bit ($^{24\,V}/_{2^{16}} = 366.21\,\mu V$ per bit)
DAC maximum current	10 mA
34901A (20 channel multiplexer)	

Table A.8: Keysight 34972A some SCPI commands

Description	Example command	Example return
Read current measurement	`READ?`	`+2.00200000E+01` (20.02 °C)
Set DAC voltage of ch 204 to 3.1 V	`SOUR:VOLT 3.1,(@204)`	

A.3 Oscilloscopes

A.3.1 Tektronix MSO64

Table A.9: Tektronix MSO64 specifications

Specification	Value
Bandwidth	6 GHz
Sample rate	25 GS/s
ADC resolution	12 bit
DC gain accuracy (@ 50 Ω, >2 mV/div)	±2 %

Table A.10: Tektronix MSO64 some SCPI commands

Description	Example command	Example return
Read mean of measurement 1 (current acq.)	`MEASUrement:MEAS1:RESUlts:CURR:MEAN?`	`3.0685821787408`

A.4 RF signal generator

A.4.1 Rohde and Schwarz SMC100A

Table A.11: Rohde and Schwarz SMC100A specifications

Specification	Value
Frequency range	9 kHz to 3.2 GHz
Maximum power level	17 dBm
SSB phase noise (@ 1 GHz, $f_o = 20$ kHz, $BW = 1$ Hz)	−111 dBc
Level error	<0.9 dB

Table A.12: Rohde and Schwarz SMC100A some SCPI commands

Description	Example command	Example return
Set RF power level to 10.5 dBm	`SOUR:POW 10.5`	
Set RF frequency to 3.1 GHz	`SOUR:FREQ:FIX 3.1e9`	
Enable the RF output	`OUTP on`	

A.5 RF power meter

A.5.1 HP E4419B

Table A.13: HP E4419B specifications

Specification	Value
Digits	4
Accuracy (abs. without power sensor)	±0.02 dB
Power probe: E4412A	
Frequency range	10 MHz to 18 GHz
Power range	−70 dBm to 20 dBm

Table A.14: HP E4419B some SCPI commands

Description	Example command	Example return
Measure power on input 1	`MEAS1?`	`+2.89435802E+000` (2.894 dBm)

A.6 Vector Network Analyzer

A.6.1 Agilent E5071C

Table A.15: Agilent E5071C specifications

Specification	Value
Frequency range	9 kHz to 8.5 GHz

A.7 Phase noise analyzer

A.7.1 Holzworth HA7062C

Table A.16: Holzworth HA7062C specifications

Specification	Value
DUT input frequency	10 MHz to 6 GHz
Measurement bandwidth	0.1 Hz to 40 MHz offsets

Bibliography

[1] L. R. Dalesio, M. R. Kraimer, and A. J. Kozubal, "EPICS Architecture'', *ICALEPCS*, vol. 91, 1991.

[2] Control System Studio. (2021). Control System Studio, [Online]. Available: `https://controlsystemstudio.org/about` (visited on 05/26/2021).

[3] M. Newville and A. Gratton. (2019). PyEpics: Python Epics Channel Access, [Online]. Available: `https://cars9.uchicago.edu/software/python/pyepics3/`.

[4] RadiaBeam, *Faraday Cups*. [Online]. Available: `http://www.radiabeam.com/upload/catalog/pdf/14272334342015-03-24_faraday-cups.pdf`.

[5] P. Synotech, *PCB 421A25 Charge Amplifier*.