1.3 Muon Pysics

Post-Lab:

* Data analysis and question discussion:
  + For task 3:

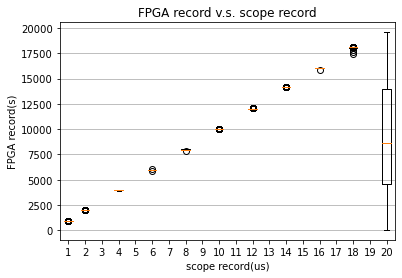
Increasing the input magnitude will make the output signal saturated, even the saturated signal output will in the same frequency with the unsaturated one, but as the signal is clipped, showing the flat, it will cause incorrect reading of timing of FPGA. The light signal indicates that a Muon has decayed at this moment.

* + For task 4

In the records, there might be several errors and I used a Python script to pre-process the data [datapro.py]. Code will be attached at the end of the Post-Lab document. After run it, all data will import to Data.csv

1) In step 1, here we recorded 10000 Muons, and 874 of their decays were recoded. Most of them decayed in 560ns or 540ns, and one of them decayed in 740ns and another one in 1740s. The mean decay timing is 558.88ns with standard division 41.02.

2) In step 2, for each time step, we recorded 10000 Muons, as that in step 1. With pre-processing, errors were removed. With plotting the figure, every point is in perfect linear position, except when scope record was 20µs. The reason is that when the timeout of signal pulse is greater than about 20µs, the FPGA will hardly receive Muons decaying but signals mixed with background noise and system errors. Then the records seemed to be out of order with huge errors and, in other words, make little sense. Showing as Figure 1, nearly or records lied in narrow ranges, even with their extreme data, but the 20µs data was difficult to analyze. To draw this boxplot, I composed a Python script and it will be attached at the end of the Post-Lab document [bplot.py].

 Figure 1

3)When the timeout set to around 21µs, there is few decays could be detected. Though, when timeout reached to 20µs, detection would be difficult as what mentioned above.

4)By reading the records, the minimum step is 20ns, which indicated that the bin width of the FPGA internal timing is 20ns. With considering that when the signal absent, the background noise would cause decay records with timing 60ns or 80ns. Hence if the records over 100ns, they could be recognized as a decay happened.

* + For task5 and task6

The set of the equipment in this lab session was the same as beginning. Switching of the pulse, connecting PMT output with input and setting HV to 800V. As this task is for examining the effects of the discriminator threshold on time interval of Muon decaying.

1)Initially, the threshold was set to 100mV. The result was not in the shape as how we expected and how it should be, as the result of a relatively high background rate, leaded by such insufficient discriminator threshold.

2)Following is the 250mV threshold. Viewing the pattern, we could figure out the descending exponential curve. Though, comparing the values located in different ranges, we found that the curve was not always decrease, i.e. there was still high background rate.

3)Detecting the decaying of muons with discriminator threshold 395mV, we obtained the nearly ideal result.

4)Details of the background rate:

When threshold is 100mV,

When threshold is 250mV,

When threshold is 395mV,

Obviously, with a higher threshold, the decay rate is smaller, and the background rate is lower, indicating better result. Therefore, 100mV is not a good choice.

5)If the threshold is 100mV and all the counts were background occurring randomly in time with the measured rate, the probability is:

6)If a muon dose not decay, it is the same as its original. In other words, we could consider a undecayed muon is an identical particle of a generated muon. Therefore, when the muon gets into FPGA, we could also get the correct records of its decaying.

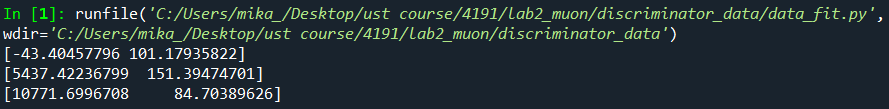
For task7:

Using 2µs as the step of intervals, I composed a Python program to output the number of the muons located in each interval. Show as below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threshold** | **2us** | **4us** | **6us** | **8us** | **10us** | **12us** | **14us** | **16us** | **18us** | **20us** |
| 100mV | 94 | 110 | 102 | 98 | 86 | 97 | 96 | 114 | 101 | 107 |
| 250mV | 867 | 392 | 228 | 170 | 119 | 128 | 109 | 93 | 122 | 137 |
| 395mV | 1496 | 611 | 219 | 95 | 41 | 17 | 17 | 15 | 12 | 10 |

Table 1, data from decay\_table.txt created by data\_fit.py

With this table, we could fit the exponential function to the data with Python: [data\_fit.py]

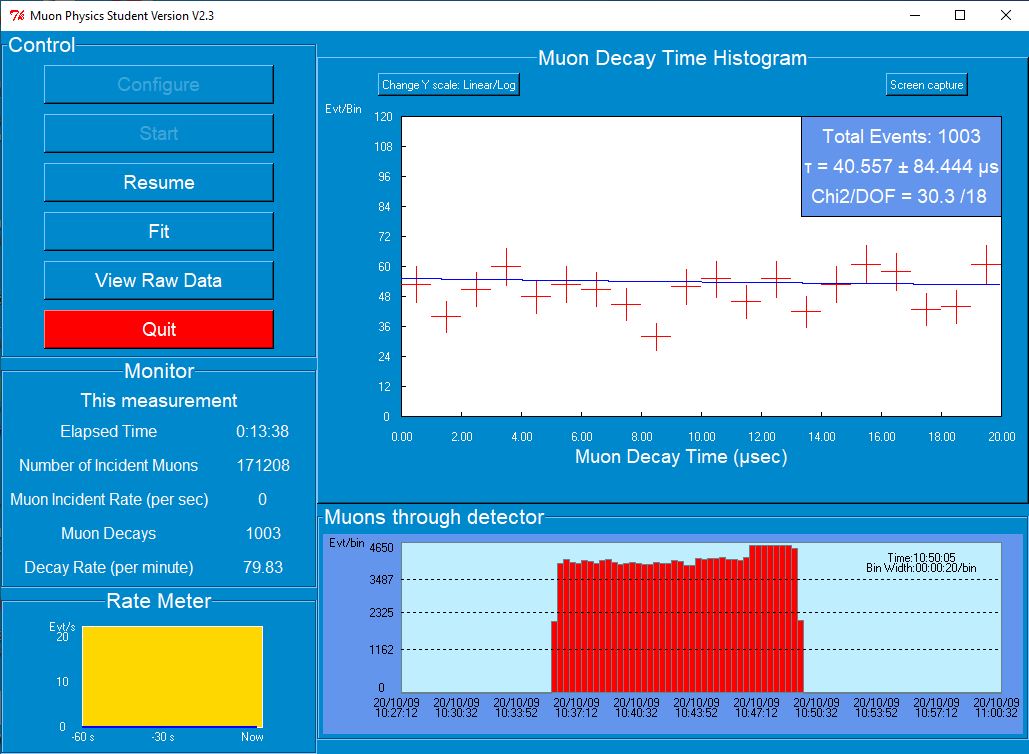


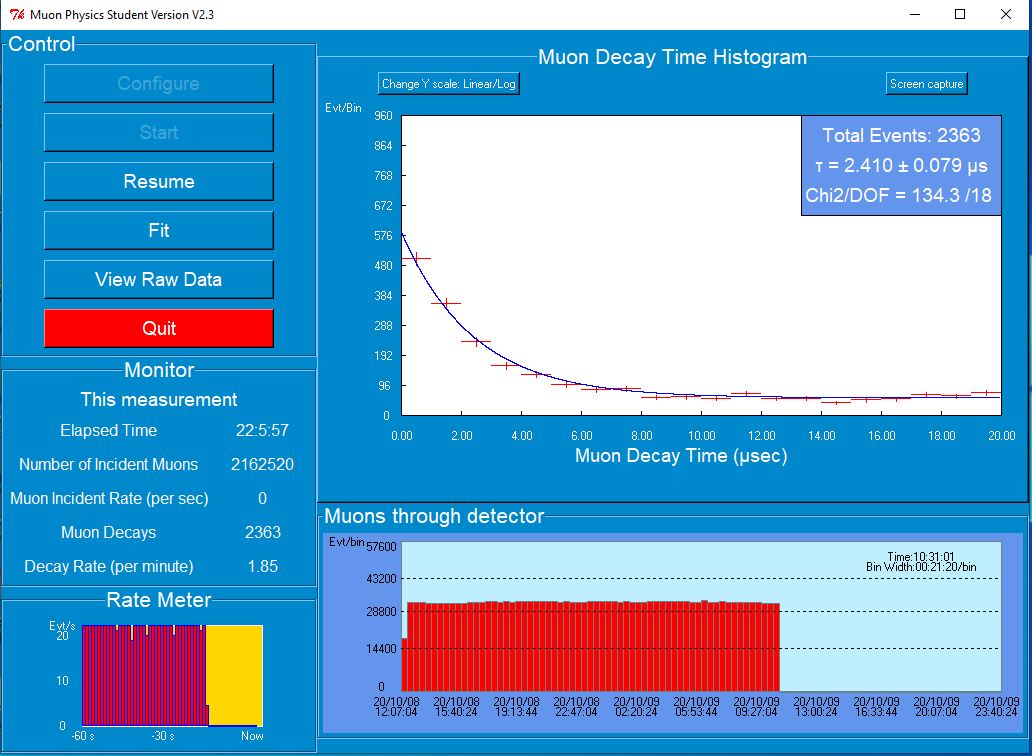
i.e. for threshold = 100mV, , showed as Figure2

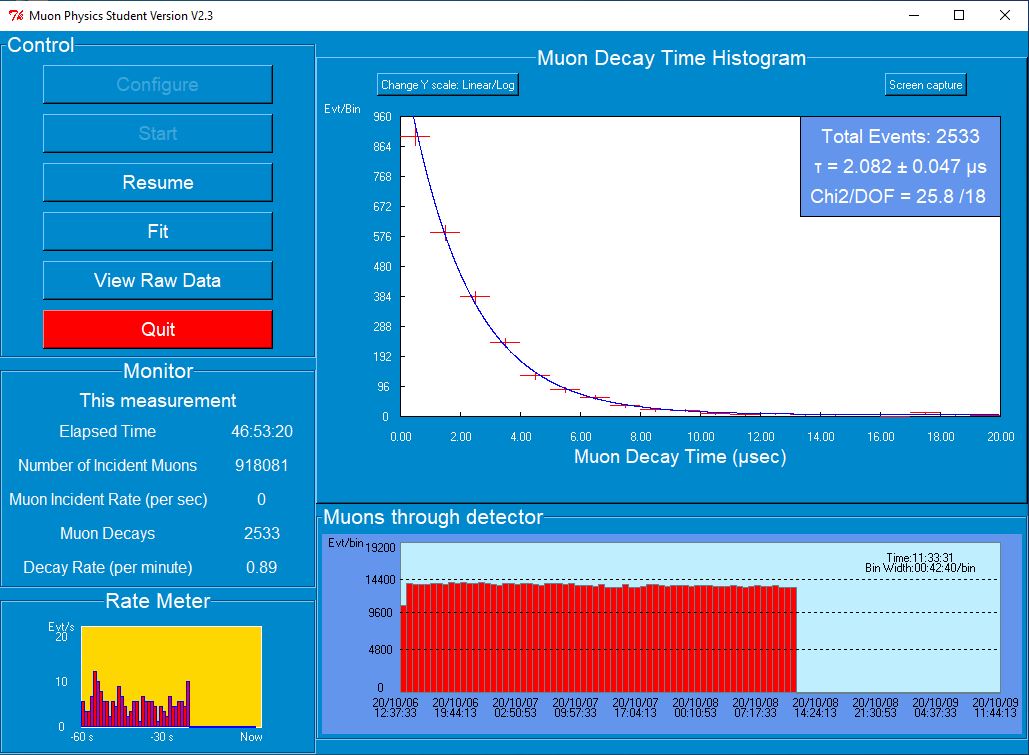
for threshold = 250mV, , showed as Figure3

for threshold = 395mV, , showed as Figure4

(all t in µs)

 Figure 2

 Figure 3

 Figure 4

* + For task8:

Here we use the threshold 395mV as the data source. We have:

Then,

* + For task9:

Based on the data obtained, the Fermi coupling constant：

The ideal value is , s.t. the error is:

* + For task10:

To calculate the mean probability, the mean number of success points and the variance, I composed a Python program to extract 2000 random records from the threshold = 395mV data cluster, totally 2533 records. The program will attached at the end of this post lab document [success.py]. Figure5 is the statistics data:

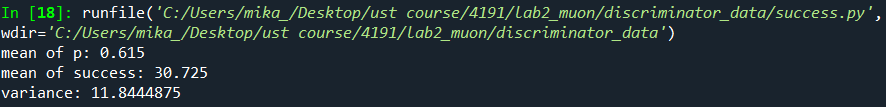
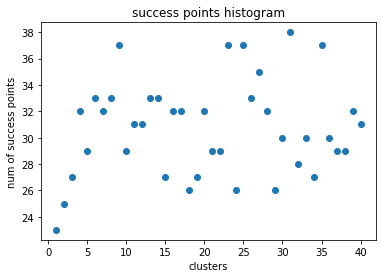


Figure 5

And Figure6 is the histogram of 40 clusters of records:

 Figure 6

The results roughly agreed on the theoretical values and the errors are small. The sources of error might be the limited size of the records, the incomplete records clusters, the background noise, i.e. system error, and the ignorance during calculating.

Code list:

* [datapro.py]

# -\*- coding: utf-8 -\*-

"""

Created on Tue Sep 29 21:04:22 2020

@author: mika\_

"""

import numpy as np

index = 20

while index<21:

t = []

fin="dx=%dus.txt"%index

fn="resolution.txt"

fout="dx=%dus.csv"%index

fnout="resolution.csv"

f= open(fn, 'r')

line=f.readline()

print(fin)

while line:

decay = line.split(" ")

dint = int(decay[0])

if dint<=21000:

t.append(dint)

line=f.readline()

f.close()

avg=np.mean(t)

i=0

p=[]

while i<t.\_\_len\_\_():

p.append(t[i])

i+=1

print("Processing: %d of %d"%(i,t.\_\_len\_\_()))

print('valid values: %d'%p.\_\_len\_\_())

print('mean: %f'%np.mean(p))

f = open(fnout,'w')

for x in p:

text = str(x) + "\n"

f.write(text)

f.close()

t.clear()

p.clear()

if index==1:

index=2

else:

index+=2

* [bplot.py]

# -\*- coding: utf-8 -\*-

"""

Created on Wed Sep 30 13:29:37 2020

@author: mika\_

"""

import matplotlib.pyplot as plt

import csv

table=[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]]

with open('Data.csv', 'r', encoding="utf-8-sig") as csvfile:

dreader = csv.reader(csvfile)

for row in dreader:

k=row[0].split(";")

for i in range(len(k)):

k[i].strip()

if k[i]!='':

if i==0:

index=0;

else:

index=(i-1)\*2+1

table[index].append(int(k[i]))

fig = plt.figure()

ax = plt.subplot()

ax.boxplot(table)

ax.grid(axis='y')

plt.title('FPGA record v.s. scope record')

plt.xlabel("scope record(us)")

plt.ylabel("FPGA record(s)")

plt.show()

* [data\_fit.py]

# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 18 22:03:58 2020

@author: mika\_

"""

def field(x):

if x<=2000:

return 0

if x<=4000:

return 1

if x<=6000:

return 2

if x<=8000:

return 3

if x<=10000:

return 4

if x<=12000:

return 5

if x<=14000:

return 6

if x<=16000:

return 7

if x<=18000:

return 8

return 9

def get\_data1(str):

f = open(str, 'r')

t = np.zeros(10, dtype=int)

line = f.readline()

while line:

decay = line.replace('\n', '')

x = int(decay)

t[field(x)] +=1

line = f.readline()

f.close()

return t

def get\_data2(str):

f = open(str, 'r')

t = np.zeros(10, dtype=int)

line = f.readline()

while line:

decay = line.split(' ')

x = int(decay[0])

t[field(x)] +=1

line = f.readline()

f.close()

return t

import numpy as np

fn1 = '100mVout.csv'

fn2 = '250mVout.csv'

fn3 = '395mV.data'

t1=get\_data1(fn1)

t2=get\_data1(fn2)

t3=get\_data2(fn3)

fn=['100mV', '250mV', '395mV']

t=[t1,t2,t3]

decay\_time=np.array([-2, -4, -6, -8, -10, -12, -14, -16, -18, -20])

f = open('decay\_table.txt', 'w')

f.write("Threshold,2us,4us,6us,8us,10us,12us,14us,16us,18us,20us")

for i in range(3):

f.write('\n')

f.write(fn[i])

for j in range(10):

f.write(str(t[i][j])+',')

print(np.polyfit(np.exp(decay\_time), t[i], 1))

f.close()

* [success.py]

# -\*- coding: utf-8 -\*-

"""

Created on Thu Oct 22 19:45:24 2020

@author: mika\_

"""

import numpy as np

import random

import matplotlib.pyplot as plt

t=2082

fn='395mVout.csv'

f = open(fn, 'r')

l = f.read()

f.close()

decaystr = l.split()

decay=[]

group=np.zeros((40,50), dtype=int)

p=np.zeros(40)

success=np.zeros(40, dtype=int)

for i in range(len(decaystr)):

decay.append(int(decaystr[i]))

for i in range(40):

for j in range(50):

k = random.randint(0, len(decay)-1)

group[i][j]=decay[k]

np.delete(decay,k)

for i in range(40):

n=0

for j in range(50):

if group[i][j]<=t:

n+=1

p[i]=n/50

success[i]=n

print('mean of p: %.3f'%np.mean(p))

print('mean of success: '+repr(np.mean(success)))

print('variance: '+repr(50\*np.mean(p)\*(1-np.mean(p))))

plt.scatter(range(1,41),success)

plt.title('success points histogram')

plt.xlabel('clusters')

plt.ylabel('num of success points')