Informal Manual on Usage of Timepix3 and Diamond Detector in METU-DBL

Baran Bodur[1]

[1] *Your colleague, student and friend*

July 24, 2017

Contents

1 Preface

I write this short manual, not to present my work on detectors in any formal way, but to give my perspective about Timepix and Diamond to the people responsible from these detectors after my departure, in the hope of helping them. I always hoped I will see our preliminary tests both because I wanted and I feel responsible for conducting my part on it. Certain unfortunate situations lead us to this point, in which I cannot conduct my responsibilities fully, however the least I can do is to write this brief manual explaining the variable parameters that might be used to calibrate the detectors.

I tried to write the MATLAB scripts and functions neat and full of comments, so that people with certain knowledge of physics, programming and our detectors can easily follow. I write this solely to speed up the process of simple parameter modifications, which might be necessary even when everything is going well. If one thinks the analysis codes are wrong or inaccurate, or simply wishes to understand how it works more deeply, they should read my more formal reports and presentations and of course spend some time with the analysis scripts and all the functions coming with them.

The MATLAB scripts are embedded in a LabVIEW program and some of the parameters I mentioned can be changed directly from there, so you may not even need to go down to MATLAB, but just change parameters from the user interface.

I will not write much about LabVIEW design, mostly because Serdar Aydın and İlker Şahin knows LabVIEW and the sequence of our program rather well. Unfortunately, I wrote my parts of these codes in short time and without proper experience on LabVIEW, therefore these programs might be both buggy and very hard to understand (because of lack of proper commenting and design). Although according to multiple trials they seem to do the trick in many different scenarios we experimented, if something goes wrong about that part, I am the one to blame.

This manual describes only operation specific to our use of the mentioned devices. When I say turn on the power, I might not say it requires 5 V, 1 A, or I might not describe how a particular software is installed. These are things one can find on the internet, in the datasheet of the devices. When I say 'The Wiring Diagram' it is obviously the 'Wiring Diagram' of the METU-DBL preliminary test, it might change in METU-DBL itself, or you may perform these operations somewhere else with spare equipment, then you will need to perform a little translation according to your specific needs.

Finally, by writing this manual I am in no way trying to say "Just read the manual and do not ask me anything". I am willing to help as much as I can (I am not only willing, but I like to help by the way and if you do not notify me while you are performing first measurements, I would be very angry), but the communication can be hard, due to work conditions and time difference.

2 Timepix3

2.1 Setting Up Measurement

1. Make sure Timepix3, AdvaDAQ and the SBC (Single Board Computer, aka lattepanda) of Timepix3 is positioned correctly in the target area.
2. Carefully connect the VHDCI cable between ADVADAQ and Timepix3, the cable is valuable beyond words.
3. Connect AdvaDAQ to the SBC in the target room with USB 3.0 cable to the USB 3.0 port of the computer, also connect an ethernet cable between SBC and the ethernet switch in the target room.
4. Connect appropriate power cables (check the wiring diagram of the detectors) to both AdvaDAQ and SBC. In addition, connect 180 V bias voltage to the Timepix3.
5. Make sure both Timepix and SBC are properly cooled, and turn on the power via switches in the control room. Monitor currents for a while to ensure their operation is correct.
6. Connect to SBC from the control computer via 'remote desktop' application of windows (There are plenty of resources on the internet about this procedure)
7. Open Pixet application on the desktop of SBC, if connected to Timepix the device image on the top-left side would be blue, if not there is a problem with connection, current supplied to devices, or one of the devices is overheating.
8. Check Timepix3 temperature regularly, try not the let it operate over 70 Celcius, ideally below 60 Celcius.
9. Check Threshold value which should be around 85-86 keV (as high as possible in energy) or if it is not calibrated to energy and is written in terms of DAC value, then it should be as low as possible (aka. 1, for some reason lowest value corresponds to highest energy, took me some time to figure out)
10. From 'Configure DACs' on the down-left side check IKrum (Krumenacher Current) value. This is a measure of signal decay speed, the higher (means faster decay) is generally the better, if it does not cause too much oscillation (causing double counts). I did not see problems related to that with my low activity alpha source even at max value of 255, but to be safe you can use it as low as 150. After all, I wrote the analysis code to handle pile-up.
11. Go to 'Tools-Python Scripting' and open our server script currently named 'pre-test-meas1.py'. This script controls Timepix3 according to a connected host's desire (you will connect with the LabVIEW control script). In other words this is the interface between the control computer and Timepix3. In here there are three very important parameters, 'no of frames' and 'measurement time' (could have slightly different names but meaning shoul be same) at the very top of the script, and 'port number' which is either a variable itself, or is inputed directly in the open port section

of the code. These three variables must be equal to the corresponding variables in the LabVIEW and MATLAB scripts running in the control computer. You might need to change them between different measurements, and remember to change them on both the python script and on the LabVIEW control interface. Difference in port number disables your communication with the SBC through LabVIEW, whereas difference in other two values cause very wrong flux measurements without any warning. So be careful about this!

12. Look where it saves the measurement file (should be a shared folder between control computer and SBC) and make sure the directory is empty. This process is automatically handled by the LabVIEW unless you quit it without finishing a measurement. Better check the directory there should be no folders under 'meas/meas1/' directory.

13. You are ready to run the python script, beware stopping it would be hard (you generally need to quit python scripting tool entirely or pixet). In that case, the previously used ports would be left open (during the same power-on of computer so do not worry if it is the first time you use it after turning on the computer), so before running the script again you might want to change the port number (Beware! you need to change port no on the LabVIEW too.).

14. Go back to the control computer and run the Timepix3 control program via LABVIEW (it has a name like 'METU-DBL-Timepix-v1.vi' which is in the 'timepix- readout' folder in desktop, you may need to further browse to 'labview-measurement' and down to the folder named 'current', obviously 'old' means an older version)

15. Run the program, check the switches at the bottom of the window. Decided connections to XY drivers, connection to Timepix3 and other options first. If you try to change them during the course of the program, press them only once, the visual effect may take some time to change.

16. There is an important situation you need to check if you decide to connect to XY drivers and move the detector around. Since all the detectors tightly put into a very small area, which prevents distortion of the beam energy spectrum and uniformity before measurement, detectors might crash into each other while moving. In addition you may need to send the XY stages to their origin points (not the origin according to our coordinate system, the origin hard-coded into the drivers), which might be on a collision course with the scintillating fibers. You might need to engineer a move sequence so that you can sen Timepix3 to origin without colliding it with scintillators, then move the scintillators back to their original position, finally make sure nothing lies in the scan area and Timepix3 park area. If need be you may change the park position from the LabVIEW (Left pane, Give Control Command TAB). If drivers give a fault during this process or after, you may use the RESET command at the same TAB.

17. On the left side, in the first tab, enter the horizontal and vertical range you want to scan. Arrange other controllable parameters to your liking, also check the parameters in this and next (first and second) tabs are the same as three parameters mentioned in the python server script.

18. Press 'Enter Data' button and check the corrected ranges, if you found them to your liking, go to the second tab and press 'Start Measurement'. The detector will automatically scan the entire area, add flux measurements into a matrix, plot it in the first tab of the right pane. You can see number of measurement positions and the current measurement number on the left pane (first or second tab). Measurement at a position typically lasts half the seconds of the frame number parameter, and analysis lasts between 7-10 seconds on the control computer (If you have a good Linux computer it lasts (1-3 seconds). You have the option of not analyzing the data until all measurements are concluded, this will reduce the time detector spends under radiation, but the flux graph could not be updated after each measurement, since analysis were not performed yet. Check bottom left pane during the measurement, the execution status is shown there (could be a little buggy). In addition, you can check the temperature of the Timepix3, through the second TAB of the left pane, try to leave your screen on this TAB, and check the temperature (it will not be updated during the measurement but between the measurements). If you are worried, check the temperature through remote desktop (on SBC running Pixet).

19. After the completion of the measurement and analysis of each position, you can go to the analysis TAB, and give a range for analysis (you might want to give a smaller range than your scan range to get rid of unfilled parts or to select only the uniform area at the center). After selection, press analyze data and check uniformity values at the same TAB or newly filled graphs at the TABs in the right pane. If any of these graphs were filled before you perform a measurement or analysis, they are from older data as you might have guessed.

20. All measured fluxes are written into a file, which is saved in a folder according to the computer's date. The raw data from Tpx3 are saved in that folder, in case you might want to analyze it with a different analysis code (Maybe yours is more efficient or accurate then mine) The analyzed data is also saved. When you open 'analyzed data.txt', the first row is X positions, second is Y positions, and the rest is the flux matrix (top left corresponds to first X and first Y position, X increases to right, Y increases downwards)

21. 'I firmly believe, self-education is the only kind of education there is.' -Isaac Asimov. I also believe the same phrase, you will learn a lot more by simply looking (and understanding) to the codes, trying various scenarios on the control command, by finding my design flaws (there are certainly some). Just be on the safe side while trying something with the hardware, both Timepix3 and the VHDCI cable are pricless (for real, in the sense that they are not commercial).

2.2 Analysis Parameters of Timepix3 Data

This section is important because you can understand what parameters do what without understanding the whole code and experimenting with it. This is ideally what you must do, but it might take a while, and you may not have that much time (we were on a tight schedule when I left).

I will describe the analysis on the code itself seen in Script 1. However, you are able to control 'no of frames' (line 19), 'row divide' (line 24), 'column divide' (line 25), 'shutter length' (line 29), 'filename' (line 46) from the control program (They have their names on them, so you should find them rather easily, I already mentioned them in the previous section) and in ideal conditions that is all you need.

Timepix Analysis Main Script (Script 1)

```
1  %% License and Manual
2  % Author: Baran Bodur
3  % Timepix3 Flux Measurement
4  % Idea: Do not use what you do not know
5  % Timepix3 Mode ToA + ToT, Frame Based, VCO on or off
6  % Timepix3 Parameters Ikrum is max, threshold is energywise max
7
8
9  start = tic(); % Start measuring time
10 addpath('/media/cubist/files/all/matlabsaves/my_functions'); % Import
11 %% Parameters
12 % Detector Parameters
13 intr_row_size = 256;
14 intr_column_size = 256;
15 pixel_area = 0.0055*0.0055; % in cm2
16 clock_to_ns = 25; % length of a clock cycle in ns
17
18 % Multi Frame Options (for combining frames, automatically done with .pmf)
19 no_of_frames = 20;
20 row_size = intr_row_size * no_of_frames;
21 column_size = intr_column_size;
22 % Choose sub-detector resolution (Possible values: 1,2,4,8,16,...)
23 % Caution: Statistics will decrease by the same ratio
```

```
row_divide = 4; % Resolution in cm will be 1.41/row_divide
column_divide = 4; % Resolution in cm will be 1.41/column_divide

%ToA Format and Acquisiton Time used in Timepix3
ToA_format = 1; % 0 for clock cycle, 1 for ns
shutter_length = 250000; % in ns or clock cycle(25 MHz) depending on ToA format

%Options (Enabled: 1 , Disabled: 0)
timing_downsample_on = 0; % For low fluxes, no need of ns precision but longer shutter
obtain_clustering_statistics = false; % If statistics are desired
first_statistics_run = 0; % Resets to previous statistics to zero
statistics_to_file = 0; % Saves statistics to file in the given order

%Time Downsampling Parameters (How many times downsample)
downsample_ratio = 1;
%Declustering Parameters (Per unit time, be careful if time is downsampled)
neighbouring_degree = 3;
degree1_ToAdiff = 80;
degree2_ToAdiff = 80;
degree3_ToAdiff = 50;
%% Obtain Data From File with Given Name and Extension
tic
filename = '../simulation_output/flux1e7_alpha_250000nsshutter_20frame'; % Saved Name (also with Directory if desired)
file_ext = 'pmf'; % Saved Extension
ToT = dlmread(strcat(filename,'.',file_ext,'_ToT.',file_ext),' ');
ToT = fliplr(ToT);
ToA = dlmread(strcat(filename,'.',file_ext,'_ToA.',file_ext),' ');
ToA = fliplr(ToA);
%fToA = dlmread(strcat(filename,'.',file_ext,'_FTOA.',file_ext),' ');

%ToA = round(25*ToA - 1.56*fToA); % Do not care about sub-nanosecond resolution
ToA = round(ToA);
[a] = toc();
disp(['Reading Time : ', num2str(a)]);

%% Timing Downsample if thought to be necessary

if(timing_downsample_on)
    [ToA,shutter_length] = downsample(ToA,shutter_length,downsample_ratio);
end

%% Decluster
tic
[total_hit, ToT_center, degree1ToA_dist_p, degree2ToA_dist_p,...
    degree3ToA_dist_p,early_pixels] = decluster5(ToT,ToA,
        neighbouring_degree,...
    degree1_ToAdiff,degree2_ToAdiff,degree3_ToAdiff,shutter_length,...
    row_size, column_size,intr_column_size,intr_row_size,...
    obtain_clustering_statistics);
[a] = toc();
disp(['Decluster2 : ', num2str(a)]);

```

```
75
76 %% Sub-Detector Hit Count
77 tic
78 if(row_divide == 1 && column_divide == 1)
79     sub_hits = total_hit;
80     error_sub_hits = sqrt(total_hit);
81 else
82 [sub_hits, error_sub_hits] = sub_hit2(intr_row_size,intr_column_size,...
83     row_size,column_size,ToT_center,row_divide,column_divide,early_pixels);
84 end
85 [a] = toc();
86 disp(['Sub Hit count : ', num2str(a)]);
87
88 %% Find Measurement Time and Area, then uncorrected flux
89 tic
90 [flux_divider] = time_area_norm3(ToT,ToA,shutter_length,row_size,...
91 column_size, ToA_format, pixel_area, clock_to_ns,timing_downsample_on,...
92     downsample_ratio, row_divide, column_divide, intr_row_size,...
93     intr_column_size,early_pixels); %in cm2*s
94 [a] = toc();
95 disp(['Time_area_norm2 : ', num2str(a)]);
96 % sub_hits
97 % total_hit
98 %flux_divider
99 flux = sub_hits./flux_divider
100 error = error_sub_hits./flux_divider
101 %% Flux Correcter (which is not neeeded anymore)
102 %dead_time_area = sum(sum(ToT))*clock_to_ns*10^-9*pixel_area; % in s*cm2
103 %flux_corrected = flux_divider/(flux_divider-dead_time_area)*
    flux_uncorrected
104 %error_corrected = flux_divider/(flux_divider-dead_time_area)*
    error_uncorrected
105
106 %% Obtain Clustering Statistics for the given chip and particle
    configuration
107 if(obtain_clustering_statistics)
108     % 0) Enable initialization only in first run
109     if(first_statistics_run)
110         degree1ToA_dist = [];
111         degree2ToA_dist = [];
112         degree3ToA_dist = [];
113         non0ToT_1D = [];
114         non0ToTc_1D = [];
115         total_hit_combined = 0;
116     end
117     % 1) Occupancy and ToA
118     %Combine statistics from all data (use clear all to restart collecting)
119     total_hit_combined = total_hit_combined + total_hit;
120     degree1ToA_dist = [degree1ToA_dist,degree1ToA_dist_p];
121     degree2ToA_dist = [degree2ToA_dist,degree2ToA_dist_p];
122     degree3ToA_dist = [degree3ToA_dist,degree3ToA_dist_p];
123     %Occupancy ratio by degree
124     degree1_occupancy = length(degree1ToA_dist)/total_hit_combined;
125     degree2_occupancy = length(degree2ToA_dist)/total_hit_combined;
126     degree3_occupancy = length(degree3ToA_dist)/total_hit_combined;
127     occupancy_per_degree = [degree1_occupancy, degree2_occupancy, ...
```

```
128         degree3_occupancy];
129     %Number of occurances
130     if (degree1_occupancy > 0)
131         degree1ToA_prob = hist(degree1ToA_dist,(0:1:degree1_ToAdiff));
132     else
133         degree1ToA_prob = [];
134     end
135     if(degree2_occupancy > 0)
136         degree2ToA_prob = hist(degree2ToA_dist,(0:1:degree2_ToAdiff));
137     else
138         degree2ToA_prob = [];
139     end
140     if(degree3_occupancy > 0)
141         degree3ToA_prob = hist(degree3ToA_dist,(0:1:degree3_ToAdiff));
142     else
143         degree3ToA_prob = [];
144     end
145     %Occurances normalized to probability
146     degree1ToA_prob = degree1ToA_prob/sum(degree1ToA_prob);
147     degree2ToA_prob = degree2ToA_prob/sum(degree2ToA_prob);
148     degree3ToA_prob = degree3ToA_prob/sum(degree3ToA_prob);
149     % 2) ToT and center ToT
150     % Seperate diffused and cneter ToTs
151     non0ToT_1D_p = TwotoOneD(ToT(ToT_center == 0));
152     non0ToTc_1D_p = TwotoOneD(ToT(ToT_center > 0));
153     non0ToT_1D_p = non0ToT_1D_p(non0ToT_1D_p>0);
154     % Combine statistics from all data (use clear all to restart collecting
        )
155     non0ToT_1D = [non0ToT_1D, non0ToT_1D_p];
156     non0ToTc_1D = [non0ToTc_1D, non0ToTc_1D_p];
157     %Number of Occurances
158     ToT_prob = hist(non0ToT_1D,(1:1:max(non0ToT_1D)));
159     ToTc_prob = hist(non0ToTc_1D,(1:1:max(non0ToTc_1D)));
160     % Occurances normalized to probability
161     ToT_prob = ToT_prob/sum(ToT_prob);
162     ToTc_prob = ToTc_prob/sum(ToTc_prob);
163     % 3) Save clustering statistics to file for later use
164     if(statistics_to_file)
165         occupancy = [length(occupancy_per_degree),occupancy_per_degree];
166         degree1ToA = [length(degree1ToA_prob),degree1ToA_prob];
167         degree2ToA = [length(degree2ToA_prob),degree2ToA_prob];
168         degree3ToA = [length(degree3ToA_prob),degree3ToA_prob];
169         ToT_probf = [length(ToT_prob),ToT_prob];
170         ToTc_probf = [length(ToTc_prob),ToTc_prob];
171         % Determine filename
172         write_to = '../statistics_data/alpha_notall.txt';
173         % Size of occupancy + Occupancy from degree 1 to 3
174         dlmwrite(write_to,occupancy,'delimiter',',');
175         % Size of 1st degree ToA prob dist + 1st degree ToA probabilities
176         dlmwrite(write_to,degree1ToA,'-append','delimiter',',');
177         % Size of 2nd degree ToA prob dist + 2nd degree ToA probabilities
178         dlmwrite(write_to,degree2ToA,'-append','delimiter',',');
179         % Size of 3rd degree ToA prob dist + 3rd degree ToA probabilities
180         dlmwrite(write_to,degree3ToA,'-append','delimiter',',');
181         % Size of diffusion ToTs + diffusion ToT probabilities
182         dlmwrite(write_to,ToT_probf,'-append','delimiter',',');
```

```
183             % Size of center ToTs + center ToT probabilities
184             dlmwrite(write_to,ToTc_probf,'-append','delimiter',',');
185         end
186     end
187
188     toc(start) % Measure elaspsed time
```

Now I need to define them and talk about their interrelations.

ToA : Time of each event recorded per pixel, if multiple events happen during the measurement time at the same pixel, only the first is recorded.

ToT : Time spent over threshold by each pixel, this has a strong correlation with energy deposition at the pixels. Similar ToA only the first event during the shutter length is recorded. Notice the 'fliplr' function used both for 'ToA' and 'ToT', this is to flip only the columns of the matrix around the center (1 becomes 256, 2 becomes 255 and so on.) I found this required, because when you use Tpx3 as we do, connector looking up, detector looking down, the right side of the detector gives the first (leftest) value in the matrix. I checked this by putting an alpha source to the corners of Tpx3 and which side of the matrix detects something. I described this in detail, because if you see a weird jump in fluxes, this might be because I miscalibrated this part, and you may need to correct me.

no of frames : Number of measurements Timepix will perform each with the duration equal to the shutter length. This will increase your statistics proportionally (decrease statistical uncertainty) but will also increase the measurement time (you wait more, chip gets more radiation). When you make a measurement each frame is recorded below the first one. Timepix3 has 256 x 256 pixels, so if you make 10 measurements, your raw data matrix would be 2560 x 256. Therefore, your data file is larger if you make more measurements, you transfer more information, hence the longer time.

row and column divide : The sensitive area of Timepix is 1.41 cm x 1.41 cm, you can divide its rows and columns to smaller parts (powers of 2, preferably up to 16). If you do this your resolution may increase up to 0.09 cm x 0.09 cm, which should be more than enough for our application. The catch is though, you reduce the statistics (of course assuming 'ceteris paribus') by the square root of your division, so if you divide detector to 16 x 16 parts, you will have 16 times more statistical uncertainty. Use it wisely.

shutter length : This is the measurement time, as long as 'ToA format' is 1 (which should be the case, unless you mess with the python script) it is in nanoseconds. This is a weird parameter, 2500 ns is usually optimum 1×10^9 protons/cm^2/s or higher fluxes. At lower fluxes it will take longer time to fill the detector, hence it is advised to increase this parameter inversely proportional to flux. Especially, if you are measuring a regular alpha sources (kBq-Mbq range), you will not detect much with low shutter lengths, you might want to increase it up to 1 ms or even more. Of course, while you are making your first measurements of proton beam, you will not know the flux, so I suggest you use 2500 ns, and increase it slowly if you detect fluxes in the range of 1×10^9 protons/cm^2/s to 1×10^6 protons/cm^2/s.

filename : The name of the file to be analyzed. LabVIEW automatically handles this, but if you want to analyze by only using MATLAB (say you reanalyze an old raw data), you need to give both filename and relative directory (or a complete path).

neighbouring degree : When a particle hits the detector, it generally creates multiple signals in the pixel it hit and the neihbouring pixels. These signals are resolved by the use of 'decluster5' (line 67) function. The neighbouring degree decides, how many pixels next to the first arrived pixel will be checked for this effect. It is OK to decrease it, but if you ever need to increase it, you will also add a little code to the 'decluster5' function.

degree# ToA diff : These parameters are there to decide how many nanoseconds after the first signal, secondary signals (due to clustering) should be checked and eliminated. These numbers certainly work for my simulated data using alpha particle parameters, and theoretically they should be more than enough for protons with higher energy, since they deposit much less energy. There I do not see

any reason to increase them, but I also do not know how much you can decrease them for protons. I suggest you to keep them as it is (that is for a real measurement, you are of course free to play with them for understanding their effects), unless a change is absolutely necessary (as in you are sure you are measuring flux wrong and you checked everywhere else).

statistics commands : These are strictly related to obtaining statistical data for a simulation. If you only want to make a measurement do not worry about this. I used this to obtain how signals of alpha particles distribute to pixels (due to clustering problem). You might only want to use it after you get some real but sparse (low flux) proton data, and you want to use that data to simulate the behavior of detector at different fluxes. If you want to continue doing simulations with alpha data you do not need to that, just find my simulation code. I suggest you contact with me before trying anything about this part.

downsample commands : If you make a reeeealllyy long measurement, but you have very sparse data, you might want to downsample your data, so that analysis will speed up. Well, at least I wrote that with this purpose, but since then I updated the code to be more efficient, so it will not bother with time intervals where no particles arrive, therefore I expect no real improvement when you use downsampling. I just like the word (it reminds me my DSP course), so I keep it there.

3 Diamond

3.1 Setting Up Measurement

1. Make sure ADC, its FPGA, Diamond Detector, and the SBC is correctly positioned in the target area. ADC should be connected to its FPGA readout module, if not connect them appropriately.

2. Connect bias, signal and power cables (Use wiring diagram to connect the proper cables) to the 'Broadband Amplifier' of the Diamond Detector. I assumed Diamond Detector is already connected to its amplifier and ready for use in its shielding.

3. Set bias to 400 V, connect the other end of signal cable to the input 1 or A of the ADC, set power to 12 V.

4. Connect ADC to USB 2.0 port of the SBC, and FPGA to USB 3.0 port of SBC with a USB 3.0 cable. In addition connect SBC to the Ethernet switch in the target room, and do not forget to connect power to the switch and the SBC too.

5. After ensuring the proper cooling of ADC, FPGA and the SBC turn the power on.

6. Connect to the SBC from the control computer via remote desktop application of Windows.

7. There is a manual written by Serdar Aydin in Turkish on the Desktop, if you cannot read it, help someone translate it for you. I followed a manual in German once, you understand most of it by just checking a few words out from your favorite translator, so do not worry if you cannot find anyone to translate it for you. Some parts of it, like connecting power to the ADC you already did, hence you can skip. In most cases you do not need to hard reset the ADC (do not need to push the reset button physically). If it does not work once without the hard reset, you can try again with it one more time. Do the rest of the manual.

8. At the end manual says you to change capture options, the capture length (aka. how many samples you want to take). Until $4-5\times10^9$ protons/cm^2/s flux 128000 samples (it corrects that number to the closest appropriate one, so do not worry if it is not exactly what I say) should be enough. Increasing it is necessary for higher fluxes (up to 1000000 samples for $4-5\times10^10$ protons/cm^2/s, I advise you too check Poisson formula, and find out the time required for the signal to hit zero with a certain confidence interval, I have matlab codes for them, and I gave them to Serdar Aydin), and would not hurt for lower fluxes but the downside is of course the measurement time. If you ever decide to increase it, you would need to make a simple change in the LabVIEW server (in the SBC Desktop, 'diamond readout' folder). You need to open the terminal of the program, find and

open the subvi responsible from the measurement itself. At its terminal, on far right there is a pause (wait, delay) term, you may need to increase it up to 10000 ms. This issue is caused because LabVIEW says measurement is complete after it starts writing to the file, but not after finishing it. If you increase the data in the file a lot, then it says file is ready for transfer to the control computer before finishing saving the data. The delay is therefore necessary, and unfortunately the amount of it is experimental (Run the code and watch the size of the file grow, measure time until it stops). You may want to contact me or Serdar Aydin, for more information or any help.

9. Try to make a measurement from HSDC Pro to ensure that connection to ADC is OK. If not check or repeat the above procedures

10. Start the server program I mentioned previously, take note of the 'Port No' since you will need to use the same number in the LabVIEW script at the control computer

11. Find the LabVIEW program for controlling diamond ('diamond readout' in control computer desktop, 'labview' and then 'current'), open it

12. Make sure diamond will not hit anything during its scan (you might need to perform similar actions) In terms of actions (but not values given) the steps (of the manual) are very similar to steps (of the manual) of Tpx3 detector between 15 and 20 inclusive on both ends (You notice the layout of the control screen is same as the Timepix3). The XY stage of Diamond Detector is different from the Tpx3's, and controlled with Arduino. You need not to worry about heating of the Diamond very much, but if a PT100 is inserted around, you should still the check its temperature once in awhile, especially when the beam is on. The analysis time for diamond is lower than second but plotting certain graphs can slow the program a little, there is a boolean control option for that (but seeing the graphs are super cool). The measurement time takes a long time a minimum of 14 seconds per position, and increases by the amount of delay you added previously). These 14 seconds are due to the speed of HSDC Pro to make a measurement and so, we could not manage to speed it up, but I would be glad if you do.

13. Diamond Data is analog on the essence (yes, you digitize it with an ADC but it is not like Tpx3 giving you definite digital numbers). Therefore the analysis of it requires careful calibration. The general idea is you need to find the average area under a single pulse due to a proton, and enter that value to calibrate your detector. For that you may want to scan edges of the beamline, where the proton rate is expected to be lower, so you can catch protons one by one. For calibration both one the right and left pane there are extra TABs in the LabVIEW. The left one enables you to enter calibration options and inputs, while the left one shows you the output histograms, so that you can decide the calibration inputs correctly.

3.2 Analysis Parameters of Diamond Data

I will again describe the analysis principles using the MATLAB Script given in Script 2. All fields required modification should be reachable from the command screen in LabVIEW, generally in the Calibrate Diamond TAB of the left pane.

The main problem with the Diamond Detector is the pile-up of pulses due to arrival of protons within very short intervals (pulse duration is around 10 ns, so if protons arrive faster pile-up happens). If a simple trigger and count pulses method is used, then pile-up to certain amount is mathematically correctable (look at my presentations to see how). However, in our case the piled-up pulses can reach 5-6 pulses on top of each other, in this case you will not even see edges of the pulses to trigger. Therefore an alternate approach was utilized, which is to simply divide the whole area (Average signal height times signal time) by the area of a single pulse. One problem encountered is that 'Broadband Amplifier' is removing the DC average of the signal, while removing 400 V DC bias with a decoupling capacitor. Since previous to subtraction of DC average, the minimum value attainable is zero, now the minimum value attainable is the DC average. Therefore if we record a data long enough, the minimum (actually maximum since the pulse is negative) value obtained would be the DC average of the signal, which could then be used to find total area and hence the proton flux.

However this method has its own problems. The detector and the ADC has a constant noise that would cause an uncertainty in the order of 10^6 protons/cm^2/s flux. At lower fluxes ($< 1 \times 10^8$ protons/cm^2/s) this would mean a large percentage uncertainty. However, at these fluxes the pile-up is manageable, hence an edge triggered readout with pile-up correction is possible, in which case this noise would not be included. Furthermore for higher fluxes it is possible to use secondary port of ADC as a noise sampler, and subtract the flux corresponding to that noise from the flux measured in the primary port.

If required, the script could automatically decide the measurement method for most accurate measurement. The parameters of both measurements and the automation will be described over the Script 2 below.

Diamond Analysis Main Script (Script 2)

```
%% License and Manual
% Authors: Baran Bodur and Serdar Aydin
% A script to evaluate diamond detector measurements obtained with a
% 1 GHz ADC and find out flux even in high pile-up situations

addpath('C:\Users\ODTU SDH Kontrol\Desktop\diamond_readout\elmas_kod_veri')
%% Parameters
% ADC Parameters
ADC_sample_step = 1; % ns
ADC_bits = 16;
ADC_max_code = 2^15 - 1;
ADC_min_code = -2^15 - 1;
ADC_max_V = 0.95;
ADC_min_V = -0.95;

% Detector Parameters
area = 0.15 * 0.15 * pi; % in cm2,  Circular with 0.15 cm radius

% Options
measurement_method = 0; % 0: automatic, 1: count pulses, 2: integral/pulse
    area

% High Flux Measurement Parameters
threshold = 1000; % in ADC codes
max_min_dif_threshold = 1.5; % difference between max and min
measurement_method = 2; % 0: automatic, 1: count pulses, 2: integral/pulse
    area
threshold = 1000; % in ADC codes
number_of_maxima_tocheck = 1;
ave_pulse_height = 2000;
ave_pulse_length = 11; % in ns
ave_pulse_area = 11*2000; % (Code * ns)



%% Obtain Data

filename = 'SpeedTest2.csv';

ADC_data_all = dlmread(filename);
ADC_data = ADC_data_all(:,1);

```

```
% Data Parameters
mean_data = mean(ADC_data)
length_data = length(ADC_data)
max_data = max(ADC_data)
min_data = min(ADC_data)
max_min_difference = abs(max_data-min_data)/ave_pulse_height
if(measurement_method== 0)
    if(max_min_difference < max_min_dif_threshold)
        measurement_method = 1;
    else
        measurement_method = 2;
    end

end

% ADC Data in Voltage
%ADC_V_data =


%% Measure Flux in Situations with Pile-Up

        sorted_data = sort(ADC_data);
        maxima_array = sorted_data((end-number_of_maxima_tocheck):end);
        maxima_average = mean(maxima_array); % Average maxima of
        integral_through_DC_ave = maxima_average*length_data; % ns*Code
        ADC_data_with_DC_ave = ADC_data - maxima_average; % Data with DC
        integral_through_sum = -1*sum(ADC_data_with_DC_ave); % ns*Code
        count_through_DC_ave = integral_through_DC_ave/ave_pulse_area;
        count_through_sum = integral_through_sum/ave_pulse_area;
        flux_through_DC_ave = count_through_DC_ave/(area*length_data*10^-9)
        flux_through_sum = count_through_DC_ave/(area*length_data*10^-9);
        staterror_through_DC_ave = sqrt(count_through_DC_ave)/(area*
            length_data*10^-9)
        staterror_through_sum = sqrt(count_through_sum)/(area*length_data
            *10^-9);
%% Measure Flux in situations without too much pile-up
        % Initializations
        pulse_length = []; %Records length of pulses
        pulse_area = []; % Records area of pulses
        pulse_height = []; % Records
        temp_pulse_length = 0;
        temp_pulse_area = 0;
        signal_started = 0; %flag
        % Pulse counter and pulse property recorder
        for i = 3:1:length_data
            if(ADC_data(i) < (mean_data - threshold))
                signal_started = 1;
                temp_pulse_length = temp_pulse_length + 1;
                temp_pulse_area = temp_pulse_area + (ADC_data(i) -
                    mean_data);
                pulse_height = [pulse_height, (max_data - ADC_data(i))];
            elseif(signal_started)
                pulse_length = [pulse_length, temp_pulse_length];
                pulse_area = [pulse_area, temp_pulse_area];
                temp_pulse_length = 0;
                temp_pulse_area = 0;
```

```
94                  signal_started = 0;
95              end
96          end
97          pulse_count = length(pulse_length)
98          if(pulse_count == 0)
99              pulse_area_occurances = 0;
100             pulse_area_index = 0;
101             mean_pulse_area = 0;
102             pulse_length_occurances = 0;
103             pulse_length_index = 0;
104             mean_pulse_length = 0;
105             pulse_height_occurances = 0;
106             pulse_height_index = 0;
107             mean_pulse_height = 0;
108             flux_through_count = 0
109             staterror_through_count = 0
110
111         else
112             %Pulse area
113             pulse_area = -1 * pulse_area;
114             pulse_area_index = 0:100:100000;
115             pulse_arae_occurances = hist(pulse_area,pulse_area_index);
116             mean_pulse_area = mean(pulse_area)
117             % Pulse length
118             pulse_length_index = 0:1:100;
119             mean_pulse_length = mean(pulse_length)
120             pulse_length_occurances = hist(pulse_length,pulse_length_index)
                    ;
121             % Pulse height
122             pulse_height_index = 0:100:30000;
123             mean_pulse_height = mean(pulse_height)
124             pulse_height_occurances = hist(pulse_height,pulse_height_index)
                    ;
125             % A simple correction valid for low pile-up
126             syms x;
127             rate_corrected = double(vpasolve((pulse_count/length_data) ==
                    ...
128                 x*exp(-1*x*mean_pulse_length),x));
129
130             flux_through_count = rate*10^9/area
131             staterror_through_count = sqrt(rate*length_data)*10^9/(
                    length_data*area)
132         end
133         if(measurement_method == 2)
134             flux = flux_through_DC_ave
135             error = staterror_through_DC_ave
136         elseif(measurement_method == 1)
137             flux = flux_through_count
138             error = staterror_through_count
139         end
```

ADC Parameters : These are the parameters of the ADC we are currently using, if you want to be sure check them from its datasheet. If they are correct you do not need to play with them.

Threshold : If you are using automatic or pulse count mode for measuring flux, then you will need to set a threshold for pulse detection. Since our pulses are negative, this threshold is actually set as -1000 ADC codes. Beware, it is not volts or millivolts but ADC code, hence you will need

to manually convert your designated threshold in millivolts to the ADC codes. It is a simple proportion problem, if 2^{15} corresponds to 0.95 V, then what does 60 mV correspond...

max min diff threshold : If you use the automatic mode, so that program automatically decides using pulse counting or DC averaging, then this is how it decided which one to use. The number you put here will be interpreted in terms of average signal height (will be defined below). If in your signal, there is a difference in value more than average signal height times the value of this threshold, the program will automatically classify this as a high pile-up situation and uses DC averaging, otherwise it will use simple pulse counting. I recommend a value between 1.5 and 3, more than there will give you wrong results that I am sure.

number of maxima to check : This is a countermeasure to reduce the noise. If you check only one maximum in area detection method to find the DC average of the signal, that signal maximum might be further enhanced by the noise. Hence you count a higher flux than normal. If you increase this number slightly, it will average multiple maxima hence the effect of noise will be reduced. Of course increasing it too much is not advised, because then you might add more maxima than there are in your short signal. The correct way to assign it, is checking how many times you expect that your signal hits the maximum(look at Poisson process, I have a simple code about it in my matlabsaves/diamond which I gave to the group). ADC is fast so it can take two or even three sample around each maximum, then you might want to set it at most three times of your expected maxima. If you apply some kind of noise reduction, or you measure high enough fluxes so that noise will only cause a slight uncertainty then you can keep it simple as 1.

average pulse height : Average pulse height you obtained by counting pulses from a previous measurement. You will need to do this again for each kind of particle you measure at each energy, since the shape of pulses will be different. This parameter is required if you will use the automatic or DC averaging method to measure flux, since then you will need to normalize your results with a single pulse area. Pulse height is also necessary as the determining property of automatic measurement. You will determine this and the other two parameters below by using the histograms in the calibration TAB of the LabVIEW. Look at the histograms, save them if you wish, and determine a reasonable average.

average pulse length : Average pulse length you obtained by counting pulses from a previous measurement.

average pulse area : Average pulse area you obtained by counting pulses from a previous measurement. Very important, since you will use it, when obtaining flux from the DC average.

filename : Name of the raw ADC file, LabVIEW automatically handles it, when you use it in there.

By the way, if you ever want to make a slight change in these code, do not forget that you need to copy the changed versions back to LabVIEW. When you do that, if any of the inputs are given by the LabVIEW, then you will need to comment them in your MATLAB code (the version which you copied to the LABVIEW).

4 Epilogue

There is probably a lot I missed in this manual, and a lot I could not explain well enough so that you can understand. You may want to check datasheets of the used equipment, my other reports and presentations for better understanding. As I said, I would like to hear from you if you have any questions (I would like to hear even if you do not have any questions, if you have any progress).

I wish you the best of everything...

Baran