



A Scanning Tool Program for FLASH

2015 DESY summer student, FLASH

Yifan Li, Tsinghua University

Supervisor: Stefan Dusterer

September 9, 2015

Abstract

A scanning tool program for the Free-Electron Laser FLASH with graphical user interface (GUI) is completed using Matlab tools. It is used to change the controlled value of a FLASH instrument, observe the measured value and plot results after processing. With the scanning tool, users can easily scan parameters of a specific instrument and get one DOOCS (FLASH control system) property versus another property. It is helpful and the GUI makes the program more easy to use. In this report, the method of realization is introduced in detail, a simple application example is given and conclusions are presented.

Contents

1	Introduction	3
1.1	A scanning tool	3
1.2	Program structure	4
2	Methodology	5
2.1	Introduction to MATLAB GUIDE	5
2.2	Realization of basic function	6
2.3	Realization of other function	7
2.3.1	Pause and continue	7
2.3.2	Save	8
2.3.3	Analyze	8
2.4	Solutions to some typical problems	9
2.4.1	Read back value	9
2.4.2	Value addresses	9
2.4.3	Network interruption	10
2.4.4	Device does not respond	10
3	A simple applicaion example	10
3.1	Application example	10
3.2	Results and Analysis	10
4	Summary	12
5	Acknowledgement	12

1 Introduction

1.1 A scanning tool

A scanning tool program with graphical user interface (GUI) was finished using MATLAB tools. It is used for scanning parameters of an instrument.

The program is useful for scientific research in some cases. For example, researchers need to get X-rays with specific energy or laser with specific time structure. It would take a large effort for researchers to manually change parameters and measure the output for many times in order to find the most proper way to obtain lasing. This scanning tool is programmed to solve this problem by doing this kind of repetitive work automatically instead of manually. It is helpful and the GUI makes the program more easy to use.

A GUI is a graphical display in one or more windows containing controls, called components, enable a user to perform interactive tasks. The user does not have to create a script or type commands at the command line to accomplish the tasks [2].

The GUI of scanning program is shown in Figure 1.

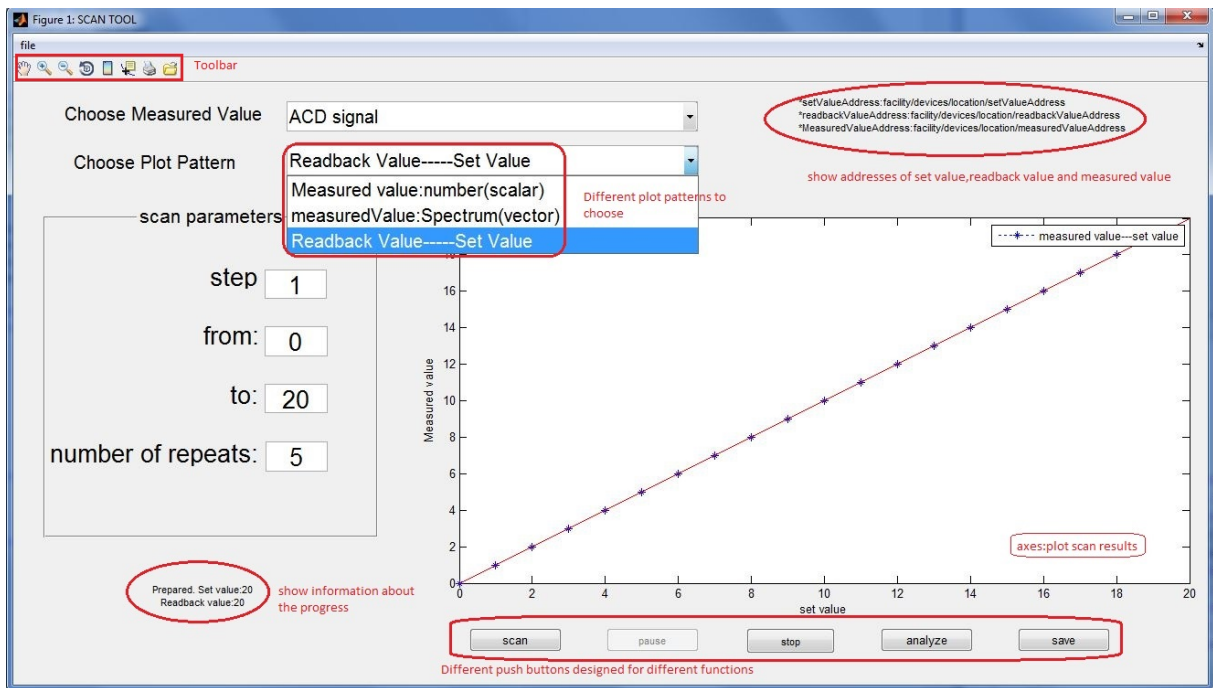


Figure 1: Graphical user interface (GUI)

- Users communicate directly with the GUI. The program allows users to type in essential parameters (like step size or controlled value range) and choose what they

want to measure by using the pop-up-menu. The measured value could be scalar or vector, users can choose proper pattern to display results on the GUI (see more at section 2.2).

- Once the scan button is clicked, the program starts to work. The scanning progress starts at minimum controlled value that users set, grows with step by step and finishes at the maximum value. And the results are displayed at axes on GUI.
- The scan program can also go backwards, with starting value larger than finishing value and a negative step length.
- Note that if the step attribute is omitted, the step size defaults to 1.
- Some buttons are put on the GUI. Users can use “pause/continue” button to control the progress, and use “stop” button to stop scanning.
- The results are not only presented in forms of graphs. Users can also get a text file of data. A text file would be saved every time when a user scans, by popup of a dialog window asking for file name.
- Analyze: calculate a fitted curve with errorbar to show tendency.

1.2 Program structure

The association diagram can be seen in Fig 2. The whole project includes 3 sections: GUI, Core Program and Adapter.

A user communicates with the GUI. A user could type in scanning parameters and click a button to command the program to execute instructions.

The core program, which is actually an m-file in MATLAB, is written as the callback function of “scan” button of the GUI. It gains information from the GUI and also passes essential data to the device. The information contains: which device or which motor a user chooses; what the controlled value is (one device may have more than one controlled value); data such as step size and controlled value range (minimum and maximum value). The information transfer is completed through the adapter.

An adapter is a special script for connecting the program with some specific instruments. There are two kinds of adapters. The Fake Adapter is used to link the program with a virtual device. The JDoocsAdapter can enable the program to get access to the FLASH control system DOOCS so that the program can execute instructions. The two adapters are independent of each other and the switch between them is free without changing the code.

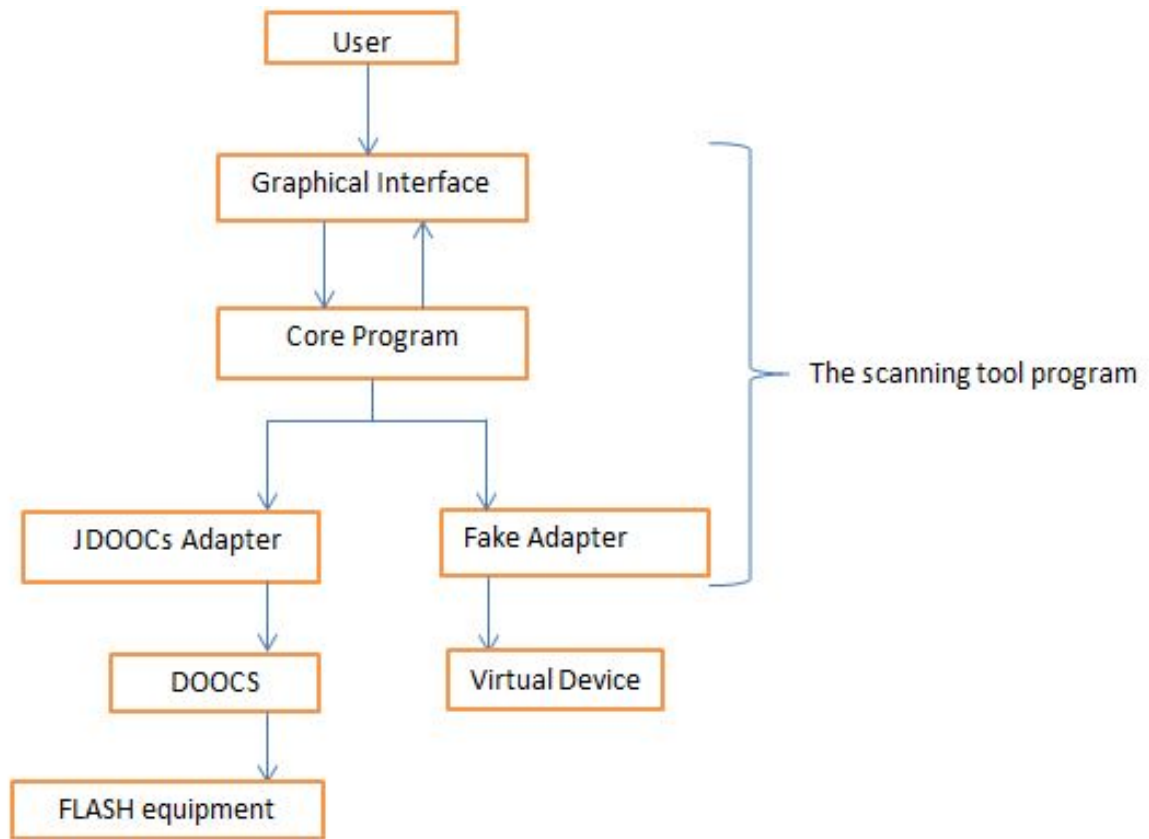


Figure 2: Program association diagram

Once a user commands to start scanning, the core program sets the parameter of the device and asks the adapter for a return of the output value, which is called measured value in this case. After some simple processing, the results are displayed on the GUI to show to users.

2 Methodology

2.1 Introduction to MATLAB GUIDE

In this program, the author chose to create the GUI using GUIDE. This approach starts with a figure that you populate with components from within a graphic layout editor. GUIDE creates an associated code file containing callbacks for the GUI and its components. GUIDE saves both the figure (as a FIG-file) and the code file [1].

There are 5 buttons, 1 axes, 2 pop-up menus and a panel with several edits on the GUI. Each of these components has a callback function. The GUI waits for a user to manipulate a control, and then respond to each user action in turn. A particular user

action, such as pressing a screen button, or passing the cursor over a component, triggers the execution of each callback. By writing callbacks that define what the components do to handle events, we can realize different functions.

2.2 Realization of basic function

The core program is put into the callback function of the button scan, used to realize the most fundamental function: scanning. Once the scan button is clicked, the scanning progress starts, from starting controlled value that users set, changing by step-length and finishing at the stop value. And the results are displayed at axes on the GUI.

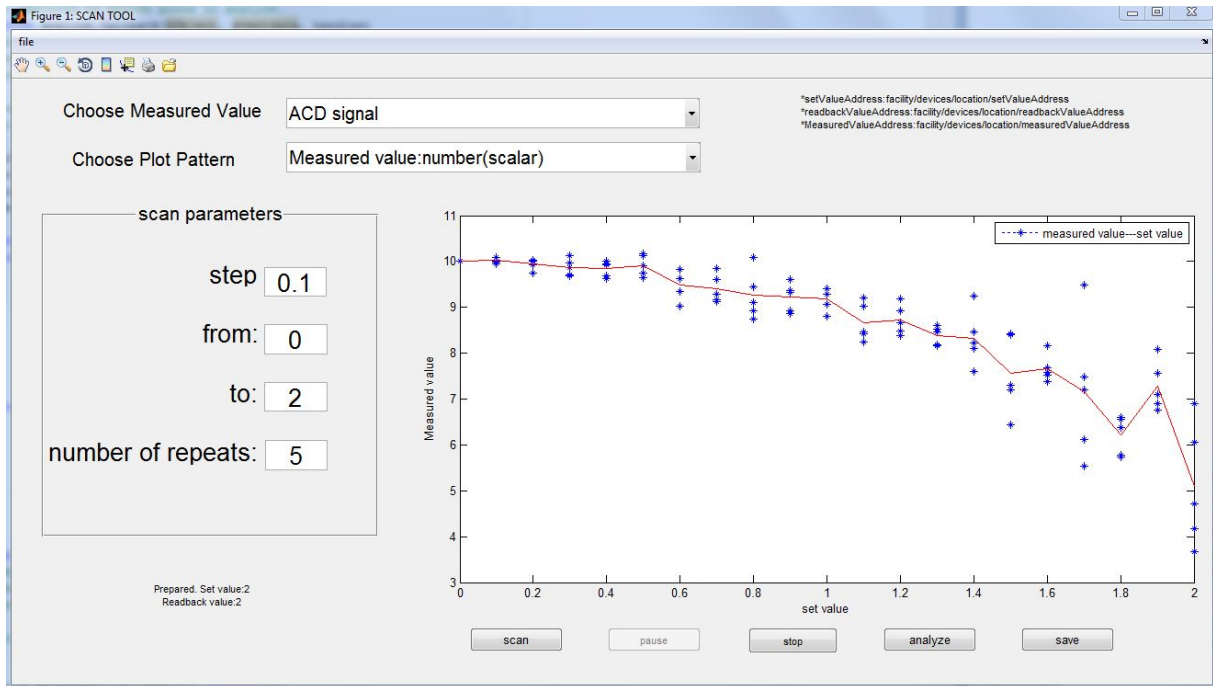


Figure 3: Program association diagram

Since there are different types of measured values, multiple plot patterns are designed. Fig. 3 shows the interface when the scalar (number) pattern is chosen. Under this pattern, the measured value is simply a number, which is collected by the program as y-coordinate of a point. The program plots corresponding points on the axes.

Fig. 4 shows the interface when the spectrum (vector) pattern is chosen. Users choose this pattern when the measured value is a vector, typically a spectrum. After the scanning is finished, some important results containing meaningful information are displayed in the picture form.

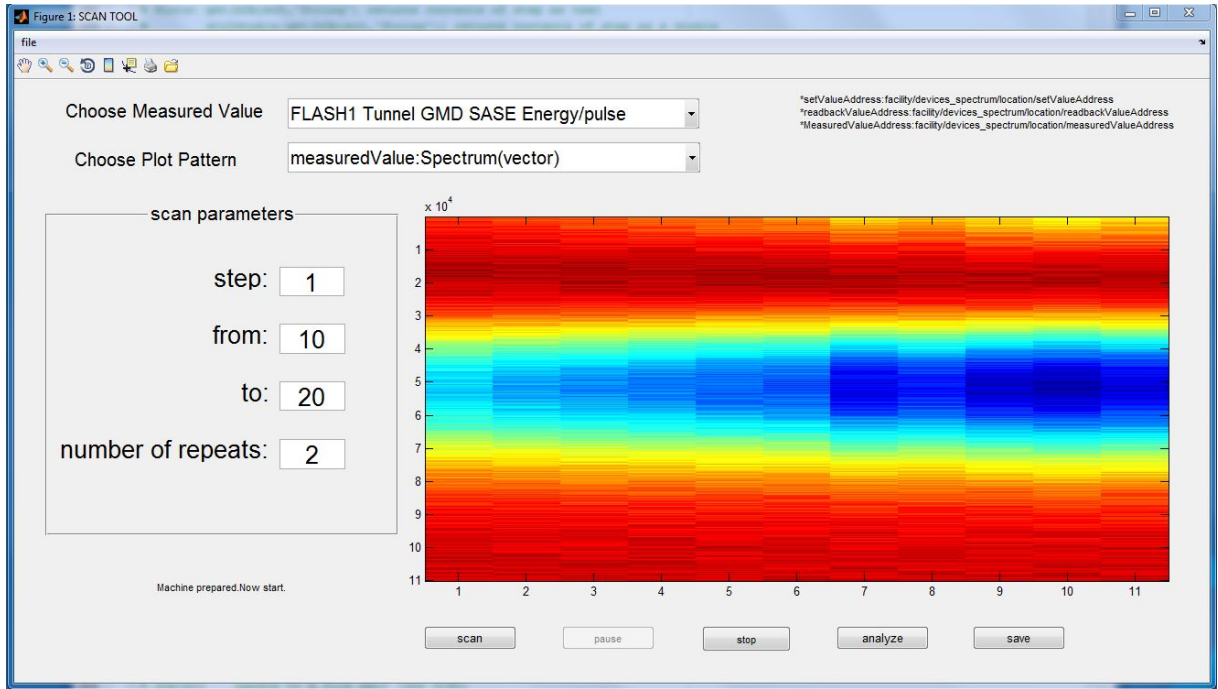


Figure 4: Program association diagram

2.3 Realization of other function

There are some other buttons on the GUI for executing different instructions. For example, the pause/continue button allows users to pause the action, and the stop button is used to stop the whole progress. These functions are realized in a modularize way. Later, when people in the FLASH group need to modify some function, they just have to change one part rather than go across the whole program.

2.3.1 Pause and continue

In order to make sure users can control the progress, a pause/continue button is designed.

To indicate the pause state, we can assign a variable named “pause.state”. The author uses 1 to represent operating state, while 0 represents suspended state. The GUI contains interdependent controls, menus, and graphics objects. Since each callback function has its own scope, a problem to solve is how to share data with those parts of GUI that need access it. The solution chosen is to associate data with a specific component using the `setappdata` function. You can access the data from other functions using the `getappdata` function. The code is shown below:

```
function pause_Callback(hObject, eventdata, handles)
% hObject    handle to pause
```

```

% handles      structure with handles and user data
pause_state=getappdata(gcf,'pause_state');
switch pause_state
    case 0
        setappdata(gcf,'pause_state',1)
        set(handles.pause,'string','continue')
    case 1
        setappdata(gcf,'pause_state',0)
        set(handles.pause,'string','pause')
end

```

Pressing the pause button triggers a change of the variable “pause_state”. In the core program “scan”, the variable is checked frequently, once it is found to be 1, the program would be suspended. The stop button is programmed similarly. Pressing the stop button triggers the change of “stop_state” variable, making further effect to stop the device at current set value.

2.3.2 Save

Every time a user starts a new scan, the program would save a text file automatically, and pop up a dialog asking for a file name. It is easily realized using `uiputfile` function. The code is shown below:

```

[filename,pathname,fileindex]=uiputfile({'*.txt'; '*.m'}, 'save data as');
file=strcat(pathname,filename);
dlmwrite(file, 'Scan data', 'delimiter', '');

```

2.3.3 Analyze

The analysis function does make results more visible. For the scalar pattern, a fitted curve with errorbar is plotted to show a tendency. For the spectrum pattern, an image is plotted using `image` function.

The data used for the analysis is the set value and the corresponding measured scalar or vector. These values are put in a matrix respectively, and the matrixes are stored at handles, which is a struct array containing all the GUI components. Therefore, the matrixes are easy to retrieve. The code is shown below:

```

function analyze_Callback(hObject, eventdata, handles)
s=str2double(get(handles.step,'string'));
a=str2double(get(handles.from,'string'));
b=str2double(get(handles.to,'string'));
numberOfRepeats=str2double(get(handles.numberOfRepeats,'string'));
switch get(handles.popupmenu1,'Value')

```



```

    case 1
        x=a:s:b;
        y=sum(handles.M')/numberOfRepeats;
        e=std(handles.M');
        errorbar(x,y,e,'b','LineWidth',0.8);
    case 2
        image(handles.M','CDataMapping','scaled');
end

```

However, the analysis is quite simple and crude, which might require improvement for further use.

2.4 Solutions to some typical problems

2.4.1 Read back value

After users set a value for the device, it takes time for motor to drive the device to the set position. Since that time depends on various factors, it is unpredictable. To solve the problem, the program observes another value, the read back value, and compares it with the set value. Before the program executes next action, the read back value would increase (or decrease) until it equals the set value. It makes sure every measurement is done at right time and right position. And there are also literal indications showing the progress on the interface. So before a new scan starts, it takes a while to wait for the device to be driven to the starting value. And before a new measured value is observed, the program does nothing until the read back value equals the set value. A tolerance within 0.001 is allowed.

2.4.2 Value addresses

It is of great significance to clarify value address when the program is used in FLASH system.

As is indicated before, the program must get access to the FLASH control system DOOCS in order to control equipment of the FLASH group. When the program set a controlled value, it actually writes a DOOCS property. When the program read a measured value, it actually reads from a DOOCS property. The JDOOCS-Adapter enables the program to get access to the FLASH control system DOOCS by transmitting addresses of the DOOCS property. The format of a DOOCS address is “facility/device/location/property”. In this way, the JDOOCS-Adapter isolates the scanning tool program from dealing with details of control system. Therefore it is necessary to clarify the addresses of the property, which contain important information about the device and property. The addresses are displayed on GUI to users.

In order to get access to DOOCS, the program also need realize java-MATLAB-DOOCS communication. How to read and write DOOCS properties with MATLAB have been covered in details on a DESY website. If you want to know more about that, please go to: <http://hasfweb.desy.de/bin/view/Setup/MatlabDOOCS>.

2.4.3 Network interruption

Sometimes the server could be unavailable. If a network interrupt like that happens, the program should identify it, give users a hint and try again to get access to the network. That is realized using try and catch statement.

2.4.4 Device does not respond

There is a situation that the device does not respond to the program and it could occur sometimes. Different from the server unavailable, this is caused by some improper actions or other on various factors. A temporary solution is to observe the read back value. If the program has sent a set value to the device, but the read back value does not equal the set value and does not change towards the set value, the program could tell that the device does not respond. There would be a hint on the interface informing users of the situation and suggesting users to try to scan again. However, the solution is not very friendly and should be modified. An expected way is to try to awaken the device for several times again automatically, and give a hint only after all efforts fail.

3 A simple applicaion example

3.1 Application example

We test the scanning tool program in Terahertz streaking experiment set-up. The experiment platform structure is shown in Fig. 5 [3].

In this example we have sampled a trace of a photodiode, which measured the HYDRA pulse scattering light at one sampling point (1MHz ADC). The scan plotted the photo diode ADC value versus the temporal position of delay stage in the THz setup. When the program is scanning, one mirror in set delay is moving slowly, driven by an electric motor. Change of the temporal position of the mirror means change of set delay. That means the set value is delay time and the measured value is HYDRA pulse scattering light.

3.2 Results and Analysis

Obviously it shows us the relationship between ADC signal and delay time. As is shown in Figure 6, at the sampling point, the HYDRA pulse scattering light reaches its bottom when the delay is 6 ps.

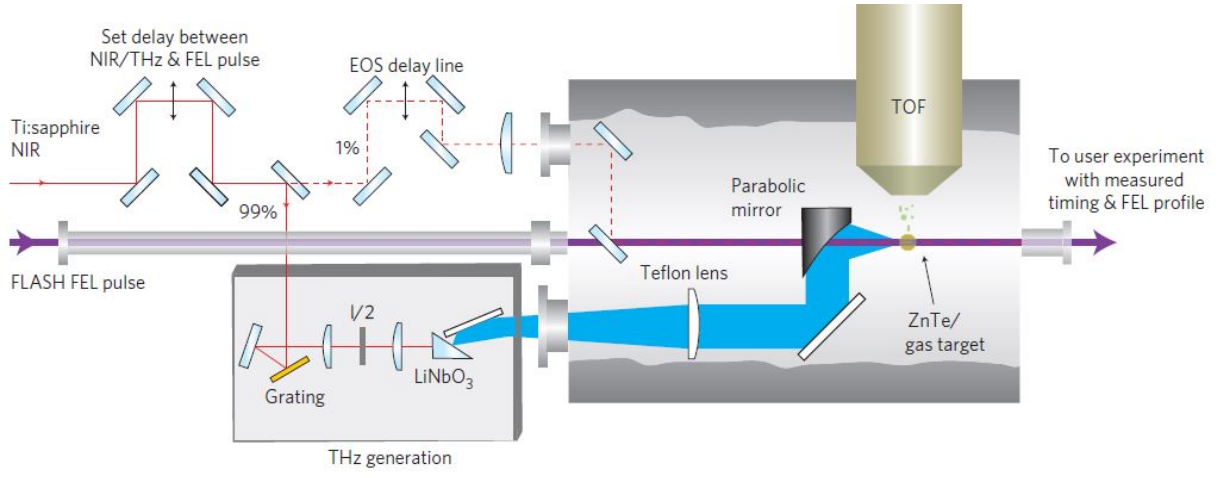


Figure 5: Terahertz streaking experiment set-up [3]

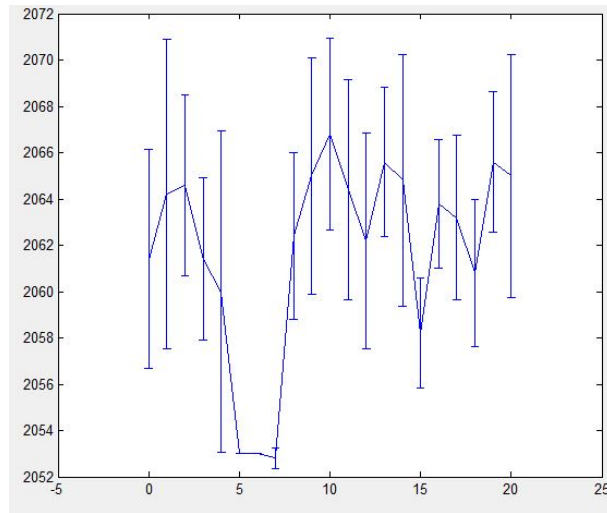


Figure 6: A photo diode ADC value versus set delay

4 Summary

This scanning tool program is a project that enables users to explore the output signals of a device under a certain range of inputs.

The GUI is accomplished by MATLAB GUIDE tools. MATLAB is a high-level language and GUIDE is an interactive environment for visualization and programming, which is a proper way to do this kind of programming. The scanning tool program is modular. The GUI, the main function and other functions are separate so that users can change any part without affecting others.

There are some limitations about the program. The interface is not so artistic, which can decrease the user-friendliness. However, it is not fundamentally flawed. And the analysis function is crude, for some further application, the author suggests it should be improved. Moreover, the stop button on the GUI is used to stop the scanning program rather than the hardware or the device. When extreme emergency happens, it is not powerful enough to brake the machine immediately. It may be safer to add a button which can cut off the power of the tested machine.

5 Acknowledgement

The author acknowledges the guidance and attendance of supervisor Dr. Stefan Dsterer. And I am also grateful to Erland Mller for his great support in this summer. The author also thanks for all the support of Deutsches Elektronen-Synchrotron (DESY) sincerely.

References

- [1] MATLAB Creating Graphical User Interfaces @ *COPYRIGHT by The MathWorks, Inc.*
- [2] MATLAB Primer @ *COPYRIGHT by The MathWorks, Inc.*
- [3] *I. Grguras*¹, *A. R. Maier*^{2,3}, *C. Behrens*⁴, *T. Mazza*⁵, *T. J. Kelly*⁶, *P. Radcliffe*⁵, *S. Dusterer*⁴, *A. K. Kazansky*^{7,8,9}, *N. M. Kabachnik*^{5,9,10}, *Th. Tschentscher*⁵, *J. T. Costello*⁶, *M. Meyer*⁵, *M. C. Hoffmann*^{1,11}, *H. Schlarb*⁴ and *A. L. Cavalieri*^{1*} Ultrafast X-ray pulse characterization at free-electron lasers. *Nature Photonics*.VOL 6.DECEMBER 852(2012)