JOINT INSTITUTE FOR NUCLEAR RESEARCH

Veksler and Baldin laboratory of High Energy Physics

FINAL REPORT ON THE INTEREST PROGRAMME

Project of an application for configuring digitizer

Supervisor:

Mr Wojciech Piątek

Student:

Przemysław Kamiński
Cardinal Stefan Wyszyński University in Warsaw

Participation period:

May 24 – July 9, Wave 4

Dubna, 2021

Abstract

The goal of the project was to create an user-friendly interface for the CAEN digitizer series 1725 for simpler and less user error-prone modification of it’s 32 bit register values, via an application created with the C++ implementation of Qt library.

Those modifications are made via clickable GUI that allows the user to change the register values by selected bit or in relation of the element that is connected with the selected bit/s, for example setting up a flag, choosing a mode or picking a numerical value.

Introduction

One of the functions of the Front End electronics in nuclear physics applications is to acquire the electrical charge pulses generated by a detector. This information is acquired and saved by data acquisition systems(DAQ). Then analyzed and processed offline by the computer.

The acquisition system is usually completed by other electronic devices which may give further information such as the particle position or trajectory, generate triggers, etc.

Nowadays, the use of waveform digitizers for readout of radiation detectors is very popular and conventional analog electronics is replaced by a full digital approach.

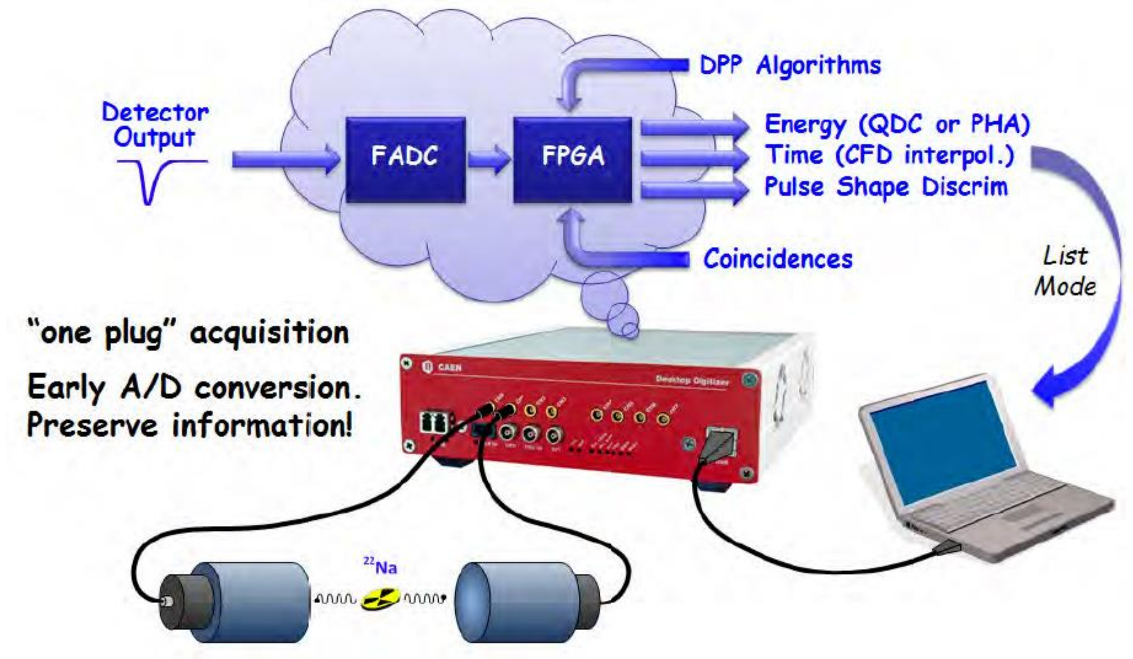


Figure 1. The schema with An example of application digitizer with FPGA.

Processing (DPP) and apply dedicated algorithms online (implemented using FPGAs), to extract the information from the raw waveform. These algorithms allow the digitizer to implement in one place/one device the different functionalities which usually require more electronics like TDC, QDC, Peak Sensing ADC, discriminator, and other analog and logic modules.

Motivation

Most of the experiments at ACCULINNA-2 facility are focused on the studies of neutron-rich nuclei. Neutron and charged-particle detectors provide an excellent selection of the decay channels aiming to obtain information about the structure and properties of light exotic nuclei. The light charged particles are usually detected by Si or CsI detector array, and the neutrons are detected by an array of stilbene - organic scintillator detectors.

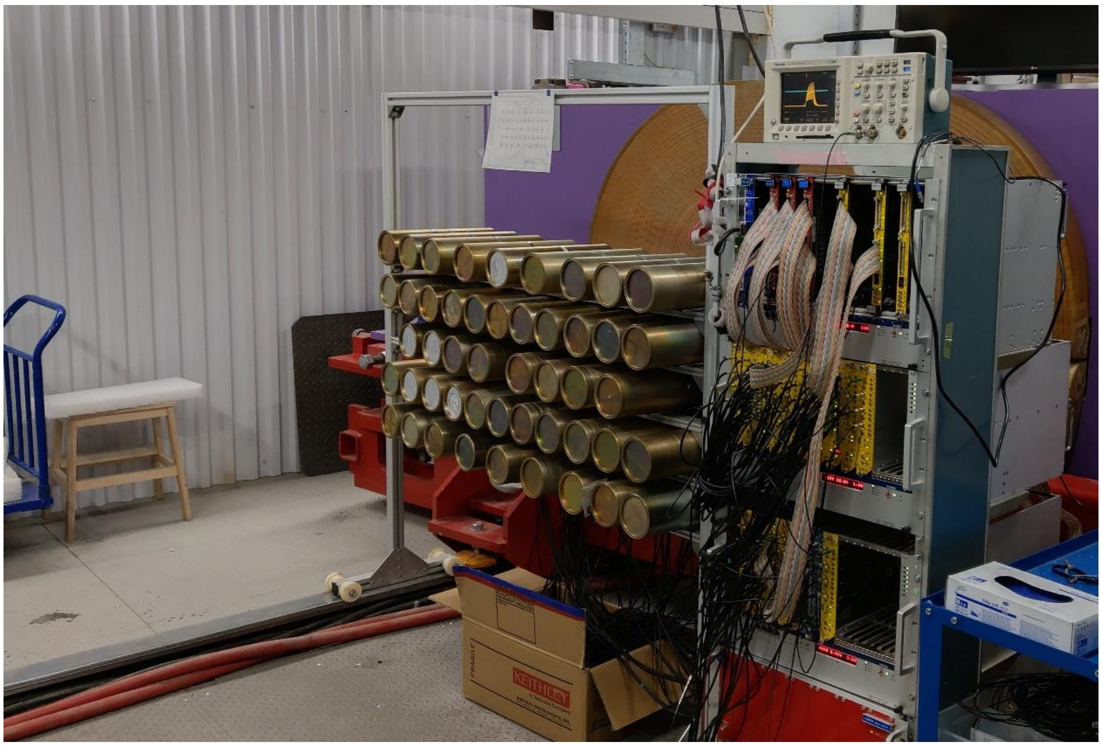


Figure 2. An array of neutron detectors at ACCULINNA-2

Fast neutron detectors are widely used in a number of nuclear physics experiments. Stilbene scintillators have very good n–γ discrimination in comparison to the liquid scintillators used in the past experiments at ACCULINNA.

The neutron detector is made of an organic scintillator module. The sensitive part of the module is a cylinder made of stilbene crystal, C14H 12. In described array, each cylinder has an 80 mm diameter and 50 mm thickness. Every crystal was covered with reflective MgO powder, and inserted into the aluminum housing, and fixed to the PMT. To decrease the background signals produced by charged particles or γ-rays, PMT-crystal systems were put into the steel tubes.

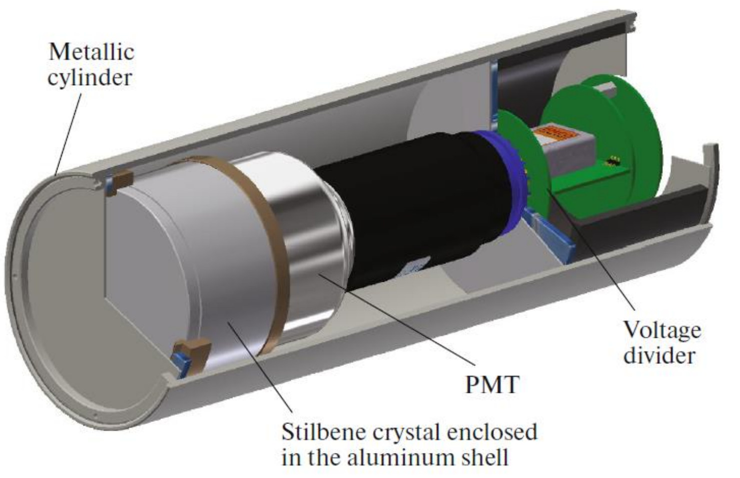


Figure 3. Construction of the neutron detector

The fast-rising of number of neutron detectors makes a purchase necessary electronic devices for DAQ. It was decided to replace part of the analog electronics with a newer digital approach.

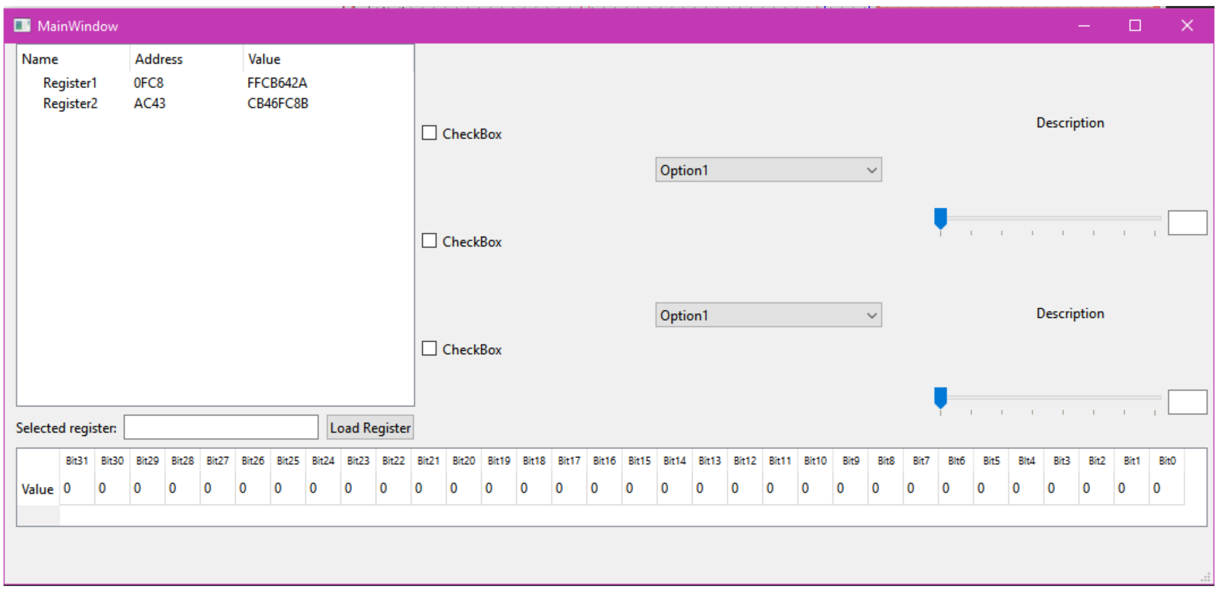
Project goals:

Creating a user-friendly interface to explicitly edit the elements of the register without the need of looking up the manual, to check up what given bits of a register mean.

Creating a scalable solution for creating a dynamic UI, to allow for any number of handled elements to be displayed, due to various properties of different registers.

Creating a base for further projects regarding communication with CAEN digitizer series 1725 and using the data collected.

Application structure

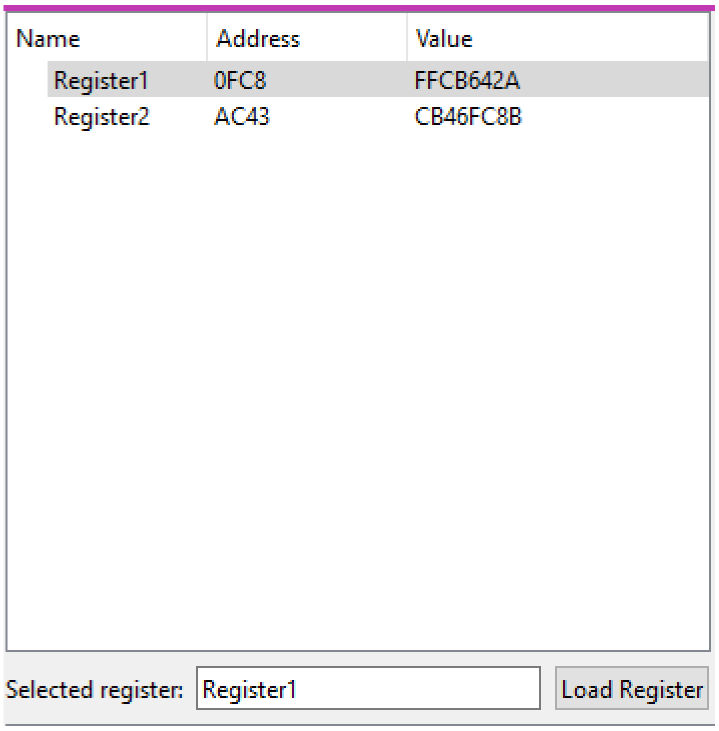


Name	Address	Value
Register1	0FC8	FFCB642A
Register2	AC43	CB46FC8B

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4. Sample application view

The application is divided into three modules created from elements provided in Qt library:

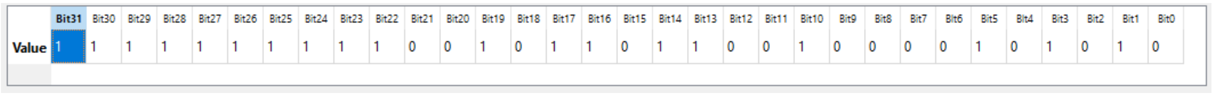


Name	Address	Value
Register1	0FC8	FFCB642A
Register2	AC43	CB46FC8B

Selected register: Register1

Figure 5. The navigation tree

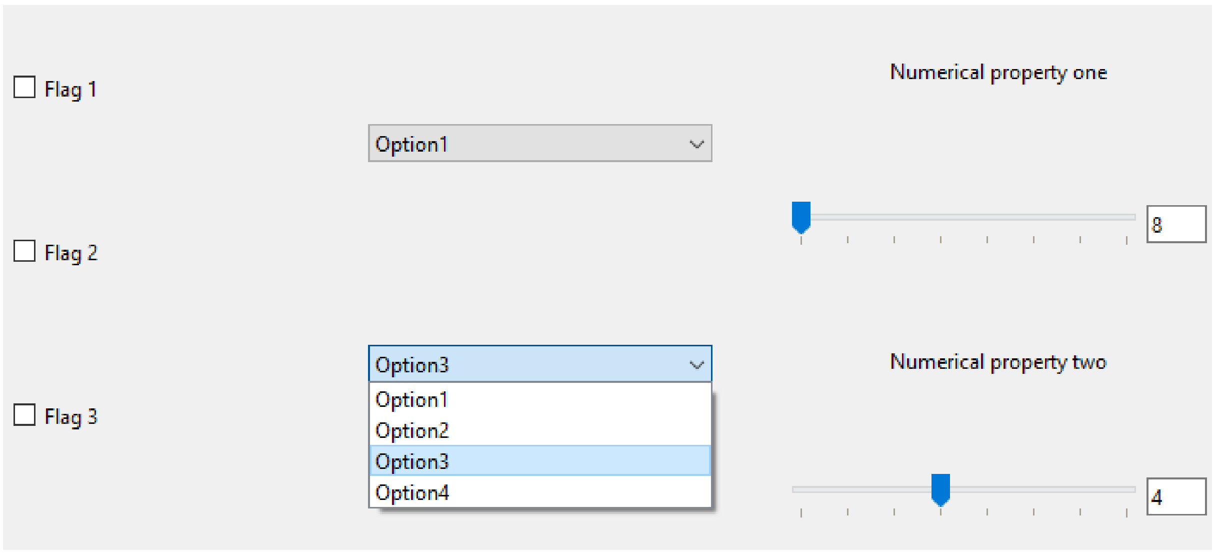
The navigation tree – a QTreeWidget, that allows user to navigate through the list of the available registers, load other register from text files and select one to be edited/checked in depth.



	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Value	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	0	1	1	0	0	1	0	0	0	0	1	0	1	0	1	0

Figure 6. The register table

The register table – a QTableWidget allowing the user to see binary value of a register selected in the navigation tree, as well as allowing them to edit its bits by hand.



☐ Flag 1

☐ Flag 2

☐ Flag 3

Option1

Option3

Option1
Option2
Option3
Option4

Numerical property one: 8

Numerical property two: 4

Figure 7. The editing panel

The editing panel – this panel is a set of dynamically created checkboxes (QCheckBox) for flags, colapsable lists (QComboBox) for modes and sliders (QHorizontalSlider) for numerical values, that allow the user to edit the register values in more intuitive and user-friendly way, than editing simple bits, thanks to the descriptions of elements changed by choosing certain options in the panel.

The editing panel uses additional text files that are connected to the register's names and hold data for creating the UI elements, as well as preset the editable bits in the register.

Text file structure:
UI element type name (i.e. CHECKBOX) -> this allows to identify which element should be created

Index of the first bit of element (the location of the edited element in the register) -> this setups the info for communication with the register table, the data is to be stored within a list that is relative to the UI setup

Bit length (isn't used in Checkboxes)-> gives information about how many bits are occupied by the given element, which allows the creation functions to calculate how many elements will be in a ComboBox, the range of a Slider or how many bits to disable if the elements is restricted, this data is also stored within a list for communication with the register table, since it allows to calculate the bits which had their values changed.

Internal multiplier (only for Sliders) -> some values have some multipliers connected to them, for example a timer may treat a value of 1 as 4 ns, so the multiplier is used to omit the need for user to calculate such values on their own.

Element description -> text string explaining the role of a given element, for ComboBoxes there are multiple lines, each describing one option.

All of above are repeated for each element residing in a register.

Sample program flow :

1) Application has been started with some register preloaded into the navigation tree.
2)
   a)
      i) User clicks “Load Register” button, below the navigation tree.
      ii) A file dialog opens, asking the user to load a premade register data file.
      iii) Selected file gets loaded into the navigation tree.
   b)
      i) User selects a register from the navigation tree, by clicking on it.
      ii) Register value is loaded into the register table to display it in binary format.
      iii) Premade register UI file (if exists) is loaded to dynamically create the UI elements – checkboxes, comboboxes and sliders, depending on the configuration written in UI file and certain bits in the register table got disabled, if the UI file states so.
3)
   a)
      i) User double clicks on a selected bit from the register table.
      ii) The value of a bit changes correspondingly from 0 to 1 or from 1 to 0.
      iii) The value of the register gets updated in the navigation tree.
      iv) The editing panel’s element changes correspondingly: checkbox checks or unchecks, combo box switches to corresponding option, slider changes its position.
   b)
      i) User clicks on a checkbox in the editing panel.
      ii) Corresponding bit in the register table changes its value to 1 for checked checkbox or to 0 for unchecked checkbox.
      iii) The value of the register gets updated in the navigation tree.
   c)
      i) User clicks on a combobox in the editing panel.
      ii) A list of options in the combobox gets expanded allowing the user to select one of them.
      iii) User selects one of the options by clicking on it.
      iv) Corresponding bits in the register table get recalculated basing on the selected option’s index.
      v) The value of the register gets updated in the navigation tree.

d)
  i) User drags on the slider in the editing panel.
  ii) The value displayed in the LineEdit next to the slider gets updated.
  iii) Corresponding bits in the register table get recalculated basing on the slider's value and internal multiplier.
  iv) The value of the register gets updated in the navigation tree.

4) Quit the program.

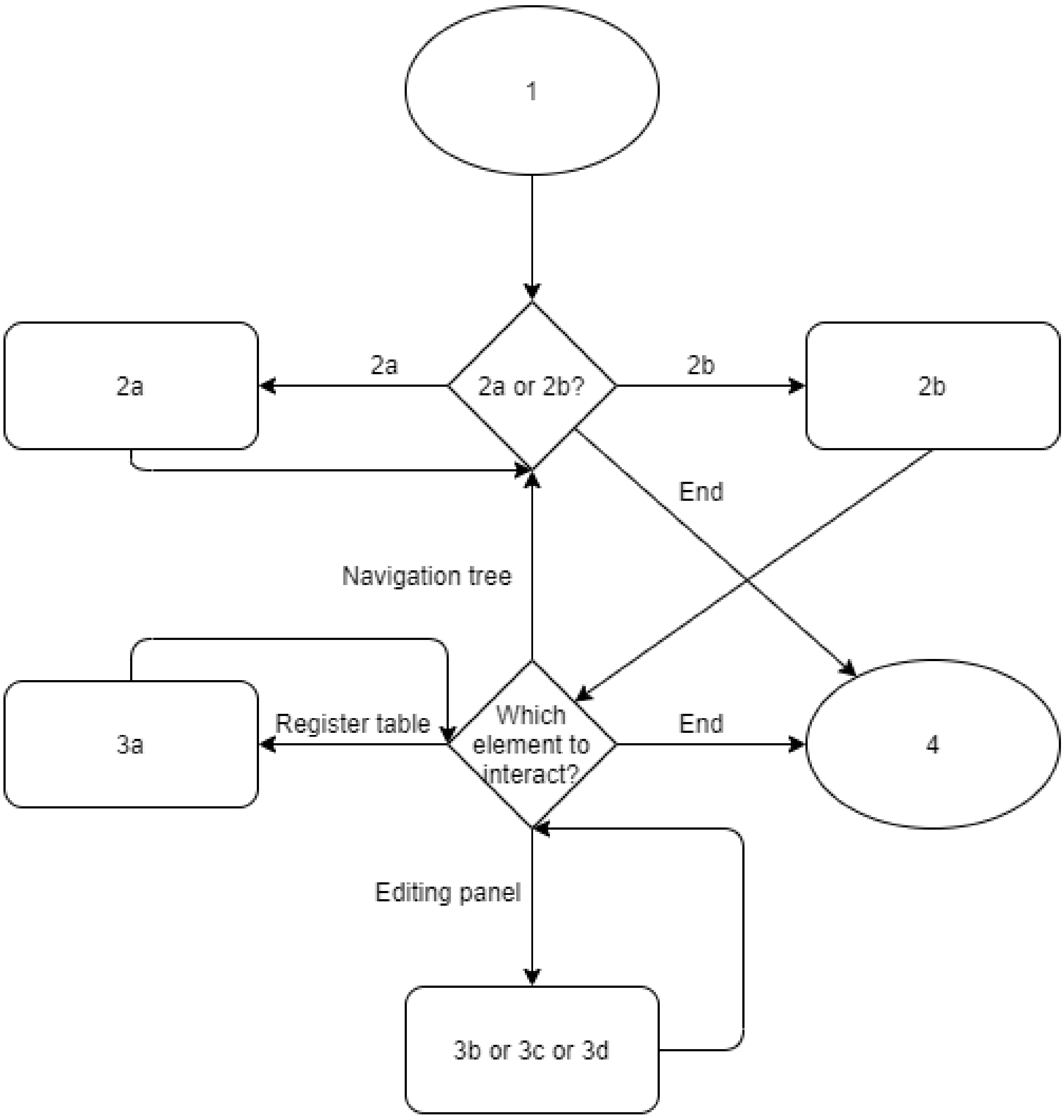


Figure 8. The program flow diagram

Summary

The final goal of the project hasn’t been met, due to the creation process taking longer than scheduled. In effect the communication with physical digitizer unit is yet impossible, but it is possible to interact with virtual instances of the registers, change their values via the register table, load new registers via the navigation tree, load dynamic UI from the UI file and interact with it.

References

1. Pro Git Book by Scott Chacon and Ben Straub - https://git-scm.com/book/en/v2
2. Qt documentation - https://doc.qt.io/
3. 3. CAEN User Manual UM2792 V1730/VX1730 & V1725/VX1725 Rev.6 – https://caen.it/download
4. CAEN User Manual UM1935 CAENDigitizer Library Rev. 21 - https://www.caen.it/products/caendigitizer-library/
5. CAEN User Manual UM2784 CAENDigitizer LabVIEW Library Rev.2 - https://www.caen.it/products/caendigitizer-library/
6. G.Kamiński et al., 2014, ‘Neutron Detector Array Based on Stilbene Crystals for the ACCULINNA and ACCULINNA-2 Separators’, *Acta Physica Polonica Series B (ACTA PHYS POL B),* vol.45, p.519
7. A.A.Bezbakh et al., 2018, ‘Neutron Spectrometer for Experiments with Radioactive Beams on the ACCULINNA-2 Fragment Separator’, *Instruments and Experimental Techniques* , vol.61 no.5, p.631-638