Nazarbayev University

Bachelor Thesis

Design of a Data Acquisition System for Coincidence Measurement of Resonant Nuclear Reactions in Inverse Kinematics

Author: Assylzhan Tenelbayev

Supervisor: Dr. Aliya Nurmukhanbetova, Facolt`a di Ingegneria e Architettura, Universit`a degli Studi di Enna "Kore",, Enna, Italy.

Supervisor: Prof. Alexander Tikhonov, Department of Physics, School of Sciences and Humanities, Nazarbayev University, Astana, Kazakhstan.

Co-Supervisor: Dr. Dosbol Nauruzbayev, NURA, Astana, Kazakhstan.

Abstract

This thesis presents the work on the modification of the existing data acquisition (DAQ) system for measuring the coincidences of proton and gamma particles in resonant nuclear reactions using the Thick Target Inverse Kinematics (TTIK) method. The aim was to enable the simultaneous detection of proton and gamma particles through the integration of the LaBr3 gamma detector into the initial DAQ system, which previously included only silicon detectors. The aim was reached by adding a second pair of ADC and TDC modules to the system and making corresponding changes in the configuration files for correct work of the modified DAQ system.

Moreover, I developed a Python script for calibration of the detectors. The silicon detectors were successfully calibrated using the alpha source. Using the known energy values of the Ra-226 alpha decay, the raw data from the DAQ system was converted into energy-count spectra suitable for further analysis. Despite the successful integration of the new hardware components, the configuration issues in the spectclSetup.tcl did not allow a visualization of the data from the LaBr3 gamma detector. Overall, the work is fundamentally important for measuring the proton-gamma coincidences in the future experiments.

1. Introduction

The α-clustering phenomenon is a key characteristic in the structure of light nuclei consisting of multiples of four nucleons (4N nuclei), such as 8Be, 12C, and 16O. An alpha cluster state is associated with a specific arrangement within the nucleus where nucleons are organized into groups consisting of two protons and two neutrons, known as alpha particles [1]. Resonant reactions are considered to be one of the most powerful experimental tools to study alpha cluster states. The analysis of the data obtained from resonant reactions allow the investigation of quantum characteristics of the nuclei such as spin, orbital angular momentum, and resonant width [2]. Recent success in rare beams development and in the experimental technique resulted in a rapid increase in studies of the cluster structure in N≠Z nuclei. New studies have shown that the alpha cluster structure in N≠Z nuclei is much more complicated than the quasi rotational bands structure in 4N nuclei, and contemporary shell models face significant challenges in fully accounting for these experimental results [1]. Resonant reactions like α+14N help to broaden our understanding of nuclear structure and the role of clustering.

The construction of DC-60 cyclotron in Astana provided the opportunity to study low energy resonance reactions. Initially, DC-60 was used for research in material science and for educational purposes [3]. Low beam energy resolution and difficulties in adjusting the energy of the beam did not allow the research in nuclear physics. However, with the use of Thick Target Inverse Kinematics (TTIK) approach, DC-60 cyclotron was successfully integrated into nuclear physics experiments [4-5].

The research group from Astana, including Dr. Aliya Nurmukhanbetova and Dr. Dosbol Nauryzbayev, studied the alpha cluster structure of 18F through the resonant 14N + α scattering by implementing the TTIK approach [1]. The resonant reaction between 14N and alpha forms the compound nucleus 18F, which then decays by three possible channels shown in Figure 1:

1) elastic scattering of α particles in 14N(α, α)14N

2) p_0 emission in 14N(α, p_0)17O

3) p_1 emission in 14N(α, p_1)17O*, where the gamma-ray was emitted from the excited 17O* nucleus.

Although the results of the study were significant, the research group could not measure proton and gamma coincidences. Measurements of the coincidences are important as they help to identify whether the registered proton and gamma particles came from the same decay channel, thereby allowing to reduce the background noise. Resolving this limitation can help to significantly improve the studies of TTIK-based resonant reactions.

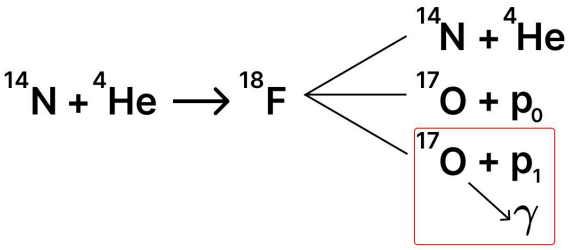

Figure 1. Decay channels of 18F nuclei

The aim of the thesis is to improve the measurements of proton and gamma coincidences in resonant nuclear reactions using the Thick Target Inverse Kinematics (TTIK) approach. For this purpose, the initial data acquisition (DAQ) system should be modified by integration of the LaBr3 gamma detector into the experimental setup. Such modification requires the corresponding integration of another pair of Analog-to-Digital converter (ADC) and Time-to-Digital converter (TDC) modules. Moreover, the configuration files are modified to launch the modified DAQ system. The primary objective of this thesis is to improve the DAQ system to support new experimental requirements, thereby allowing the detection and analysis of proton-gamma coincidences from LaBr3 and silicon detectors.

2. Theoretical Background

2.1 Thick Target Inverse Kinematics (TTIK) and Time of Flight (TOF) methods

The direct approach is widely used in nuclear physics for resonant reaction studies with the use of electrostatic accelerators. This method involves the launch of a beam of light particles, such as protons or heavy ions, into a thin target and observing the final products [6-7]. One of the main advantages of this method is nearly perfect resolution, which is important for studying resonant reactions. However, it has disadvantages such as an impurity of the target and the impossibility of making measurements near 180 degrees. Moreover, in order to obtain the whole excitation function the researchers are required to make small adjustments to the beam energy. This makes the experiments with the use of direct approach time-consuming.

The Thick Target Inverse Kinematics (TTIK) is an alternative method, which is widely used at cyclotrons to study low-energy heavy ion reactions. This method works by directing the heavy ion beam into a thick target, often a helium or hydrogen gas inside of a scattering chamber. Due to interactions with the gas particles, the ions gradually slow down as they pass through the chamber. The heavy ions stop within the target, while light recoils like protons or alpha particles reach the silicon detectors located at the end of the chamber. It is possible due to the lower ionization losses of the light recoils compared to heavy ions. One of the main advantages of the TTIK approach is that researchers can obtain a continuous excitation function in only one launch as the energy of the beam decreases while slowing down inside of the chamber. Moreover, as the heavy ions are stopped within the gas not reaching the detectors, this allows the measurements at 180 angles with the minimal Rutherford scattering. Such

conditions are difficult to be achieved in a direct approach, that is why TTIK is a suitable method for studying low energy resonant reactions [8].

The Time-of-Flight (ToF) method is commonly used in combination with TTIK to improve the identification of particles. The ToF method measures the time it takes for the beam to travel from the cyclotron to the reaction points and the time of travel of recoil particles from the reaction point to the detector. In order to calculate the time interval, the radiofrequency of the cyclotron is set as the “start” signal, while the moment of reaching the detector is set as the “stop” signal. Using these time intervals, the velocity and energy of the light recoils can be calculated [8].

Therefore, while a direct approach is effective for many nuclear reactions, TTIK combined with the ToF method is more suitable for studying resonant reactions involving rare isotope beams or low-energy scattering [9].

2.2 Data Acquisition Systems in Nuclear Physics

Fundamental and applied research in nuclear physics is done through the use of ion accelerators or cyclotrons and radiation detectors, which provides the relevant physical information. However, the data is usually obtained in the form of voltage or charge signals, which should go through the processes of digitization and processing before they can be recorded in a computer for further analysis [10].

In this context, the Data Acquisition (DAQ) systems work as a communicator between the radiation detectors and the recording computer system. Their role lies in converting analog signals obtained from the detectors into the digital form, suitable for the further analysis and storage. The primary components of the DAQ system are (1) detectors, (2) signal conditioning electrons, (3) digitization modules (such as ADCs and TDCs), and (4) software for managing data flow and analysis.

(1) Silicon detectors are a type of semiconductor detectors widely used in nuclear physics experiments. In our lab, we use 10x10 mm silicon detectors from Hammatsu. The operating principle of these detectors is based on the ionization of a semiconductor medium by incoming radiation: when a charged particle passes through the silicon material, it interacts with the silicon atoms and loses energy, producing electron-hole pairs. A reverse bias voltage (usually in the range of 60–100 V, depending on the thickness and design of the detector) is applied to the detector, creating a depletion region. The newly created electrons and holes in the p-n junction drift due to the concentration gradient, generating an internal electric field across the junction. This field drives electrons toward the N-type electrode and the holes toward the P-type electrode, resulting in a small current at the electrodes. This pulse is then processed by the DAQ system to extract timing, energy, and possibly position information about the incident particle.

(2) The signal from the silicon detectors goes through the preamplifier, which strengthens the weak electronic signal and clears it from noise, making it suitable for further processing. In our lab, we use a 16-channel preamplifier Mesytec.

(3) Analog-to-Digital Converter (ADC) and Time-to-Digital Converter (TDC) are the modules used to digitize the signal. The role of ADC lies in converting the conditioned analog signal into a digital form. An ADC works by repeatedly measuring an analog signal at specific intervals, and then turning that value into a digital value. The TDC is used to measure the time intervals between “start” and “stop” signals, which is then used in the time-of-flight method.

The entire process is managed by the DAQ software, which is used to configure the ADC and TDC modules, collect and digitize the data, and finally visualize the spectra.

Configuration files are used to set the settings for the DAQ system hardware, such as ADC and TDC modules, which allows the proper collection and processing of the data. The *daqconfig.tcl* configuration file is used to set the parameters for both ADC and TDC modules.. For example, as it can be seen from Figure 2, there are different options available for the customization of the ADC modules. The *-base* subcommand is used for setting the base address for the module as configured in its rotary switches. This base address is used to access the register and event memory of the module. Each module must have a unique base address to avoid the conflicts.

```
madc create adc1 -base 0xAAAA0000 -id 7 -ipl 0
madc config adc1 -gatemode common
madc config adc1 -gategenerator disabled
madc config adc1 -inputrange 10v
madc config adc1 -resolution 4khires
madc config adc1 -timestamp off
```

Figure 2. Configuration of ADC module

Another configuration option, such as *-id*, is used to assign the unique virtual slot number (vsn) for the modules. The vsn is integrated inside of the event data sent by the module, which is used by the system for correct interpretation of the data. The *-ipl* subcommand is used to set the priority level (from 1 to 7) for the module that is used to trigger the interrupt driven stack. This value must match the interrupt priority level used to trigger the stack.

Other settings, like -timestamp, are used to control whether or not the module tags each event with a trigger number or with a timestamp. Subcommands such as -gatemode and -gategenerator are used to configure the gate inputs used to control the timing of the data collection. The -inputrange programs the input voltage range for the module, taking the value of

4v, 8v, or 10v. The -resolution subcommand sets the resolution of the module and has an impact on the conversion time.

As it can be seen in Figure 3, the configuration of the TDC module uses similar subcommands.

```
adc create tdc1 0xBBBB0000
adc config tdc1 -supressrange false
adc config tdc1 -ipl 0
adc config tdc1 -vector 0
```

Figure 3. Configuration of TDC module

The *spectlSetup.tcl* is a configuration file linked with the Spectcl program, which is responsible for reading binary data (typically in *.evt* files) saved by the DAQ system and making the visualizations of the spectra for the signals recorded by the ADC and TDC modules. Furthermore, the visualized spectra can be exported as a .txt file, which includes the data about the number of channels and the corresponding registered counts. This data is later used for the calibration of the detectors through the construction of the calibration curves, resulting in the final energy-counts spectra, essential for further analysis.

(1) We use the MSU NSCLDAQ and NSCL SpecTcl software to operate our DAQ system. This customized code, developed by Ron Fox from Michigan State University (USA) [https://nscl.msu.edu/public/index.html], provides a robust framework for data acquisition and analysis, tailored to our specific experimental needs. This capability allows us to avoid being limited to specific brands or technologies, enabling the integration of diverse equipment and customization of the system to meet different experimental requirements. A limit of this system is that when using modules from different manufacturers, it is necessary to modify the code of each module to ensure synchronized operation. This also requires configuring individual

configuration files for each component, which can be time-consuming and labor-intensive. As a result, integrating new modules can become a challenging task, especially if different manufacturers have varying standards and requirements.

3. Experimental Setup and DAQ Configuration

3.1 Overview of the Experiment

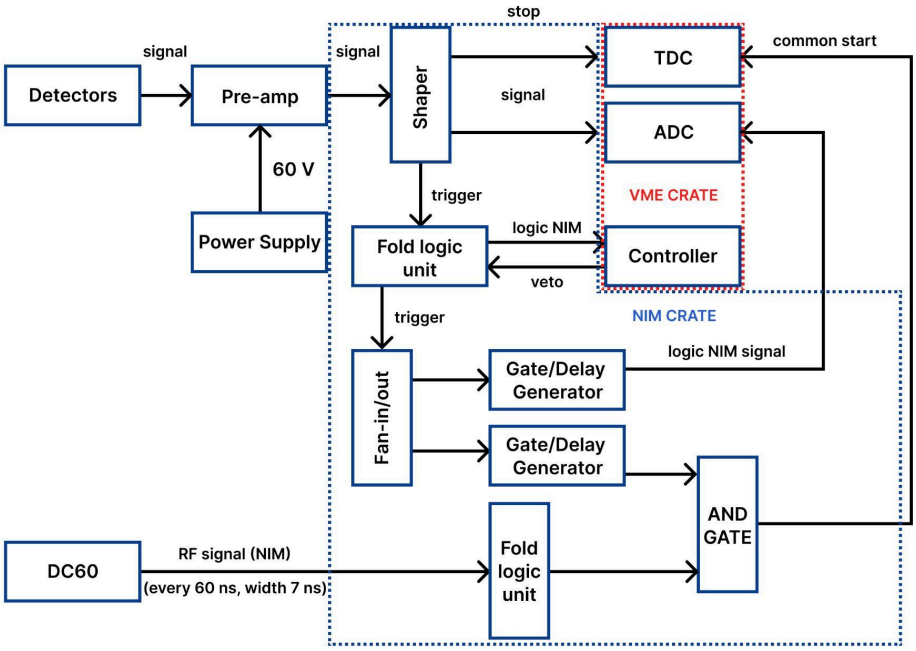

Figure 4. Initial DAQ system

As it can be seen from Figure 4, the initial DAQ system included only one pair ADC and TDC used for digitizing the signals from the silicon detectors. The analog signal from the detectors goes through the several steps before being digitized. Firstly, it passes through the

pre-amplifier, which strengthens the weak electronic signal and clears it from noise, making it suitable for further processing. Apart from that, the pre-amplifier generates a trigger signal, which is formed using the Leading Edge method. This method works by creating a logical trigger to signal the event by detecting the rising edge of the amplified signal. The amplified signal goes into the shaper, where the signal is transformed into a well-defined pulse with a specific rise and fall time, which is important for the proper detection and measurement. The shaped signal then splits into two paths: one goes to the ADC, and the other to TDC. This setup allows the measurement of energy and time of the detected events simultaneously.

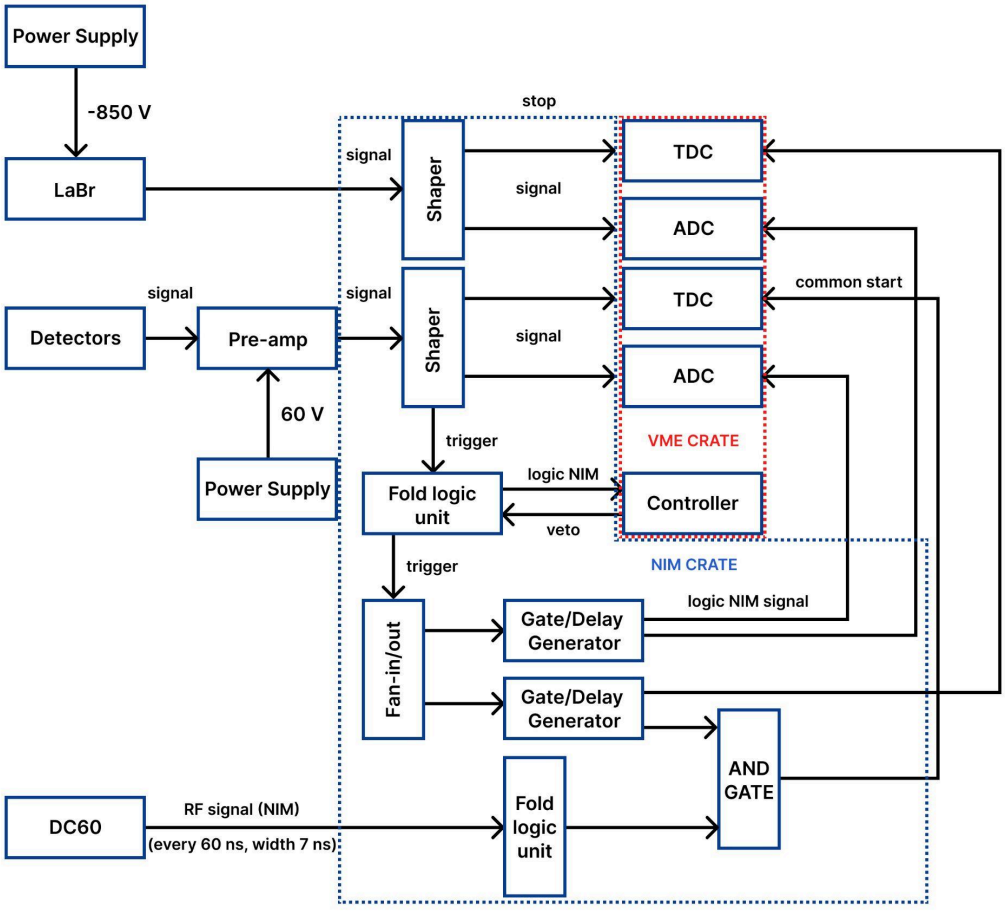

Figure 5 . Modified DAQ system

To allow the detection of the gamma particles, the setup is modified by integrating the LaBr3 gamma detector, a new shaper, and another pair of ADC and TDC (Figure 5). As gamma particles have relatively higher energy, the signals generated by the LaBr3 detector go directly to the shaper, following the same path to ADC and TDC as the signals from the silicon detectors. This kind of modification allows the system to register high-energy gamma signals along with the signals from the silicon detectors, which is essential for measuring the proton and gamma coincidences.

3.2 Modification of Configuration Files for DAQ System

Initially, the configuration files were written to support a single pair of ADC and TDC within the DAQ system, but the integration of another pair of ADC and TDC for registering gamma particles required the modifications in the configuration files for supporting the new setup. Within the code, I created unique names for identifying each of the modules: *adc1* and *adc2* for two ADC modules, *tdc1* and *tdc2* for two TDC modules. Each module was assigned with unique base addresses as indicated in their rotary switches.

Moreover, I carefully performed a configuration of the new modules to ensure the correct synchronization. For example, I set the identical threshold values to both ADC modules, which allows the uniform processing of the signals from both types of the detectors. I also set the timescale of 160 and disabled the common stop setting for both TDC modules to ensure proper time synchronization. Furthermore, I included a chain configuration (caenchain), so all the four modules are treated as a single entity, facilitating more streamlined data collection.

3.3 Calibration

The exported raw data from the Spectcl program must go through the process of the calibration before the analysis. For this purpose, I developed a Python script (Appendix 1), which uses the raw data and known energies of the alpha decay to construct the calibration

curve that is used to obtain the calibrated spectra. Firstly, the script reads the .txt file, where the data is organized into two columns: “Channel” and “Counts”. Here, “Channel” corresponds to a specific range of energies of the detected particle, while “Counts” refers to the number of particles detected within a specific channel over a given time period. Then I plotted the spectra, where the x-axis stands for the “Channel” and the y-axis represents the “Counts”. Using the known peaks resulting from the alpha decay of Ra-226, I calculated and plotted the calibration curves by identifying the peaks in the spectra and matching them with the known energy values of the Ra-226 decay. As a result, the detector channels are converted into their corresponding energy values.

4. Results and Discussion

The primary objective of this thesis was to make the modifications to the existing DAQ system to allow the measurements of the proton-gamma coincidences in resonant nuclear reactions. The initial setup was suitable for detecting the proton particles by silicon detectors. To detect gamma particles, the setup went through several modifications such as the integration of the LaBr3 gamma detector, shaper, and a second pair of ADC and TDC modules.

```
madc create adc1 -base 0xAAAA0000 -id 7 -ipl 0
madc config adc1 -gatemode common
madc config adc1 -gategenerator disabled
madc config adc1 -inputrange 10v
madc config adc1 -resolution 4khires
madc config adc1 -timestamp off

madc create adc2 -base 0x22220000 -id 8 -ipl 0
madc config adc2 -gatemode common
madc config adc2 -gategenerator disabled
madc config adc2 -inputrange 10v
madc config adc2 -resolution 4khires
madc config adc2 -timestamp off

set cutting_value_adc {}
for {set i 0} {$i < 32} {incr i} {
    lappend cutting_value_adc 100
}

madc config adc1 -thresholds $cutting_value_adc
madc config adc2 -thresholds $cutting_value_adc

adc create tdc1 0xBBBB0000
adc config tdc1 -supressrange false
adc config tdc1 -ipl 0
adc config tdc1 -vector 0

adc create tdc2 0xCCCC0000
adc config tdc2 -supressrange false
adc config tdc2 -ipl 0
adc config tdc2 -vector 0

set cutting_value_tdc {}
for {set i 0} {$i < 32} {incr i} {
    lappend cutting_value_tdc 5
}

adc config tdc1 -thresholds $cutting_value_tdc
adc config tdc2 -thresholds $cutting_value_tdc
adc config tdc1 -timescale 160
adc config tdc2 -timescale 160
adc config tdc1 -commonstop false
adc config tdc2 -commonstop false
```

```
caenchain create chain

caenchain config chain -base 0x10000000 -modules [list adc1 adc2 tdc1 tdc2]

stack create events
stack config events -trigger nim1
stack config events -delay 15
stack config events -modules chain

set adcS1 {}
for {set i 0} {$i < 32} {incr i} {
    set name [format "adc1.%02d" $i]
    lappend adcS1 $name
}
set adcChannels(adc1) $adcS1

set adcS2 {}
for {set i 0} {$i < 32} {incr i} {
    set name [format "adc2.%02d" $i]
    lappend adcS2 $name
}
set adcChannels(adc2) $adcS2

set tdcS1 {}
for {set i 0} {$i < 32} {incr i} {
    set name [format "tdc1.%02d" $i]
    lappend tdcS1 $name
}
set adcChannels(tdc1) $tdcS1

set tdcS2 {}
for {set i 0} {$i < 32} {incr i} {
    set name [format "tdc2.%02d" $i]
    lappend tdcS2 $name
}
set adcChannels(tdc2) $tdcS2
```

Figure 6. Modified daqconfig.tcl configuration file

Moreover, as it can be seen from Figure 6, the configuration files went through corresponding changes in order to launch the modified DAQ system. The program was successfully launched, allowing the data collection from both silicon and LaBr3 detectors. However, visualization was only achieved for the data from the silicon detectors, indicating that the *spectclSetup.tcl* configuration file still requires further modifications to process data from the LaBr3 detector. Figure 7 presents the 1D and 2D spectra from Si detectors.

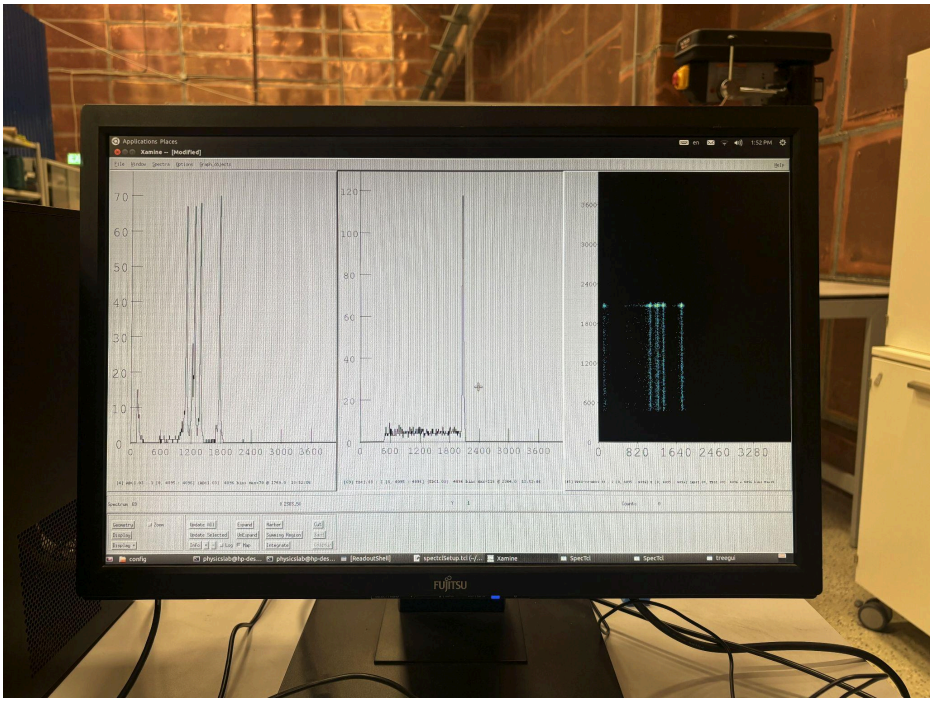

Figure 7. 1D and 2D Spectra from Si detectors

Furthermore, I exported the raw data from the silicon detectors and constructed the calibration curve (Figure 9), which was used to convert the channel-counts spectra (Figure 8) into energy-counts spectra (Figure 9) suitable for further analysis. The red points on the graph indicate the energy peaks of the alpha decay from Ra-226, with the known energies of 4.78, 5.49, 6.01, and 7.89 MeV. The resulting spectra displayed sharp and narrow peaks, indicating stable detector performance and a low noise level. The narrowness of the peaks further suggests a good energy resolution of the system.

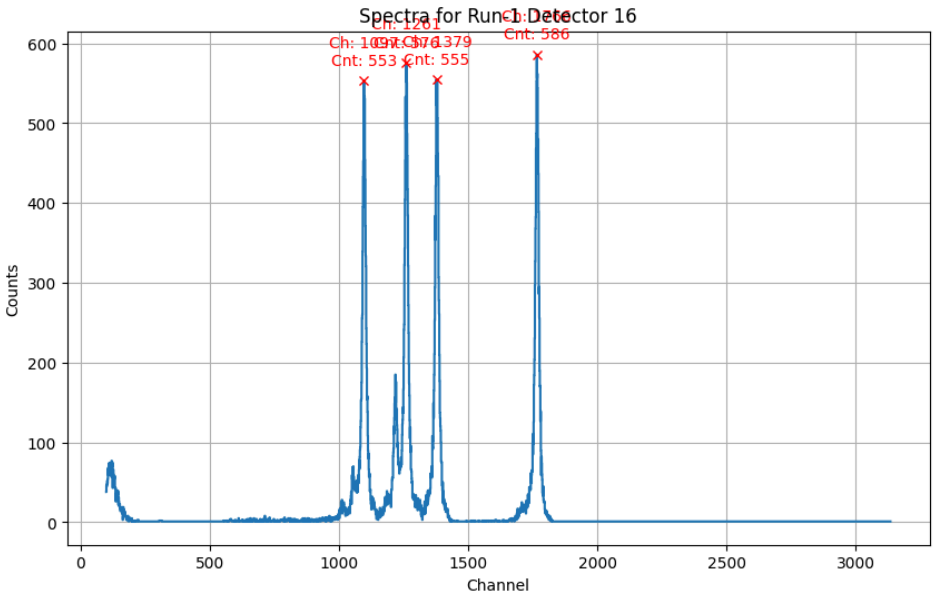

Figure 8. Uncalibrated Spectra

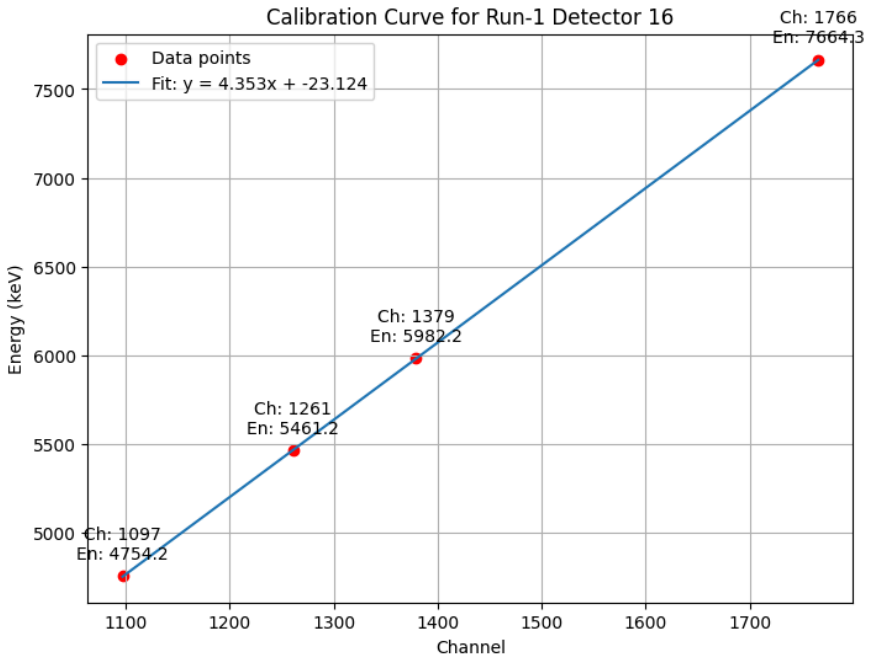

Figure 9. Calibration curve

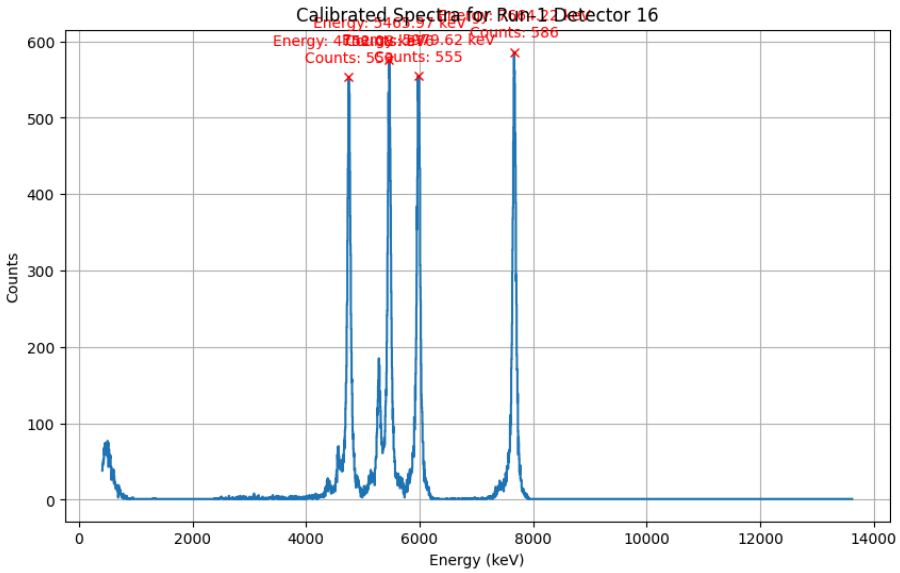

Figure 10. Calibrated spectra

5. Conclusion

The main objective of this thesis was to upgrade the initial data acquisition (DAQ) system for the proton-gamma coincidence measurements in resonant nuclear reactions using the Thick Target Inverse Kinematics method. The DAQ system was successfully modified by integrating LaBr3 gamma detector, shaper, and a pair of ADC and TDC modules. With the corresponding changes in the DAQ configuration files, the program was launched without issues. However, the objective of obtaining proton-gamma coincidence was not fully achieved. Although both pairs of ADC and TDC modules were functioning simultaneously, I was able to obtain the spectra only from the silicon detectors.

Although the data was properly collected from both silicon and gamma detectors, I could not obtain any spectra from the LaBr3 detector. One of the possible problems is the incomplete modification of the spectclSetup.tcl configuration file, which is used by the Spectcl program to visualize the spectra.

Moreover, I successfully developed a calibration script on Python, which converts the raw data from the DAQ system into the energy spectra suitable for analysis. The script was successfully tested on the silicon detectors, which were calibrated using the alpha source. This code can be further used to calibrate the LaBr3 detector using the gamma source.

The results show that while the most modifications of the DAQ system were successful, the *spectclSetup.tcl* configuration file requires further improvements to obtain the visualization of data from the LaBr3 detector.

Overall, the presented thesis work significantly contributes to the development of the DAQ system for improving the measurements of proton-gamma coincidences. With further modifications in the configuration files, the new DAQ system will be capable of improving the studies of the resonant reactions by providing data on proton-gamma coincidences.

6. References

[1] A. K. Nurmukhanbetova, V. Z. Goldberg, A. Volya, D. K. Nauruzbayev, and G. V. Rogachev, “18F alpha cluster structure in the resonant 14N+alpha scattering,” *Eur. Phys. J. A*, vol. 60, no. 11, 2024.

[2] I. Lombardo and D. Dell’Aquila, “Clusters in light nuclei: history and recent developments,” *Riv. Nuovo Cimento*, vol. 46, no. 9, pp. 521–618, 2023.

[3] B. N. Gikal *et al.*, “Secondary fusion reactions in the bombardment of light-element targets with low-energy heavy ions,” *Phys. Part. Nucl. Lett.*, vol. 11, no. 4, pp. 462–466, 2014.

[4] K. P. Artemov, O. P. Belyanin, A. L. Vetoshkinи . and others, «Effective method of investigation of α-cluster states», Yadernaya Fizika, т. 52, вып. 3, сс. 634–639, 1990.

[5] A. K. Nurmukhanbetova *et al.*, “Implementation of TTIK method and time of flight for resonance reaction studies at heavy ion accelerator DC-60,” *Nucl. Instrum. Methods Phys. Res. A*, vol. 847, pp. 125–129, 2017.

[6] G. F. Ciani and A. Formicola, “Direct measurement of the $^{13}C(\alpha,n)^{16}O$ reaction in the Gamow window of the s-process nucleosynthesis,” *J. Phys. Conf. Ser.*, vol. 2453, no. 1, p. 012018, 2023.

[7] M. Heil *et al.*, “The13C(α,n)reaction and its role as a neutron source for the s process,” *Phys. Rev. C Nucl. Phys.*, vol. 78, no. 2, 2008.

[8] A. K. Nurmukhanbetova, V. Z. Goldberg, D. K. Nauruzbayev, M. S. Golovkov, and A. Volya, “Evidence for α- cluster structure in 21Ne in the first measurement of resonant 17O+α elastic scattering,” *Phys. Rev. C*, vol. 100, 2019.

[9] A. Volya, M. Barbui, V. Z. Goldberg, and G. V. Rogachev, “Superradiance in alpha clustered mirror nuclei,” *Commun. Phys.*, vol. 5, no. 1, 2022.

[10] Peter, Mr, et al. *Working Material Report of the Technical Meeting (F1-TM-1805383) on Data Acquisition Systems Used for Nuclear Instrumentation at Particle Accelerator Facilities*. 2019.

Appendix 1

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.signal import find_peaks
from scipy.optimize import curve_fit

# Function to split data at negative values
def split_data(data):
    segments = []
    current_segment = []
    for index, row in data.iterrows():
        if row['Counts'] < 0:
            if current_segment:
                segments.append(pd.DataFrame(current_segment, columns=['Channel', 'Counts']))
                current_segment = []
        else:
            current_segment.append(row)
    if current_segment:
        segments.append(pd.DataFrame(current_segment, columns=['Channel', 'Counts']))
    return segments

# Function to find and annotate top peaks
def find_and_annotate_peaks(segment, num_peaks=4, distance=8):
    channels = segment['Channel']
    counts = segment['Counts']

    # Find peaks
    peaks, _ = find_peaks(counts, distance=distance)

    # Get the top peaks
    top_peaks = peaks[np.argsort(counts[peaks])[-num_peaks:]]
    top_peaks.sort()  # Sort the top peaks to correspond with known energies

    return top_peaks

# Function to process and plot segments
def process_and_plot(file_path, run_number, j):
    # Load the data
    data = pd.read_csv(file_path, sep='\s+', header=None, names=['Channel', 'Counts'])

    # Split the data
    segments = split_data(data)
```

```
    # Dictionary to store peak channels for each detector
    peak_channels = {i: [] for i in range(1, 16)}

    # Process and plot each segment
    for i, segment in enumerate(segments, start=1):
        j = j + 1
        # Clean and process data
        segment['Channel'] = segment['Channel'].str.replace('(', '').str.replace(')', '').astype(int)
        segment = segment[segment['Counts'] >= 0]
        segment = segment.groupby('Channel', as_index=False)['Counts'].sum()

        # Find and annotate peaks
        top_peaks = find_and_annotate_peaks(segment)
        peak_channels[i] = segment['Channel'].iloc[top_peaks].values

        # Plot the spectrum
        plt.figure(figsize=(10, 6))
        plt.plot(segment['Channel'], segment['Counts'], drawstyle='steps-mid')
        plt.xlabel('Channel')
        plt.ylabel('Counts')
        plt.title(f'Spectra for Run-{run_number} Detector {j}')
        plt.grid(True)

        # Annotate peaks
        #for peak in top_peaks:
        #    count = segment['Counts'].iloc[peak]
        #    plt.annotate(f'Ch: {channel}\nCnt: {count}',
        #                 (channel, count),
        #                 textcoords="offset points",
        #                 xytext=(0,10),
        #                 ha='center',
        #                 color='red')
        #    plt.plot(channel, count, "x", color='red')

       # plt.show()

    return peak_channels

# File paths for the runs
file_paths = {
    1:   '/content/drive/MyDrive/Data/adc1.txt'
}
```

```
# Dictionary to store peak channels for each run
all_peak_channels = {}

# Process and plot for each run
for run_number, file_path in file_paths.items():
    peak_channels = process_and_plot(file_path, run_number, 15)
    all_peak_channels[run_number] = peak_channels
```

```python
# Known energies (in keV) from the Ra-226 source
known_energies = [4754.2, 5461.18, 5982.2, 7664.3]

# Linear function for calibration
def linear_fit(x, a, b):
    return a * x + b

# Function to fit calibration curve and plot results
def calibrate_and_plot(run_number, peak_channels, j):
    calibration_params = []
    for i, channels in peak_channels.items():
        j = j + 1
        if len(channels) >= 4:  # Check for at least 4 peaks
            channels = channels[:4]  # Use only the first 4 peaks

            # Fit linear function to the data
            params, covariance = curve_fit(linear_fit, channels, known_energies)
            a, b = params
            calibration_params.append((a, b))

            # Plot calibration curve
            plt.figure(figsize=(8, 6))
            plt.scatter(channels, known_energies, color='red', label='Data points')
            plt.plot(channels, linear_fit(np.array(channels), a, b), label=f'Fit: y = {a:.3f}x + {b:.3f}')
            for channel, energy in zip(channels, known_energies):
                plt.annotate(f'Ch: {channel}\nEn: {energy:.1f}',
                             (channel, energy),
                             textcoords="offset points",
                             xytext=(0,10),
                             ha='center')
            plt.xlabel('Channel')
            plt.ylabel('Energy (keV)')
            plt.title(f'Calibration Curve for Run-{run_number} Detector {j}')
            plt.legend()
            plt.grid(True)
            plt.show()

            print(f'Detector {i} - Run {run_number}: a = {a:.3f}, b = {b:.3f}')
        else:
            print(f'Detector {i} - Run {run_number}: Not enough peaks detected.')

    return calibration_params

# Perform calibration for each run
calibration_params_all_runs = {}
for run_number, peak_channels in all_peak_channels.items():
    print(f'\nCalibration for Run-{run_number}')
    calibration_params = calibrate_and_plot(run_number, peak_channels, 15)
    calibration_params_all_runs[run_number] = calibration_params
```