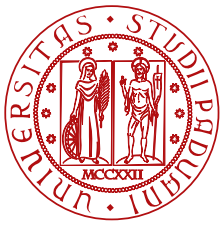


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Design and development of a coordination and protection system for the SPES project

Relatore
Prof. Gian Antonio Susto

Laureando
Filippo Mazzarotto

Correlatore
Davide Marcato

ANNO ACCADEMICO 2023-2024

Data di laurea 12/03/2024

For all the people who, with small steps, shape the future:
teachers, educators, researchers, designers, engineers.
And for all those who shaped mine.

Abstract

The National Laboratories of Legnaro (LNL) are currently developing a project called Selective Production of Exotic Species (SPES), aimed at generating radioactive beams for nuclear physics experiments and medicine applications. The project involves a complex set of systems and machines that require close coordination to ensure the proper functioning of the relevant procedures and the integrity of the machines.

This thesis describes the analysis and development of the coordination and protection system between two subsystems: the vacuum system and the coupling table system. The vacuum system controls several pumps distributed along the line and manages vacuum and venting operations. The Coupling Table manages the operations related to the correct placing of the target chamber to the beam line trunk. Given the frequent need for maintenance of the chamber, ensuring high reliability in coupling and decoupling operations is crucial.

The design foresees that the two subsystems request permission before performing any operation. The handling of requests and acknowledgements was implemented through a Programmable Logic Controller (PLC). The communication with the control system of the whole facility is guaranteed by the adoption of an EPICS (Experimental Physics and Industrial Control System) communication layer based on the MODBUS protocol, which is also used to build the Graphical User Interface (GUI). Finally, automated testing of the software on all possible scenarios has been implemented by exploiting the PLC simulator and a custom python test utility.

Sommario

I Laboratori Nazionali di Legnaro (LNL) stanno sviluppando un progetto denominato Selective Production of Exotic Species (SPES), finalizzato alla generazione di fasci radioattivi per esperimenti di fisica nucleare e applicazioni mediche. Il progetto prevede un complesso insieme di sistemi e macchine che richiedono uno stretto coordinamento per garantire il corretto funzionamento delle procedure e l'integrità delle macchine.

Questa tesi descrive l'analisi e lo sviluppo del sistema di coordinamento e protezione tra due sottosistemi: il sistema del vuoto e il sistema della piattaforma di accoppiamento. Il sistema del vuoto controlla diverse pompe distribuite lungo la linea e gestisce le operazioni di vuoto e di ventilazione. La piattaforma di accoppiamento gestisce le operazioni relative al corretto posizionamento della camera del bersaglio sul troncone della linea del fascio. Data la frequente necessità di manutenzione della camera, è fondamentale garantire un'elevata affidabilità nelle operazioni di accoppiamento e disaccoppiamento.

La progettazione prevede che i due sottosistemi richiedano l'autorizzazione prima di eseguire qualsiasi operazione. La gestione delle richieste e delle conferme è stata implementata attraverso un controllore logico programmabile (PLC). La comunicazione con il sistema di controllo dell'intera struttura è garantita dall'adozione di un livello di comunicazione EPICS (Experimental Physics and Industrial Control System) basato sul protocollo MODBUS, utilizzato anche per realizzare l'interfaccia grafica utente (GUI). Infine, è stato implementato il test automatico del software su tutti gli scenari possibili, sfruttando il simulatore di PLC e un'utility di test python personalizzata.

Contents

List of Figures

Introduction

The National Laboratories of Legnaro (LNL) are an international research centre whose strengths are the development and innovation in the field of particle accelerators, radiation detectors and associated technologies. Here, the Selective Production of Exotic Species (SPES) project is in an advanced state of development, a project that aims to produce high-intensity, high-quality neutron-rich Radioactive Ion Beams (RIBs) and use them for nuclear physics research, astrophysics, medicine and material science.

To protect all the delicate and expensive equipment of the SPES facility a system called Machine Protection System (MPS) is under development. The MPS is designed to block dangerous actions preventing damages and to coordinate the interaction of different subsystems that share a resource or responsibility.

This thesis shows the work done to design and develop a part of the MPS designed to coordinate the Coupling Table (CPTA) system and the Vacuum Control System (VCS). The CPTA manages the precise placing of the target chamber in a bigger apparatus, which houses the first step in the creation of the RIB. The VCS maintains the essential high-vacuum environment within the target chamber. The final goal of this thesis was to create the system in all its various components. The outline of this thesis is described below:

1. Introduction of the SPES project and the role of the MPS in safeguarding its equipment, including an outline of its core functions.
2. Exploration of the technologies used at LNL for its control systems, focusing on the EPICS framework, PLCs and the Modbus protocol.
3. Explanation of the fundamental problem of coordinating the VCS and the CPTA, identifying potential risks of miscoordination and establishing the need for a dedicated system, called Coupling-Vacuum Coordination System (CVCS).
4. Definition of the CVCS requirements and explanation of the use of a Finite State Machine (FSM) to model the system's behavior.
5. Showcase of the technical implementation of the CVCS. This includes the development of the PLC logic, the EPICS server, and the GUI design.
6. Explanation of the testing phase of the CVCS's behaviour.
7. Conclusions and overview of future developments.

Chapter 1

SPES and the Machine Protection System

The LNL are one of the four national laboratories of the Italian Institute of Nuclear Physics (INFN). The LNL are an international center for the design and production of particle accelerators and detectors. Among the many projects, the ambitious SPES project aims to generate and study neutron-rich Radioactive Ion Beam (RIB).

This chapter introduces to the SPES project and the currently developing system that is responsible for the protection of all of its machinery.

1.1 The SPES Project

The SPES project aims to build a facility capable of producing high quality and high intensity neutron-rich RIBs. These beams unlock research opportunities in nuclear physics, astrophysics, medicine and materials science. [1] The main goal of SPES is using the produced RIBs to study exotic isotopes.

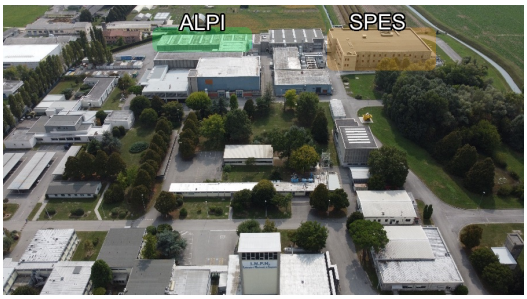


Figure 1.1: *Nationals Laboratories of Legnaro highlighting the ALPI and SPES buildings.*

Another aspect of SPES involves the use of its infrastructures for producing innovative radioisotopes for nuclear medicine. Two core projects carry this goal. The ISOLPHARM project aims to expand the range of available medical isotopes to improve diagnostics and treatment options, leveraging SPES's ISOL (described in 1.1.2) technology to produce radioisotopes with high-specific-activity [2]. The LARAMED project aims to produce medical radionuclides and radio pharmaceuticals directly from the proton beam generated by the SPES's cyclotron. These initiatives hold the potential to significantly impact nuclear medicine by improving disease diagnosis and treatment. [3]

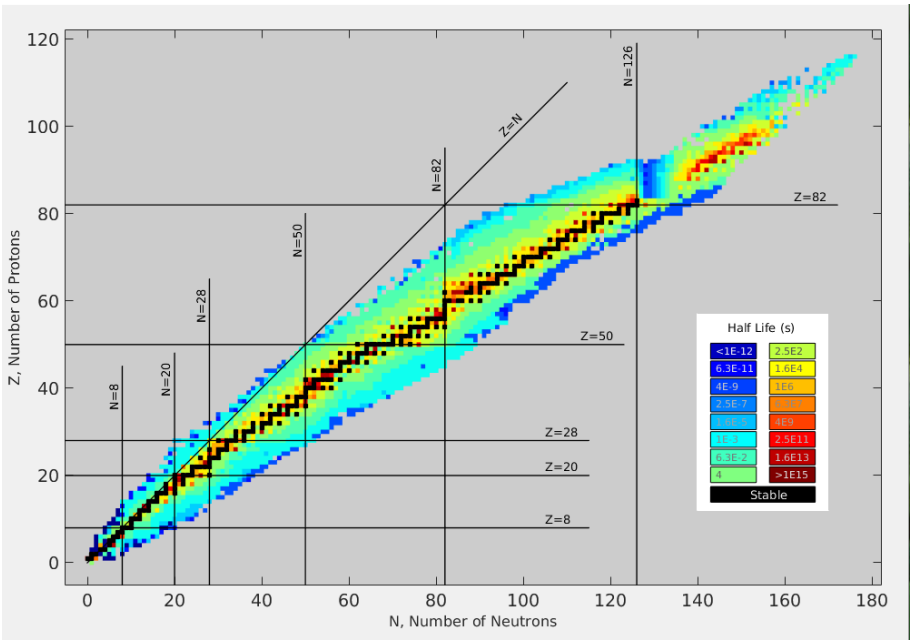


Figure 1.2: *Chart of nuclides highlighting the black "Valley of Stability." Nuclides outside this valley are unstable and undergo radioactive decay. [4].*

1.1.1 Research on the limits of the chart of nuclides

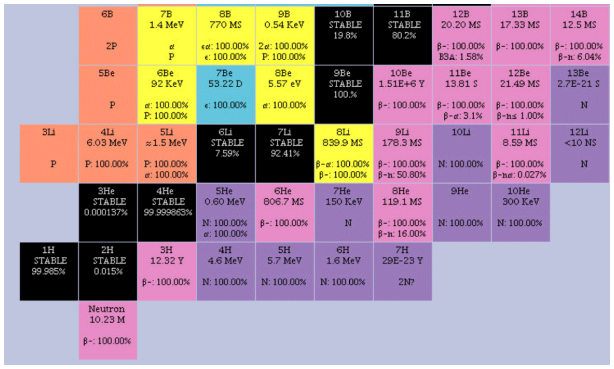


Figure 1.3: *Chart of nuclides, zoomed to the bottom-left corner, containing light isotopes.*

The chart of nuclides (figures 1.2 and 1.3) is the visual representation of all the known atomic nuclei and their isotopes. For physicists it's the equivalent to Mendeleev's periodic table.

Each square represents a distinct nuclide, with the vertical axis (Z) indicating the number of protons (determining the element) and the horizontal axis (N) showing the number of neutrons.

Nuclei are said to be stable if they are resistant to natural decay, and in the chart they appear as black squares and form the "Valley of Stability". Surrounding these nuclei are colored squares representing radioactive isotopes with varying degrees of instability.

A major focus of today's research in nuclear physics focuses on the search and study of exotic nuclides, isotopes with extreme neutron to proton ratios. Studying these isotopes is crucial to better understand the strong force, the fundamental interaction responsible for binding protons and neutrons together, as our current model breaks for nucleids far from stability. [5]

SPES aims to produce and study exotic nuclei near the frontier of instability, a goal shared

with other worldwide facilities. This effort is part of a larger European roadmap leading to a next-generation ISOL facility called EURISOL. [6] The key challenge lies in the instability of radioactive nuclei: the more exotic they are, the shorter their life-times. Facilities like SPES must produce huge amounts of these exotic isotopes and analyze them before they decay.

1.1.2 Production and use of Radioactive Ion Beams

The SPES facility generates high-intensity, high-quality RIBs employing the ISOL technique (fig. 1.4), established today as one of the main methods for isotope production. [7]

This multi-step process begins with a driver accelerator that produces a powerful Primary Protonic Beam (PPB), in the SPES case a cyclotron up to 70 MeV (fig. 1.5). This proton beam is directed towards a specially designed thick target where the impact of the beam induces nuclear fission reactions within the target material, resulting in the creation of a multitude of radioactive isotopes. SPES uses a special target made of Uranium Carbide (UC_x) or Tantalum Carbide (TlC_x) disks (fig. 1.10) housed within a component called Target Ion-Source (TIS) (fig. 1.9). The target can reach temperatures over 2000 °C, this causes the isotopes to be released through evaporation, reaching the ion-source (fig. 1.11) in which they are ionized, forming the initial mixed RIB.

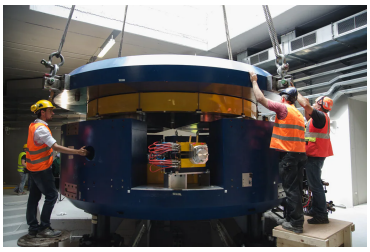


Figure 1.5: *The cyclotron at use at the SPES facility.*

An electrode causes the RIB's particles to exit the TIS and, after some intermediate steps, the production hall (fig. 1.6). If necessary, the RIB can be purified through the High Resolution Mass Separator (HRMS), to filter out the unwanted isotopes with a high degree of success. [1]

The RIB can then either be directed towards an experimental hall for low energy experiments or towards Acceleratore Lineare Per Ioni (ALPI), a linear accelerator that takes the beam to energies up to 10 MeV [9]. Before reaching ALPI, the beam goes through the ADIGE section,

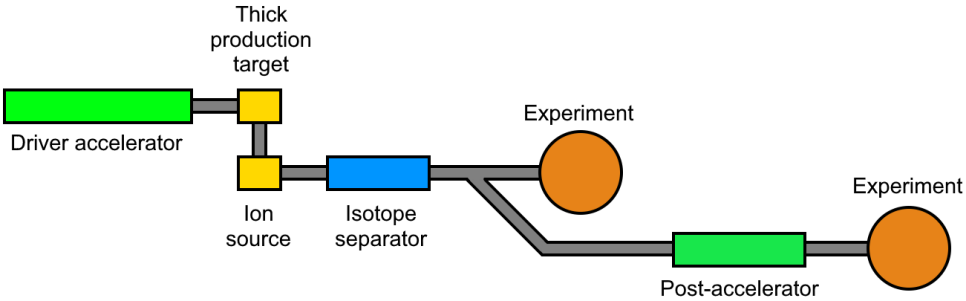


Figure 1.4: *A diagram of the ISOL method, adapted from [8]*

.

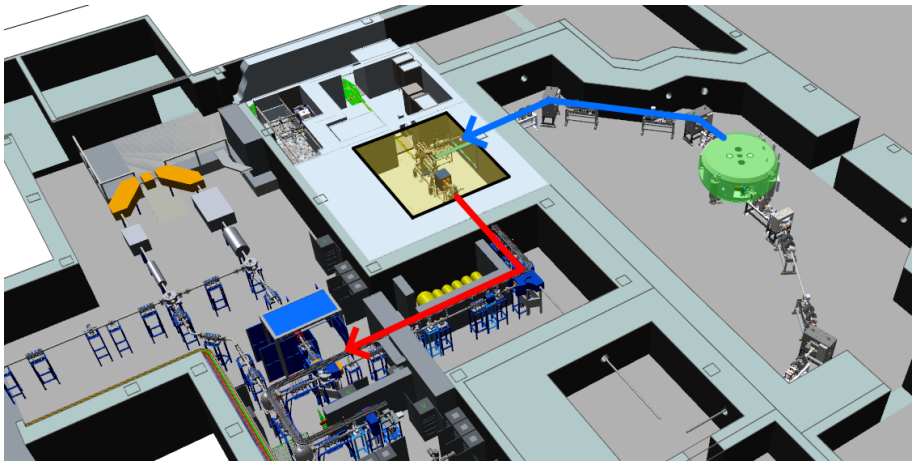


Figure 1.6: *A diagram of initial steps of the ISOL: the cyclotron (green) accelerates the PPB (blue) that goes in to the production bunker (yellow). There the RIB (red) is generated and exits the area.*

that consists of a Charge Breeder and a medium mass resolution separator (MRMS) [10]. The beam also encounters the RFQ (Radio Frequency Quadrupole) injector. This pre-acceleration phase is required to reach the operational conditions of ALPI. The schema of all the components involved in the producion of the RIBs is depicted in figure 1.7.

After the RIB has been accelerated by ALPI, it's ready for experimental use. Experiments involve the RIB colliding with a fixed target inside a reaction chamber in vacuum. The reaction chamber is surrounded by detectors, especially gamma-rays detectors. These detectors pick up gamma rays and particles coming from the reactions. Different detectors are ready for SPES-ALPI radioactive beams.

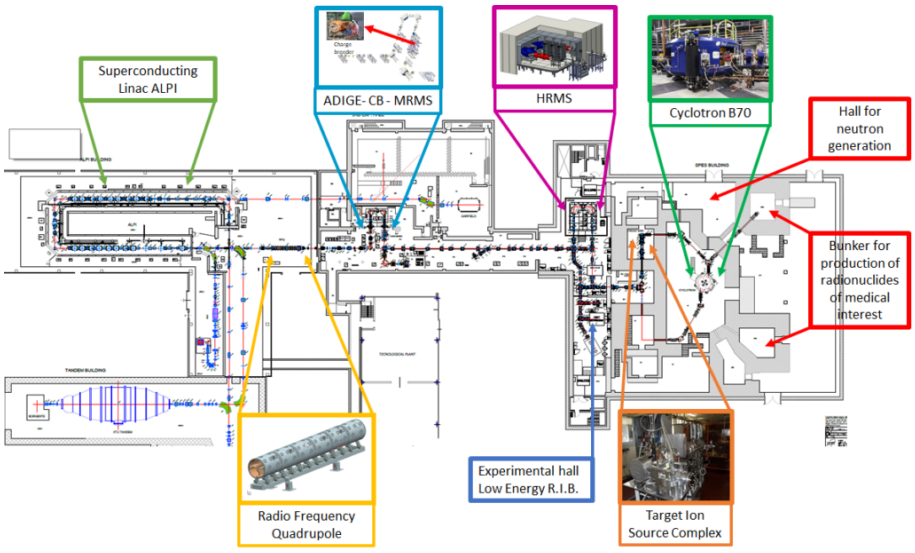


Figure 1.7: *All the components involved in the production of the RIBs.*

An example is AGATA (Advanced Gamma Tracking Array), a highly segmented High Purity Germanium (HPGe) detector array arranged in a spherical structure. This design enables it to capture gamma rays emitted during nuclear reactions. Sophisticated algorithms allow AGATA to trace the paths of gamma rays, revealing their position. AGATA is a European collaboration of 13 countries and it can integrate with complementary detectors, such as PRISMA and EUCLIDES, for deeper insights into the reactions.[11]

1.1.3 The Front-End and the Target Ion-Source chamber

The Front-End (FE) is the apparatus that stores all the main components and systems involved in the RIB production (fig. 1.8). The core of the FE is the Target Ion-Source (TIS) chamber (fig. 1.9), an aluminum structure that houses both the production target (fig. 1.10) and the ion-source (figure 1.11).

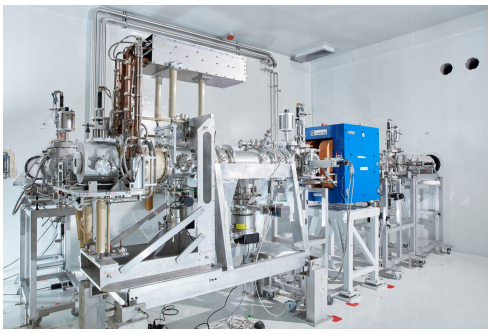


Figure 1.8: *The Front-End apparatus, inside the production hall.*

During RIB production the proton beam impinges on the designed target, increasing the chamber's radioactivity levels and decreasing target and ion-source's efficiency [12]. To account this, SPES operates on a two-week cycle: two weeks of active RIB production followed by a two-week pause. During the pause, the chamber remains idle within the FE, allowing radioactivity levels to decrease. After this period, the chamber is substituted with a new one, ensuring optimal RIB production efficiency.

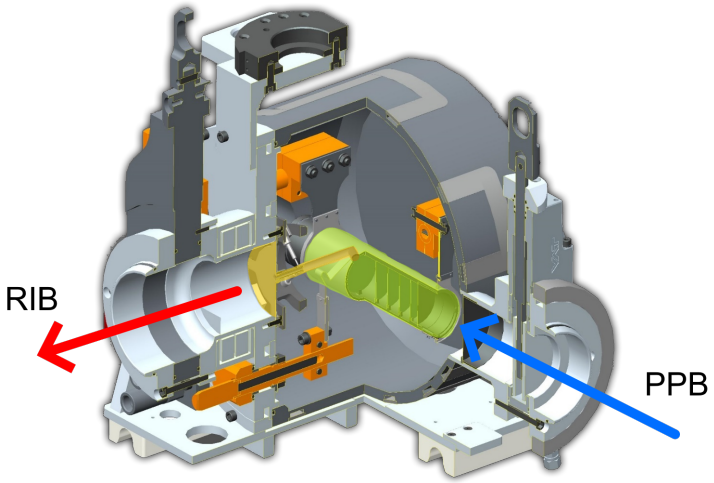


Figure 1.9: *The TIS chamber, where the RIB is formed. The key components are the target (green) and the ion-source (yellow).*

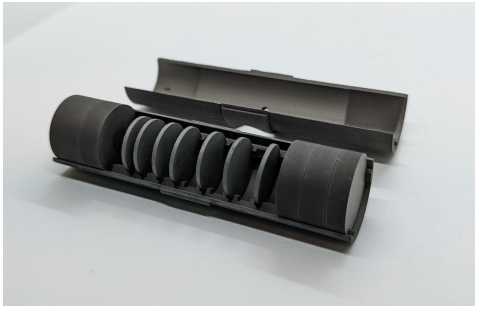


Figure 1.10: *The production thick target, made of Uranium Carbide disks.*

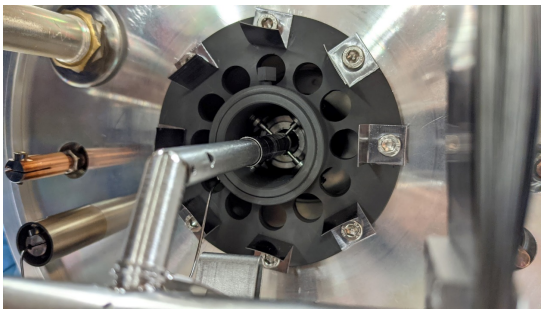


Figure 1.11: *The ion-source, home of the first ionization.*

The installation, movement and storage of the chamber is provided by a series of systems, either automatic or remotely controlled, as manual interactions are not allowed for radioactivity. The two main systems are the Horizontal Handling Machine (HHM) and the Coupling Table (CPTA). The HHM is an automated guided vehicle responsible for moving the chamber around the various rooms. The CPTA is the system that secures the chamber in the FE, and is explained in more details at section 3.1. The list of steps describing the chamber's journey from supply to storage is illustrated in figure 1.12 and described below:

1. An operator manually places the new unit onto the supply point.
2. The HHM retrieves the chamber and brings it to the FE.
3. The CPTA secures the chamber to the rest of the apparatus.
4. The beam is on target for two weeks, increasing its radioactivity.
5. The chamber sits in the FE for two weeks, allowing its radioactivity to decrease.
6. The HHM retrieves it and moves it to a storage room for up to 5 years of decay.

1.2 The Machine Protection System

The SPES facility consists of several expensive and delicate machinery. To ensure the longevity and functionality of all the components, a dedicated Machine Protection System (MPS) is currently being developed. The MPS is designed to carry actions to prevent machine damage and to coordinate complex interactions between subsystems within the SPES facility (fig. 1.13). We can split the responsibilities of the MPS in protection functions and coordination functions [13]:

The protection functions of the MPS guarantee that each device is used without exceeding its operating range. These functions include:

- Monitoring the beam intensity and stopping it if an unsafe environment is detected or if it exceeds safe thresholds.

- Verifying that the target is ready before allowing beam exposure. Parameters may include temperature and pressure.
- Monitoring the water cooling system to protect devices from overheating.
- Block high-voltage if safety parameters are compromised.

The coordination functions of the MPS protect the machine from human errors, blocking actions that would cause damage. Some of the subsystems share common resources and the one's action may preclude another's. In order to prevent these kinds of conflicts a coordination protocol must be implemented by the MPS, enabling each subsystem to operate. When they want to execute a potentially dangerous operation, they must send a request to the MPS, which will process the statuses of all systems and respond to the request, authorizing it or not.

To achieve an industrial level of reliability all the logic must be implemented through Programmable Logic Controllers (PLCs) or hardwired logic. Software installed in general purpose PCs should be avoided to implement critical functions. Every device that implements MPS functions must be powered by an UPS and installed in a technical room that is not directly accessible to non-operators.

1.2.1 Architecture

The MPS is designed with a distributed tree architecture consisting of a central node and some peripheral ones. This architecture allows for a modular design, important for future expansion and scalability as the system grows and new systems are included in the project.

The central node is the heart of the MPS. It houses most of the protection functions and all of the coordination functions. To achieve high availability two redundant PLCs are used in a hot-standby configuration. This means that only one of them is active and sends continuous updates to the passive one. If the active PLC fails, the standby one takes over seamlessly, preventing downtime.

The peripheral nodes are needed in cases where network interfaces are not possible, due to conditions like the presence of high voltage. An independent PLC is responsible for providing basic functionalities and, when possible, to communicate all the needed information with the central node. Two peripheral nodes may not communicate independently from the central node. Peripheral nodes must respect the same protection and reliability requirements of the central node.

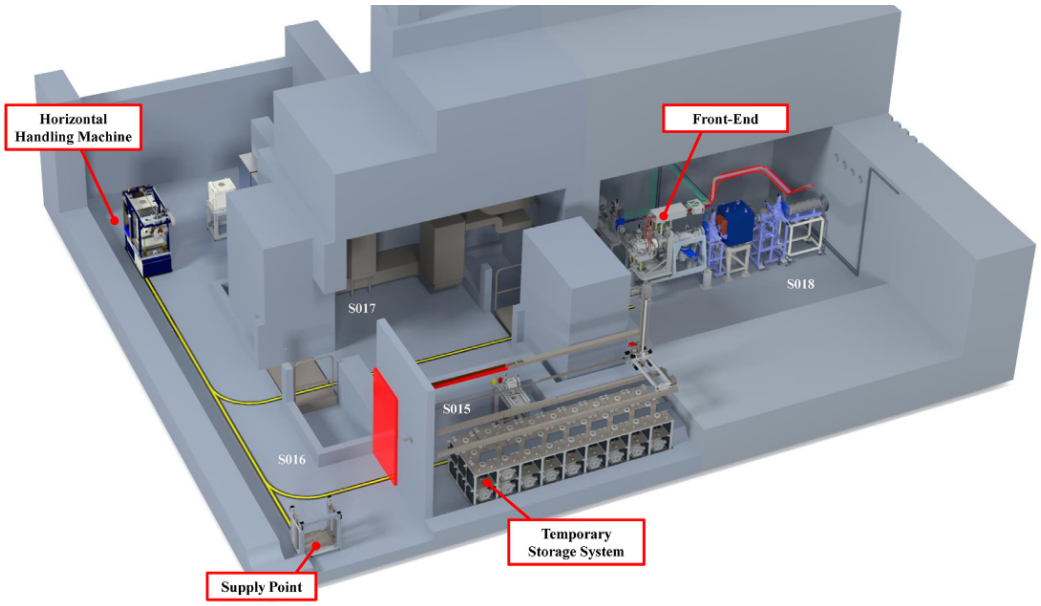


Figure 1.12: *The overall handling system. The HHM follows predesigned paths from the supply, to the FE, to the storage room.*

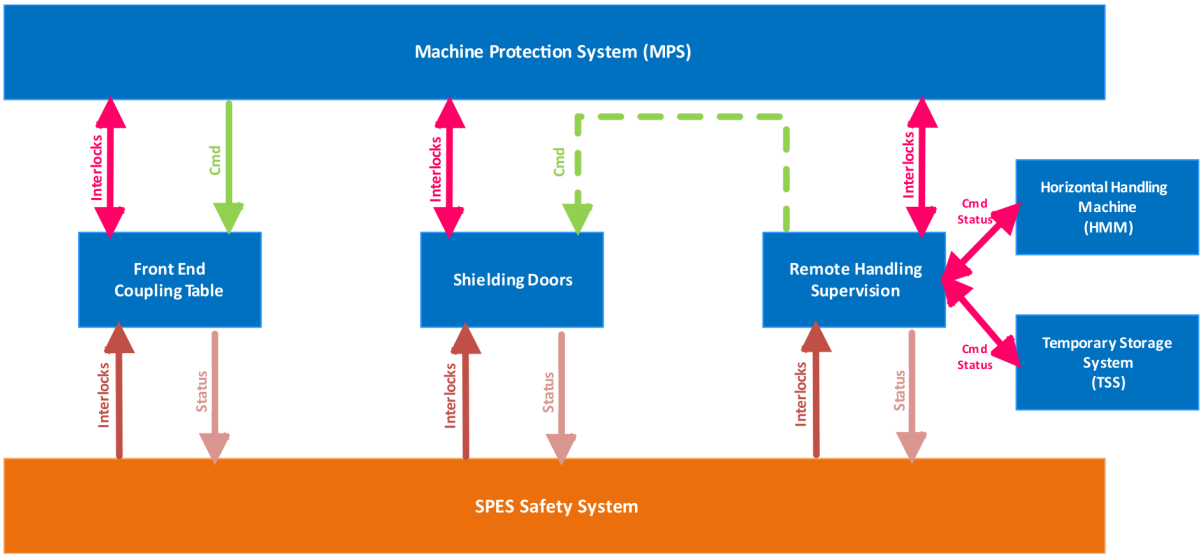


Figure 1.13: *Diagram illustrating communication between the MPS and the handling systems, with the MPS performing coordination functions trough interlocks and commands.*

Chapter 2

LNL Control System Infrastructure

The Legnaro National Laboratories (LNL) relies on a sophisticated control system to manage their complex experimental facilities. This system is composed of several technologies, including the EPICS (Experimental Physics and Industrial Control System) framework, which provides software tools to build a distributed control network, Programmable Logic Controllers (PLCs), offering reliable and real-time control of devices, and industrial communication protocols like Modbus, allowing the integration of PLCs in the EPICS network to remotely controlling their variables and devices.

2.1 Introduction to EPICS

EPICS is a public and open source set of software tools to create distributed control systems, developed by an international collaboration between many scientific facilities. For this reason it is built towards control systems for devices such as particle accelerators, large experiments and major telescopes [14].

Figure 2.1: *The EPICS logo.*

EPICS uses an infrastructure consisting of clients and servers. Servers publish data to the network (usually data retrieved from hardware devices) while clients may access it for storing, displaying or other types of manipulation. These communication follow a specific EPICS protocol called Channel Access (CA), designed for distributed and heterogeneous control systems. This protocol allows EPICS to support up to hundreds of applications running.

EPICS also provide Control System Studio (CSS), a set of tools for building customized graphical user interfaces (GUIs) to manage and control EPICS systems.

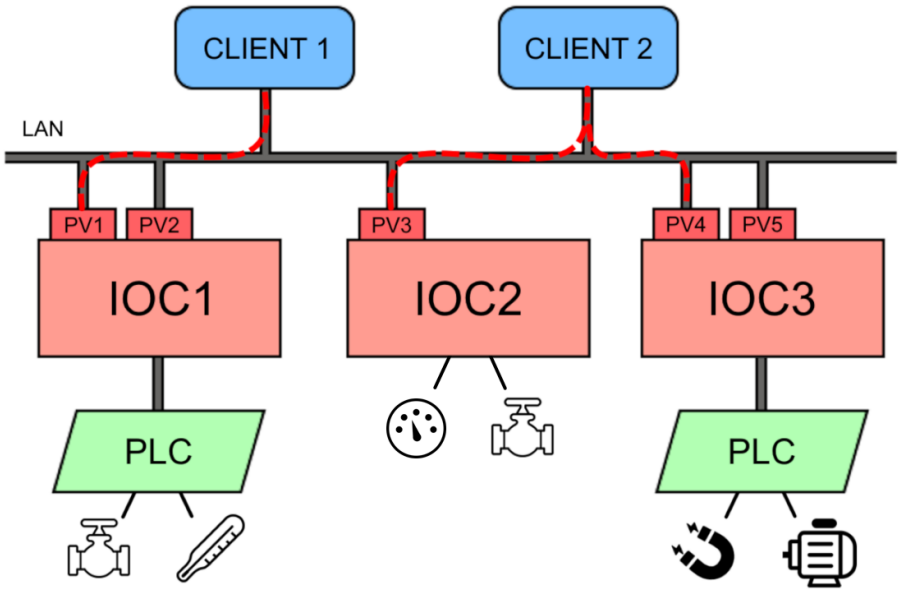


Figure 2.2: *Example of an EPICS control system consisting of three IOCs and two clients, that established a connection to some PVs. IOCs communicate to PLCs to allow remote control of the PLC's connected devices.*

2.1.1 Input-Output Controllers and Process Variables

An EPICS server is called Input-Output Controller (IOC) and it handles the communication between the physical world (sensors, motors, etc.) and the control system's network. The IOCs are the core of every EPICS control system, as they define all the Process Variables (PVs) used by the system in their database.

A PV is a data entity that represents an aspect of the control system. A PV may represents a physical quantity being read (e.g. a voltage, current or temperature) or something more abstract, but still useful for calculation.

PVs are defined as records in the IOC's database using a record type. [15] Some common examples include:

- Analog Input (ai): Retrieves real-number values from hardware inputs.
- Analog Output (ao): Sends real-number values to output devices.
- Binary Input (bi): Similar to ai, for binary (on/off) values.
- Binary Output (bo): Similar to ao, for binary (on/off) values.
- Calculation (calc): Performs operations using values from other PVs.

The record type can be thought as a template containing various fields, that a developer may overwrite, which define the PV's data and behavior. Here's a breakdown of key fields:

- VAL: The primary data value of the Process Variable (temperature, status, etc.).

- SCAN: Determines how often the IOC updates the PV. Can be periodic or event-driven.
- EGU: Engineering Units. Defines the units of measurement for VAL (e.g., °C, V, mA).
- HIGH and LOW: High and low alarm limits which trigger an alarm if exceeded. Variants for higher severities exist.
- STAT and SEVR: Current alarm status and severity.
- HOPR and LOPR: High and Low Operating Range. Provides an operating limit, useful for calibrating the PV visualization.

Listing 1 uses the cited fields to create a PV for displaying temperature. A visual widget linked to this PV would use the defined limits ((15 °C and 30 °C). Additionally, it triggers a low-severity alarm if the `office:temp` value goes above 27°C or below 18°C. The PV would also need to read raw data from a device, a process better explained in chapter 5.

```
record(ai, "office:temp") {
    field(DESC, "Office Temperature")
    field(SCAN, "1 second")
    field(EGU, "°C")
    field(HOPR, 30)
    field(LOPR, 15)
    field(HIGH, 27)
    field(LOW, 18)
}
```

Listing 1: *Example of a simple possible PV for the display of a temperature.*

Clients interact with PVs in various ways: reading for monitoring, writing to control devices, and subscribing for real-time updates. The EPICS system itself processes PVs, potentially associating timestamps, triggering alarms based on limits, or archiving values for historical analysis.

2.1.2 Channel Access

The Channel Access (CA) is the communication protocol used by EPICS that allows clients to remotely access PVs published from IOCs. The protocol tries to minimize overhead information (meaning it maximize the communication of only relevant information are sent), but it still requires a large bandwidth network to manage a high number of IOCs and clients. [16]

Both UDP and TCP messages are used by the protocol:

- Clients request access to a specific PV by broadcasting an UDP message containing the PV name. The server which hosts the PV responds.
- After locating the PV, the client establishes a TCP connection to exchange the PV's data.

2.2 Usage of PLCs

Figure 2.3: *Two Schneider M580 PLCs installed in a rack situated in the SPES building.*

Programmable Logic Controllers (PLCs) are industrial computers specifically design for continuous operations offering a high grade of reliability. They are essential components of industrial control systems, including those at LNL.

PLCs ensure real-time control with precise timing for the managing of processes with safety interlocks. They also are quite easy to integrate in a a network-based control system, like the one produced by the EPICS framework, allowing for remote control of devices connected to the PLCs.

LNL uses PLCs from manufacturers like Schneider Electric (fig. 2.3) and Siemens, and protocols like Modbus and S7 to connect them with the network.

2.3 The Modbus Protocol

Modbus is a widely used industrial communication protocol that lives in the Application layer of the ISO/OSI model. It provides a communication based on service requests from a client to a server. Various versions exist and the most popular is Modbus TCP, which enables the integration of Modbus devices into modern Ethernet-based networks. The base concept of the protocol consists of a request/response paradigm, allowing the client to send requests to the server. The request's core is made of the function code and data. The data is specific to the function code and may contain address, quantity, write value. These together make the Protocol Data Unit (PDU). A header is added for the Application Data Unit (ADU) (fig. 2.4).

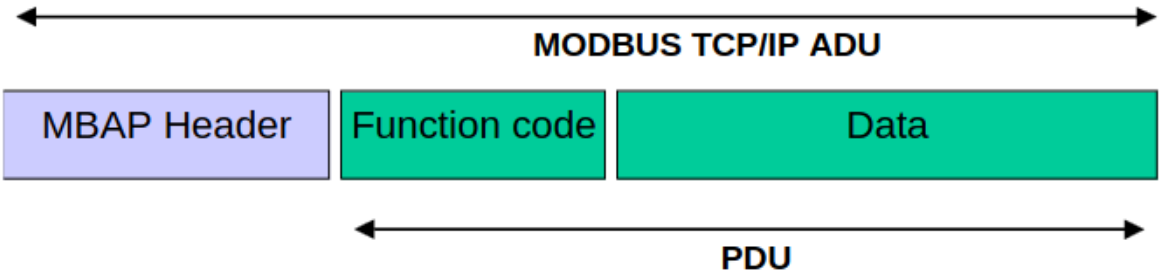


Figure 2.4: *The MODBUS TCP Application Data Unit*

Function codes are 1-byte numeric codes that tell the server what kind of operation to execute. Most codes access data, either reading or writing, but some exist to manage diagnostics,

and some codes are reserved to be customized to user-specific necessity.

The modbus data model allows for the four following data types:

- Discrete Input, a 1 bit read-only data
- Coil, a 1 bit read-write data
- Input Register, a 2 bytes read-only data
- Holding Register, a 2 bytes read-write data

The table 2.1 lists the most used function codes. All of them access data, but they differ in the data type they access. Note that there are no codes to write Discrete Input and Input Registers as these types are read-only.

Code	**Name**	**Description**
0x01	Read Coils	Reads the status (on/off) of multiple output addresses.
0x02	Read Discrete Inputs	Reads the status (on/off) of multiple addresses.
0x03	Read Holding Registers	Reads the contents of multiple contiguous addresses.
0x04	Read Input Registers	Reads the contents of multiple contiguous addresses.
0x05	Write Single Coil	Sets the value (on/off) of a single address.
0x06	Write Single Register	Sets the value of a single address.

Table 2.1: *Example Modbus Function Codes and Descriptions*

Chapter 3

The Coupling-Vacuum Coordination Problem

One of the many challenges within the MPS arises from the need for coordination of the operations of the Coupling Table (CPTA) and the Vacuum Control System (VCS). This chapter describes these systems and explains the details of problem.

3.1 Coupling Table Control System

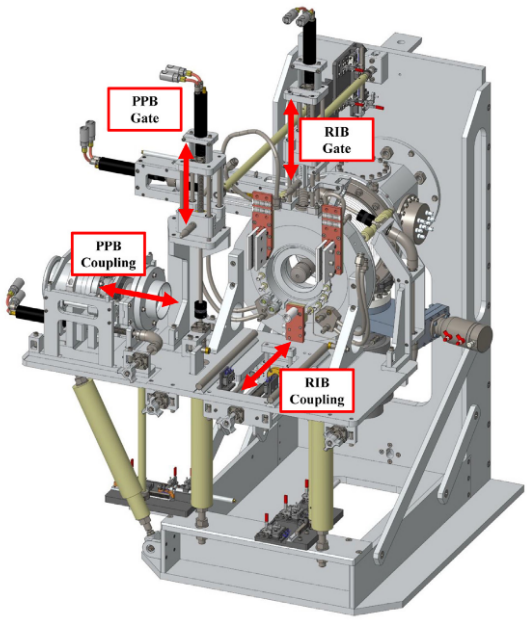


Figure 3.1: *The coupling table and its four motors. Two controlling the coupling, two controlling the gate valves.*

The CPTA is part of the Front-End (FE) and its control system is responsible for the coupling and decoupling operations of the TIS chamber to the protonic channel (that carries the PPB) and the radioactive channel (where the RIB is sent to). The CPTA provides stable connections to ensure vacuum, electrical supply, water cooling, gas and signal exchange to the TIS Chamber. The system is part of a larger handling system that operates to safely remove, relocate and install TIS chambers according to their life cycle, briefly described in 1.1.3. [12]

The components of the CPTA were chosen to be resistant to the radiation and include: four pneumatic motors, four electrovalves (which control the pneumatic motors), limit switches, two redundant potentiometers and a resolver.

A dedicated Schneider M580 PLC [17] implements the logic that enables the chamber to be moved between the limit switches, controlling the electrovalves. It's also possible to specify positions using the potentiometer readings. The resolver provides additional position feedback. The coupling and uncoupling procedures are automated in the PLC, which includes protection interlocks in case of unexpected changes in the operating conditions. A GUI and EPICS IOC have also been developed, providing remote access to the PLC's variables. [18]

The coupling procedure starts after the chamber is positioned in the CPTA by the HHM (see section 1.1.3) and operates as follow:

1. The chamber is secured to the RIB line.
2. The chamber is coupled with PPB line.
3. The two TIS chamber gate valves are opened to allow air flow.

3.2 Vacuum Control System

The VCS is a distributed system that manages the pressure levels along all the line in the SPES facility. Only recently, in 2022, the system was upgraded to use the EPICS framework [19]. This section provides a general overview of the whole system, the next section will explain in more detail the case of the system controlling the TIS area.

The beam lines need to meet a required vacuum level to lower the probability of beam particles crashing into air particles. In order to do this the beam line is divided in many vacuum trunks by gate valves, which may be activated to control the vacuum level of a section.

Every trunk has at least one of each of the following devices:

- Pumps, of which two types are used
  - Primary pumps, which can get the trunks from atmospheric pressure to low vacuum.
  - Turbopumps, which can get the trunks from low to high vacuum.
- Gate Valves, which control flow between trunks and other pipes.
- Pressure gauges, to monitor pressure levels in various parts of the system. There are different types for different sensitivities.

The VCS must manage pressure changes across different sections of the beam line with caution, as opening valves connecting areas with significant pressure differences can create turbulence, potentially damaging equipment or disrupting the vacuum.

The control of all the trunks is divided among a multitude of independent Siemens PLCs, each controlling roughly the same amount of devices. Since all the PLCs control the same types of devices in similar structures, a single configurable code base has been developed and each

PLC may configure it following the specific structure of its trunks. These PLCs communicate with EPICS IOCs to allow remote control from CSS GUIs.

The VCS's procedures are described below:

1. Fore-vacuum. Brings the trunk from atmospheric pressure to a low vacuum level. It does so activating the primary pumps.

2. Vacuum. This procedure executes after the fore-vacuum and brings the trunk to a high vacuum level using Turbopumps, which are then kept active until the stop of the operation.

3. Stop. Brings the system into a safe state. The command is also propagated to the connected pumps.

4. Venting. This is done by opening, briefly but continuously, gate valves connected to atmospheric pressure pipe to let the air flow enter without much turbulence.

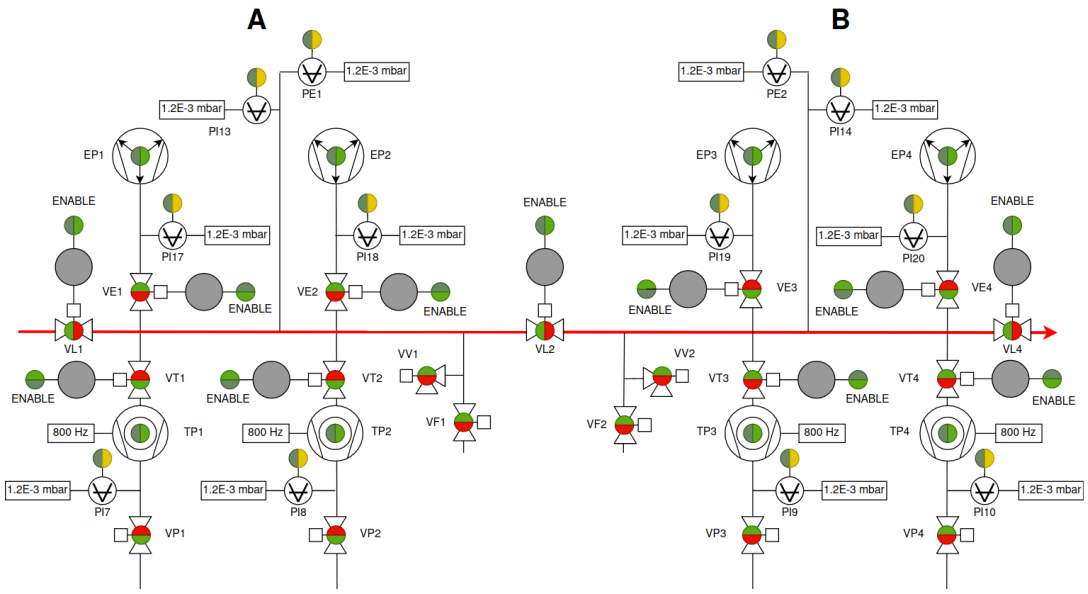


Figure 3.2: *The GUI used by one of many vacuum PLCs, composed of valves, gauges and pumps.*

3.3 The need for coordination

Both the CPTA and the VCS share the responsibility of the vacuum level within the TIS chamber. The diagram in figure 3.3 details the situation:

- G1 and G2 are two gate vales near the TIS controlled by the CPTA (visible from fig. 3.1).
- V1 and V2 are two other gate valves further apart, controlled by the VCS.

The VCS can regulate pressure within the trunk of the chamber thanks to the pumps and the gauges connected between G2 and V2, but it can't control the chambers's vacuum level directly and independently, as the system doesn't have the capacity to see if G1 and G2 are open or to control their opening and closure. To do this it must coordinate with the CPTA.

Furthermore, starting the fore-vacuum and vacuum with the chamber uncoupled (and G1 and G2 closed) makes the coupling operation dangerous. Because opening the gates would cause turbulence due to the pressure difference between the atmospheric-pressure chamber and the vacuum in the trunk.

This is a coordination function which must be implemented within the MPS to avoid the direct communication of the systems, following the MPS architecture. The following chapters follow this system's phases of design, development and testing.

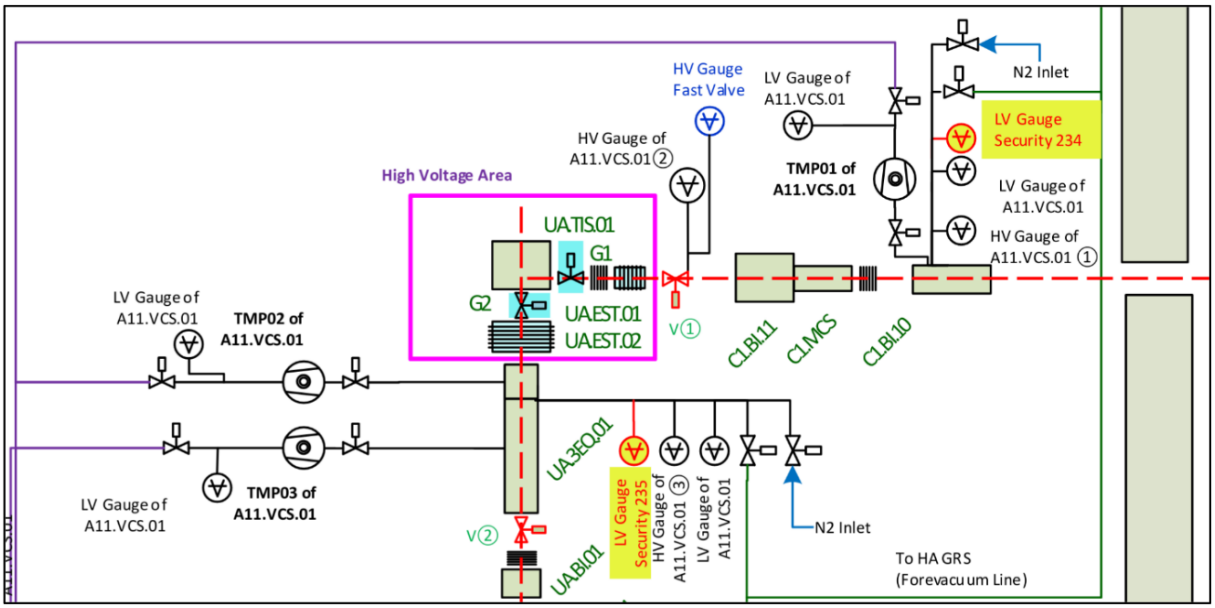


Figure 3.3: *The vacuum scheme of the TIS. The TIS is inside the High Voltage Area. G1 and G2 are highlighted in blue. V1 and V2 are along the red line. Pumps and gauges are connected to the section between G2 and V2.*

Chapter 4

Design

The work of this thesis starts with the design of the Coupling-Vacuum Coordination System (CVCS). This chapter shows the steps I went trough to analyze the problem.

4.1 Requirements

Functional requirements for the Coupling-Vacuum Coordination System (CVCS) were decided by the LNL's MPS and control teams and provide the framework for my work on the project. They are outlined in an internal document describing the MPS functions. I'm summarizing the key elements here, integrating with the results of various meetings:

1. The CVCS needs to be able to receive a request signal and check for cohesion in the CPTA and VCS' statuses.
    - Coupling requires an uncoupled chamber and atmospheric pressure in the trunk (From now on, we call "venting" this trunk status).
    - Uncoupling requires a coupled chamber and a venting trunk.
    - Vacuuming requires a coupled chamber and a venting trunk.
    - Venting requires a coupled chamber and vacuum in the trunk.
2. The CVCS needs to raise an ACK if the requested operation can be started.
3. The CVCS may not raise an ACK if another operation is in progress.
4. The CVCS must detect falling requests and verify successful operation completion
5. If an operation fails, the CVCS must halt safely and wait for a reset.
6. If the CVCS halts suddenly, the ACK has to be lowered and a safe state must be ensured.
7. If a halt occurs during an operation, the CVCS' next reset needs to resume the operation. The CVCS may not accept any request other than the one that caused the error until the correct operation's completion.

Other non functional requirements were decided and used in the development process

1. The CVCS' GUI shall:
   (a) show the statuses of the CVCS and the subsystems
   (b) show the acknowledgments generated by the CVCS
   (c) Allow the operator to reset error states
   (d) Allow to send operational requests
2. The CVCS shall contain most of the logic in the PLC, ensuring reliability and allowing for automatic testing.

4.2 Coordination protocol

The goal of the CVCS is to coordinate the CPTA and the VCS. In order to do this a communication protocol had to be designed and implemented. The protocol is based on a transition sequence of requests and acks (fig. 4.1), and works as follow:

1. One of the two subsystems raise a request signal to to the CVCS.
2. The CVCS receives the request and, if there are the right conditions, it acknowledges it. The subsystem shall not operate without the CVCS permission.
3. The subsystem starts the operation until it finishes, then it lowers the request signal.
4. The CVCS sees that the request fell, and lowers the acknowledgment signal.

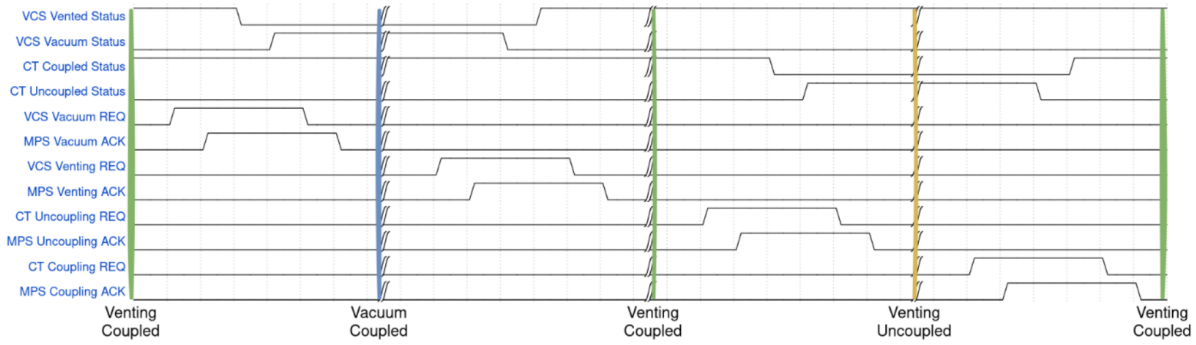


Figure 4.1: *Diagram showing the CVCS' expected signals for statuses, requests and responses.*

4.3 Architecture and Finite State Machine

A quick analysis of the requirements reveals that the CVCS must track the states of its coordinated subsystems. Both the CPTA and the VCS possess only two possible states, and the number

of their operations is limited. Also, every operation's output is deterministic, meaning there's only one possible next state. For example, the Coupling operation always brings the CPTA from uncoupled to coupled (provided there are no errors).

We can describe this behavior of the system as a Finite State Machine (FSM). This formal mathematical model describes dynamic systems with a finite number of states and inputs, of which the output is deterministic.

A Finite State Machine is formally defined with 5 parameters:

- The set S of all the states of the system
- The set I of all the possible inputs
- A function f that given a state and an input, gives the output state.
- The initial state s_0
- The set $F \subset S$ of the final states

In the case of the CVCS, the parameters were firstly defined as:

- S = {Uncoupled, Coupled, Ready, Reset, Halt}
- I = {Couple, Uncouple, Vacuum, Vent, Error, Reset}
- The function f is represented as the table 4.1. The columns represent the input, while the rows represent the state of the system. The intersection shows the next state the FSM will meet. A dashed line (-) means the input has no effect and the state is not changed.
- s_0 = Reset
- F = {}, as the system is designed for continuous operation

	Uncoupled	**Coupled**	**Ready**	**Halt**	**Reset**
Couple	Couple	-	-	-	-
Uncouple	-	Uncoupled	-	-	-
Vent	-	-	Coupled	-	-
Vacuum	-	Ready	-	-	-
Halt	Halt	Halt	Halt	-	Halt
Reset	Reset	Reset	Reset	Reset	-

Table 4.1: *Function f. Columns are the state, rows are input operations.*

The Reset state is a special state that operates reading the statuses of the CPTA and the VCS: if the chamber is coupled and in vacuum the state is set to Ready, if it's coupled and venting the state is set to Coupled, if it's uncoupled is set to the Uncoupled state.

During further analysis emerged the need to promote the four operations of the subsystems from FSM inputs to states. Figure 4.2 depicts the state diagram resulting from this model.

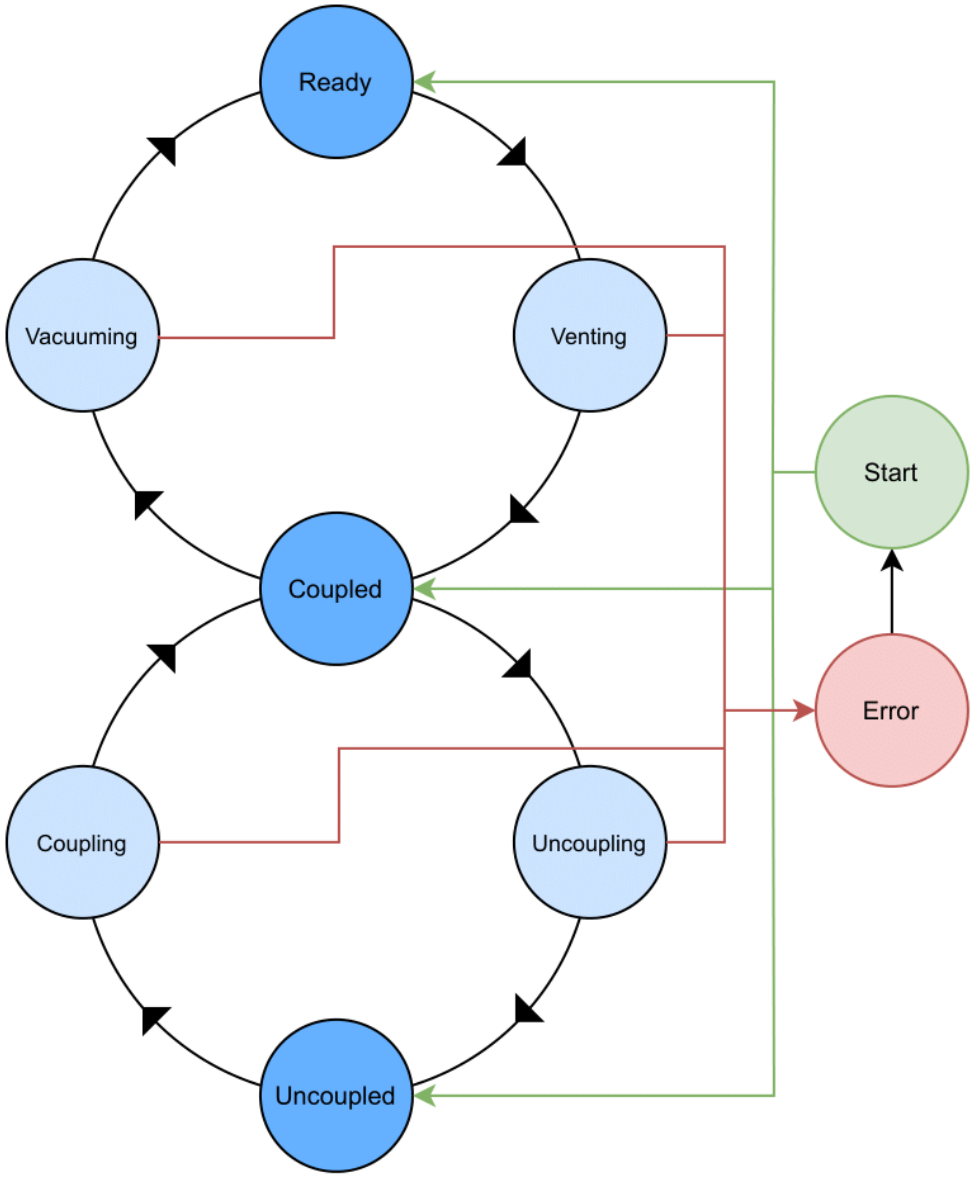


Figure 4.2: *State diagram of the CVCS FSM. The start state was later renamed to Reset, and the Error state to Halt. The four light blue states represent the transitions, while the three dark blue states represent the idle states.*

Chapter 5

Development

After the design phase, I started the CVCS development. This chapter explains the implementation of the system's various components and how they work together to achieve the desired coordination and requirements.

5.1 Development process and environment

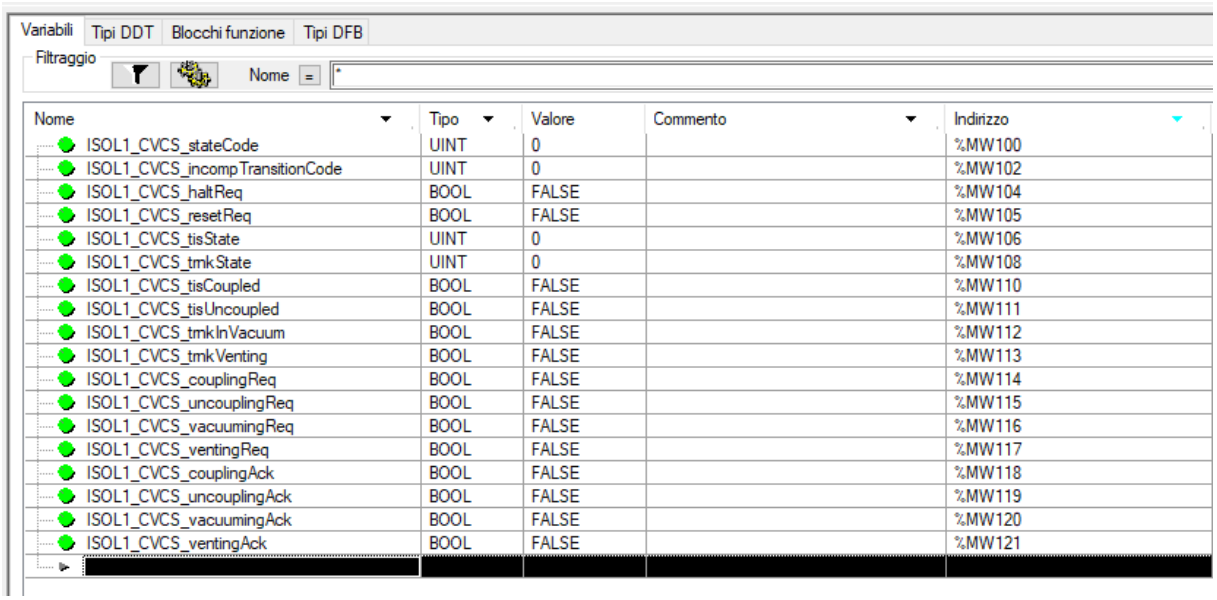
The CVCS consists of three core components: the main logic implemented in a PLC, the EPICS server to expose the PLC variables to the network, and the EPICS client to access the variables from a graphical user interface (GUI). The development of all the parts involved a variety of software programs and a cyclical review to make sure every component sends and receives correct information to each other.

The development happened in a Linux environment, a standard choice for technical projects. Linux offers many utilities, particularly so for EPICS and networking. This allowed for seamless integration of the CVCS components, from the control logic to the user interface.

The PLC programming was done in Control Expert, a proprietary software by Schneider Electrics available only for Windows. The software allowed me to simulate a PLC and run it, but it required the installation of windows virtual machine connected in NAT to the main linux PC, where the IOC was running.

The development of the IOC involved EPICS' shell commands and Visual Studio Code. The entire IOC's development was managed under version control using Git.

The GUI was developed in Control System Studio (CSS) in its newest variant Phoebus. CSS is an open source software that provides a drag-and-drop environment for creating and designing the interfaces. The program has a variety of widgets which can be linked to EPICS PVs to let the user monitor and eventually modify them, covering many possible control-based use cases.



Nome	Tipo	Valore	Commento	Indirizzo
ISOL1_CVCS_stateCode	UINT	0		%MW100
ISOL1_CVCS_incompTransitionCode	UINT	0		%MW102
ISOL1_CVCS_haltReq	BOOL	FALSE		%MW104
ISOL1_CVCS_resetReq	BOOL	FALSE		%MW105
ISOL1_CVCS_tisState	UINT	0		%MW106
ISOL1_CVCS_tmkState	UINT	0		%MW108
ISOL1_CVCS_tisCoupled	BOOL	FALSE		%MW110
ISOL1_CVCS_tisUncoupled	BOOL	FALSE		%MW111
ISOL1_CVCS_tmkInVacuum	BOOL	FALSE		%MW112
ISOL1_CVCS_tmkVenting	BOOL	FALSE		%MW113
ISOL1_CVCS_couplingReq	BOOL	FALSE		%MW114
ISOL1_CVCS_uncouplingReq	BOOL	FALSE		%MW115
ISOL1_CVCS_vacuumingReq	BOOL	FALSE		%MW116
ISOL1_CVCS_ventingReq	BOOL	FALSE		%MW117
ISOL1_CVCS_couplingAck	BOOL	FALSE		%MW118
ISOL1_CVCS_uncouplingAck	BOOL	FALSE		%MW119
ISOL1_CVCS_vacuumingAck	BOOL	FALSE		%MW120
ISOL1_CVCS_ventingAck	BOOL	FALSE		%MW121

Figure 5.1: *PLC variables, following the MPS internal naming convention.*

5.2 Finite State Machine code in PLC

The first step was to develop the Finite State Machine (FSM) designed in 4.3. The main logic of the CVCS was written as a PLC program, as PLCs offer industrial-level reliability, and this also ensures consistency with other control systems within the SPES facility, simplifying maintenance and integration.

First, I defined all the essential variables (fig. 5.1). These variables include the procedure's requests and acknowledgments, and the states of the chamber for its placement (coupled / uncoupled) and its pressure levels (venting / vacuum). During development emerged the need of some more for internal logic, like the variable `incompTransitionCode`, needed for correctly resetting the system after a halt during an operation.

The program is divided into one task and one section for each state of the FSM. All the code is written in Structured Text (ST), a standard high-level programming language for PLCs.

The task contains the main logic of the FSM: it checks the value of the current state and runs the corresponding procedure (table 5.1). It also initializes the triggers that signal when a request rises or falls, important information accessed by the sections.

State Code	PLC Procedure
0	ISOL1_CVCS_resetState()
10	ISOL1_CVCS_readyState()
20	ISOL1_CVCS_coupledState()
30	ISOL1_CVCS_uncoupledState()
40	ISOL1_CVCS_transitionCoupling()
50	ISOL1_CVCS_transitionUncoupling()
60	ISOL1_CVCS_transitionVacuuming()
70	ISOL1_CVCS_transitionVenting()
400	ISOL1_CVCS_haltState()

Table 5.1: *PLC Procedures and corresponding state codes.*

The task also contains the code to detect halt requests at every cycle (Listing 2). `RISING_HALT()` is a function block that detect is the variable containing the halt requests went from 0 to 1 in the last cycle. This means that the code is only executed once at every halt request.

```
1   IF RISING_HALT.Q THEN (* Force Stop *)
2       ISOL1_CVCS_couplingAck := 0;
3       ISOL1_CVCS_uncouplingAck := 0;
4       ISOL1_CVCS_vacuumingAck := 0;
5       ISOL1_CVCS_ventingAck := 0;
6       ISOL1_CVCS_haltReq := 0;
7
8       (* If we stopped during a transition *)
9       IF ISOL1_CVCS_stateCode >= 40 AND ISOL1_CVCS_stateCode <= 80 THEN
10              ISOL1_CVCS_incompTransitionCode := ISOL1_CVCS_stateCode;
11      END_IF;
12
13      ISOL1_CVCS_stateCode := 400;
14  END_IF;
```

Listing 2: *The part of the main task responsible for halting the system.*

Every section contains code specific to each state. The three idle state's sections check for rising requests. Following is an example of the code of the Coupled state, which detects vacuuming and uncoupling requests (Listing 3). In case of both arriving together, a second condition was inserted, to ensure that one and only one can ever be acknowledged. The code for the Uncoupled and Ready state is similar.

The four transition state's sections (uncoupling, coupling, vacuuming, venting) all contain three core phases. First, the code checks that the states didn't changed without the ACK. Then

```
    (* State changed without permission*)
    IF ISOL1_CVCS_tisState <> 1 OR ISOL1_CVCS_trnkState <> 2 THEN
        ISOL1_CVCS_stateCode := 400;

    ELSIF RISING_VACUUMING.Q AND NOT ISOL1_CVCS_uncouplingReq THEN
        ISOL1_CVCS_vacuumingAck := 1;
        ISOL1_CVCS_stateCode := 60; (* Enter Transition Vacuuming *)

    ELSIF RISING_UNCOUPLING.Q AND NOT ISOL1_CVCS_vacuumingReq THEN
        ISOL1_CVCS_uncouplingAck := 1;
        ISOL1_CVCS_stateCode := 50; (* Enter Transition Uncoupling *)

    END_IF;
```

Listing 3: *The PLC's procedure for the Coupled state.*

it checks if the request fell and if the state changed correctly. Finally it gives the possibility to resume an halted operation, looking for the rise of its request. In Listing 4 is shown the core of the Coupling section.

```
    (* End Active Transition *)
    IF FALLING_COUPLING.Q AND ISOL1_CVCS_couplingAck = 1 THEN
        ISOL1_CVCS_couplingAck := 0;

        IF ISOL1_CVCS_tisState = 1 THEN (* Coupled *)
                ISOL1_CVCS_stateCode := 20; (* go to Coupled State *)
        ELSE
                IF ISOL1_CVCS_tisState = 0 THEN (* Still in transition *)
                        ISOL1_CVCS_incompTransitionCode := ISOL1_CVCS_stateCode;
                END_IF;
                ISOL1_CVCS_stateCode := 400; (* Final state not as planned *)
        END_IF;
    ELSIF RISING_COUPLING.Q THEN
        ISOL1_CVCS_couplingAck := 1;
    END_IF;
```

Listing 4: *A snippet of the PLC's procedure for the uncoupling transition state.*

The reset section checks the states of the chamber and the trunk and sets the correct state code (e.g. if the TIS is coupled and the trunk is at atmospheric pressure, the FMS is set to the coupled state). If the trunk or the TIS are still in transition, it returns to the incomplete transition's state (Listing 5).

```
    ...
    ELSIF (ISOL1_CVCS_tisState = 0 XOR ISOL1_CVCS_trnkState = 0)
        AND ISOL1_CVCS_incompTransitionCode > 0 THEN
        ISOL1_CVCS_stateCode := ISOL1_CVCS_incompTransitionCode;
        ISOL1_CVCS_incompTransitionCode := 0;
    ELSE
        ISOL1_CVCS_stateCode := 400; (* go to halt State*)
    END_IF;
```

Listing 5: *Final lines of the PLC reset state procedure.*

The halt section checks if a reset request is raised and set the corresponding state code.

5.3 EPICS Process Variables

Once the control logic was defined, the next step was the development of the IOC server to allow the reading and writing of values on the PLC. I firstly created the empty IOC using an EPICS template. To connect the IOC to the PLC via Modbus TCP protocol, I installed and used the 'modbus' module for EPICS. I then added the following lines in the startup script (Listing 6).

```
drvAsynIPPortConfigure("CVCS_PLC", "192.168.122.231:502", 0, 0, 1);
modbusInterposeConfig("CVCS_PLC", 0, 2000, 0);

drvModbusAsynConfigure("MpsRd", "CVCS_PLC", 0, 3, 100, 10, 0, 100, "M580");
drvModbusAsynConfigure("MpsWr", "CVCS_PLC", 0, 6, 100, 10, 0, 100, "M580");
drvModbusAsynConfigure("StatWr", "CVCS_PLC", 0, 6, 110, 4, 0, 100, "M580");
drvModbusAsynConfigure("ReqWr", "CVCS_PLC", 0, 6, 114, 4, 0, 100, "M580");
drvModbusAsynConfigure("AcksRd", "CVCS_PLC", 0, 3, 118, 4, 0, 100, "M580");
```

Listing 6: *Lines of the IOC's startup script for connection and port definition.*

The first two lines define the TCP/IP port used to communicate with the PLC. The port 502 is the standard port reserved to the modbus protocol.

The other lines configure the modbus requests that will be used to access the PLC's memory. The important highlights are the second, third and fourth number. They are, in order, the function code, the initial memory address and the length.The only function codes used are 0x03 and 0x06, which are the function to read and write Holding Registers. Even if some of the values are boolean and 1 bit would be sufficient, a Holding Register is used to lower the code complexity and avoid sending a network packet for every bit.

The creation of the PVs involved the definition of some templates. A template in EPICS is a record containing some macros. The macros are later substituted and for every substitution a different PV is created. This allowed me to save time defining less records.

Listing 7 shows the record template for the Acknowledgement PVs. The Macro `$(PORT)` is then substituted with `AcksRd`. Every PV's that uses the template define a different `$(OP)` and `$(OFF)` Macro, defining the operation for which the PV ack is created, and the exact address on the PLC.

```
record(bi, "$(FAC)$(APP)$(ELM):$(OP)Ak") {
    field(DESC, "$(DESC)")
    field(DTYP, "asynUInt32Digital")
    field(INP, "@asynMask($(PORT),$(OFF), 0xFFFF, 0) UINT16")
    field(VAL, "0")
    field(SCAN, ".5 second")
    field(ZNAM, "NOT ACK")
    field(ONAM, "ACK")
    field(PINI, "YES")
}
```

Listing 7: *Example of the records used for the acknowledgment signals of all operations.*

Other interesting PVs are related to the status of the CVCS (Listing 8), these two PVs form a chain that allows for a human-readable status:

1. The ai record is defined for reading the PLC every 0.5 seconds. Its 'FLNK' field connects it to the next PV in the chain, the mbbi record.

2. The mbbi record takes the raw number from the ai record and matches it to a descriptive status message. This makes the status easy to understand when displayed on a screen or printed in a report.

```
record(ai, "$(FAC)$(APP)$(ELM)-Stat") {
   field(DESC, "Read state of the coordination")
   field(SCAN, ".5 second")
   field(DTYP, "asynInt32")
   field(INP,  "@asyn($(PORT_RD), 0) INT32_LE")
   field(PINI, "YES")
   field(FLNK, "$(FAC)$(APP)$(ELM):Stat")
}

record(mbbi, "$(FAC)$(APP)$(ELM):Stat") {
   field(DESC, "State of the coordination")
   field(INP,  "$(FAC)$(APP)$(ELM)-Stat")
   field(DTYP, "Raw Soft Channel")
   field(ZRST, "Resetting")              field(ZRVL, "0")
   field(ONST, "Coupled and in vacuum")  field(ONVL, "10")
   field(TWST, "Coupled and venting")    field(TWVL, "20")
   field(THST, "Uncoupled and venting")  field(THVL, "30")
   field(FRST, "Coupling")               field(FRVL, "40")
   field(FVST, "Uncoupling")             field(FVVL, "50")
   field(SXST, "Vacuuming")              field(SXVL, "60")
   field(SVST, "Venting")                field(SVVL, "70")
   field(EIST, "Halt")                   field(EIVL, "400")
}
```

Listing 8: *Example of the Dl1MpcsCvcs:Stat and Dl1MpcsCvcs-Stat Process Variables.*

5.4 Graphical User Interface

A key deliverable of this thesis is the system's Graphical User Interface (GUI). It is fundamental that all the states and errors are easily readable by the operator. The current GUI (figure 5.2) is composed of two areas: to the left the area dedicated to the CVCS, to the right an area used to simulate the statuses and the requests of the two subsystems. The slide buttons of the subsystems' statues are toggled manually for testing purposes and won't be included in the final delivery of this interface. The CVCS GUI hosts the following components:

- A main label containing the state of the system.
- Buttons to Halt and Reset the system.
- Two labels which read the states of the subsystems.
- Four LEDs to read the ACKs sent by the CVCS
- An area containing a brief error message.

The GUI shows the operation that is being performed and subsystem's states (figure 5.3). If

an error occurs during an operation, the GUI shows the error state and a concise message (figure 5.4).

Resetting the CVCS after an operational halt restores the system to the operation state which was halted (figure 5.5). The requirements say that at this point the request must be sent once more, and the GUI indicates that a new request is needed by keeping the Acknowledgment LED off. After the new request the subsystem's status can be changed, the request can be lowered and the operation can successfully end.

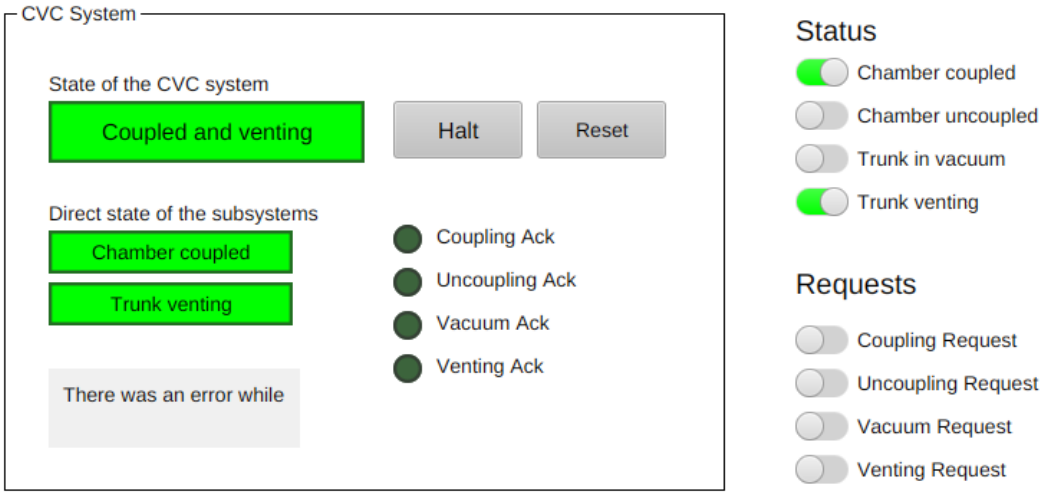


Figure 5.2: *The CVCS GUI at idle state.*

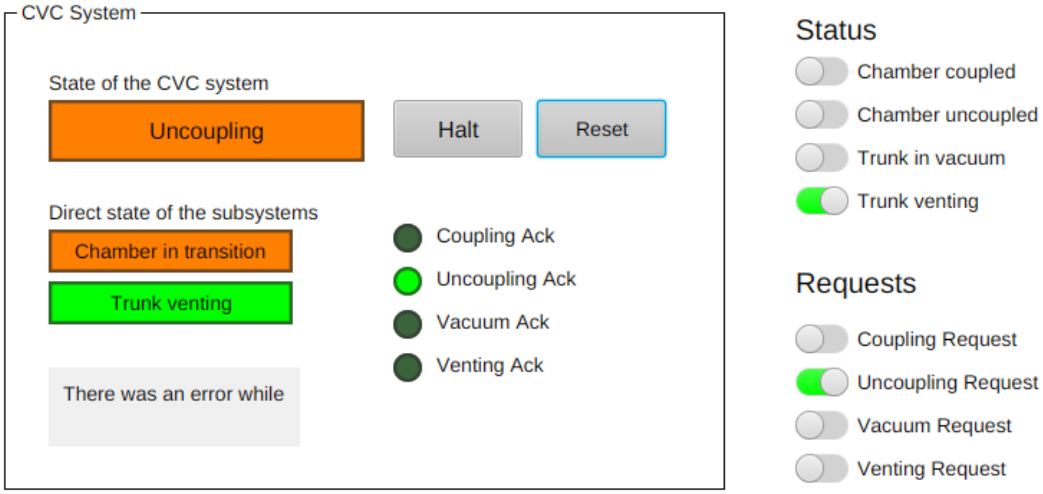


Figure 5.3: *The CVCS GUI when the system is uncoupling.*

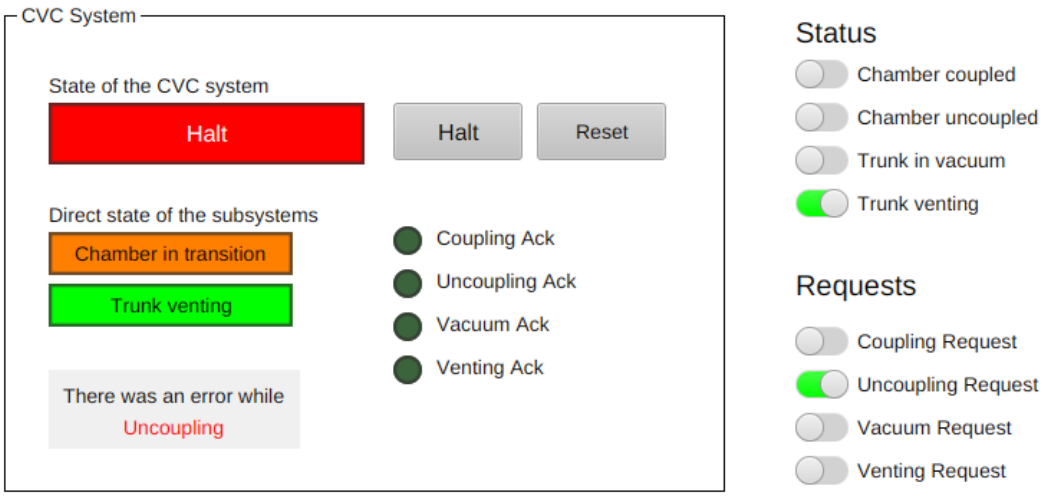


Figure 5.4: *The CVCS GUI after an operation halt. The message specifies the operation that caused the error.*

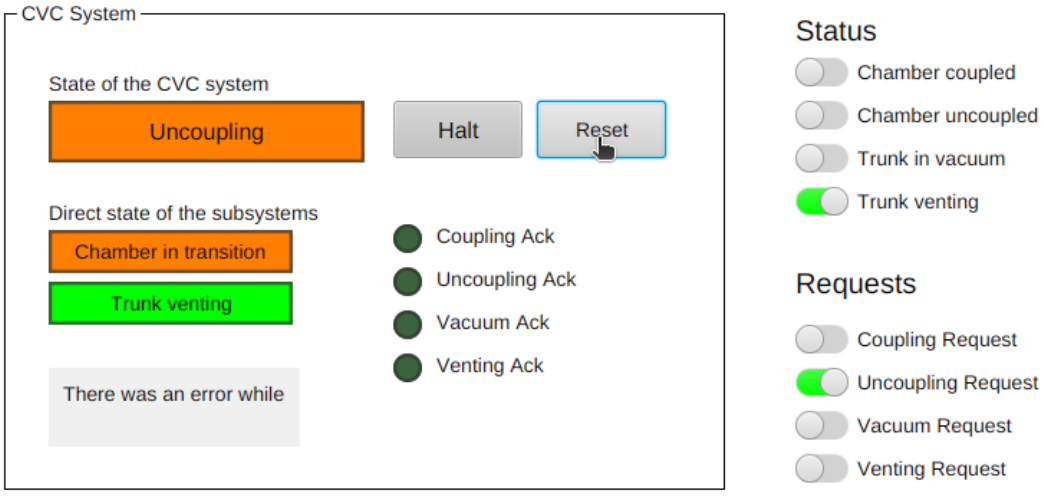


Figure 5.5: *The CVCS GUI after a reset. It returns to the operation's state without the acknowledgment to continue.*

Chapter 6

Testing

The testing phase plays a key role in the development of robust systems, particularly so for the Machine Protection System (MPS). To guarantee the integrity of the CVCS, I developed a dedicated testing environment. This allowed me to test edge cases, identifying and resolving potential failure points and minimizing the risk of unexpected behavior.

Automated unit testing was employed to verify all the procedures of the PLC program trough python script using modbus requests to read and write to the PLC variables.

Integration testing involved tests to ensure that all the PLC's variables were correctly exposed to the IOC and shown in the GUI. This was done periodically and manually, given the low number of possible inputs and states.

6.1 Pytest

Pytest is a powerful Python testing library that simplifies test creation and execution. By using python's syntax, it promotes readable tests that you can write quickly. Pytest's process of writing tests is streamlined and adaptable, both for small functions and scaling to complex applications.

Tests are composed of python functions prefixed with `test_`. These functions use assertions to verify specific conditions, determining test's success or failure.

A core part of the library is the use of "fixtures". These are functions that can be executed before or after individual tests or test groups. This allows for easy setup and cleanup of the test environment.

Another powerful technique is Pytest's parametrization. This allows to run the same test with different sets of input data.Pytest executes the test for each set of parameters, eliminating the need for writing repetitive test functions.

6.2 Unit testing on PLC simulator

The unit testing setup consisted of a Python script using Pytest, a library for testing functionality, and Pymodbus, to establish a Modbus connection with the PLC. The PLC program was executed by the Control Expert simulator within a Windows virtual machine (see 5.1).

I implemented several utility functions to improve code readability and maintainability. These functions allow the use of descriptive enumerations for PLC addresses and state codes. Furthermore, a utility function for forcing PLC states and dedicated functions to automate `halt()` and `reset()` operations streamline the testing process.

The comprehensive test suite includes 80 parameterized test cases, designed to validate a wide range of CVCS behaviors. These tests cover:

- `test_startup_with_state`: to verify that the reset state recognizes each valid and invalid state and brings the FSM to the correct status.
- `test_ack_response_from_state`: to verify that each operation request is correctly acknowledged or ignored correctly by each state.
- `test_transition`: a set of tests that verify that transitions ends as expected. They cover
    1. transitions where the status changed correctly
    2. transitions where the status of the other subsystem changed
    3. transitions where no status changed.
    4. transitions where a status rises and immediately fall.
    5. transitions where statuses of both systems changed.
    6. transitions started without permission.
- `test_reset_after_halt_during_transition`: to verify that the system halts during transition, the reset operation brings the system back to that transition (Listing 9).
- `test_multiple_requests_at_once`: to test that if the system receives multiple requests, only the first one is acknowledged.
- `test_reset_after_request_during_halt_results_in_no_ack`: to verify that if the system receive a request during a halt, after a reset no request is acknowledged. Another version exist for multiple requests.

The Listing 9 shows an example of the `test_reset_after_halt_during_transition` function. The test is enabled, thank to a pytest parametrization, to run multiple times. Each time it uses a different set of parameters, describing a different procedure. These parameters simulate

```python
    @pytest.mark.parametrize
        ("initial_state,  req_addr,            falling_addr,        expected_state", [
        (State.READY,     Addr.VENTING_REQ,    Addr.TRNK_IN_VACUUM, State.VENTING),
        (State.COUPLED,   Addr.VACUUMING_REQ,  Addr.TRNK_VENTING,   State.VACUUMING),
        (State.COUPLED,   Addr.UNCOUPLING_REQ, Addr.TIS_COUPLED,    State.UNCOUPLING),
        (State.UNCOUPLED, Addr.COUPLING_REQ,   Addr.TIS_UNCOUPLED,  State.COUPLING),
        ])
    def test_reset_after_halt_during_transition
        (initial_state, req_addr, falling_addr, expected_state, client):

        forceState(client, initial_state)
        write(client, req_addr, 1)
        write(client, falling_addr, 0)
        halt(client)
        assert read(client, Addr.INCOMPLETE_TRANSITION) == expected_state.value
        reset(client)
        assert read(client, Addr.STATE_CODE) == expected_state.value
```

Listing 9: *Example of a parameterized test for verification of the state code of the after a reset.*

various PLC states and trigger conditions. The test's core logic verifies that if the system halts during a transition, a subsequent reset correctly restores the system to the expected transition state.

The first three lines of the function define the setup phase: starting from a particular `initial_state`, an operation request is sent, accepted triggering the appropriate PLC signal change, and then a state changes, simulating the start of the transition. Here, the system is requested to `halt()` and it is verified, in the first assertion, that the systems correctly saved the code of the `INCOMPLETE_TRANSITION`. Then we ask the system to `reset()` and we check that the new state is the operation state the system was in.

The testing process demonstrated the system's adherence to the designed protocol. Through these tests, I discovered and resolved areas where the system's diverged from expectations. This increased the level of confidence in the CVCS's performance within the future production environment, contributing to the overall reliability of the MPS.

Chapter 7

Conclusions

This thesis addressed the challenge of coordinating the Coupling Table (CPTA) and the Vacuum Control System (VCS) within the SPES facility. This coordination is important for the safe and efficient operation of these systems, contributing to the broader Machine Protection System (MPS) goals.

To address this challenge, I developed the Coupling-Vacuum Coordination System (CVCS). The system employs a Finite State Machine (FSM) to manage the transactions between the CPTA and VCS. PLC-based logic ensures a reliable implementation, while an EPICS server allows for remote control. A graphical user interface (GUI) allows operators to monitor and control the system.

Testing in a simulated environment has demonstrated the CVCS's ability to enforce the coordination. This ensures that the vacuum level within the Target Ion-Source (TIS) chamber is correctly managed, contributing to the equipment protection.

7.1 Future works

The next step is to install the CVCS's PLC logic on the main MPS PLC and test it within a controlled environment, using the actual physical systems and their associated statuses. This will validate its functionality under real-world conditions.

Some additional features can be implemented. Important ones include to allow the operator to manually give consent to an authorisation, in series with automatic consents, and to set bypasses or operating modes. to override the systems coordination.

Finally, the CVCS is only one of the many critical components within the larger MPS. Future development will involve implementing additional protection and coordination functions to ensure its operative at every step of SPES.

Bibliography

[1] T. Marchi, G. Prete, F. Gramegna, *et al.*, "The SPES facility at Legnaro National Laboratories," *Journal of Physics: Conference Series*, vol. 1643, p. 012 036, Dec. 2020. doi: 10.1088/1742-6596/1643/1/012036.

[2] A. Andrighetto, M. Tosato, M. Ballan, *et al.*, "The ISOLPHARM project: ISOL-based production of radionuclides for medical applications," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 322, pp. 73–77, 2019.

[3] National Laboratories of Legnaro, *Spes gamma laramed*. [Online]. Available: https://www.lnl.infn.it/spes-gamma-laramed/ (visited on 03/03/2023).

[4] Bdushaw, Wikimedia Commons, Licensed under Creative Commons Attribution-Share Alike 4.0 International (CC BY-SA 4.0), 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:HalflifeNuDat2.png.

[5] M. Thoennessen, "Current status and future potential of nuclide discoveries," *Reports on Progress in Physics*, vol. 76, no. 5, p. 056 301, Apr. 2013. doi: 10.1088/0034-4885/76/5/056301. [Online]. Available: https://dx.doi.org/10.1088/0034-4885/76/5/056301.

[6] EURISOL Collaboration, *The Eurisol Project*. [Online]. Available: http://www.eurisol.org/ (visited on 02/28/2024).

[7] M. Lindroos, "Review of ISOL-type radioactive beam facilities," in *Proceedings of EPAC*, vol. 2004, 2004, p. 6.

[8] J. C. Cornell *et al.*, "Radioactive beam facilities in europe: Current status and future development," *arXiv preprint nucl-ex/0501030*, 2005.

[9] G. Prete, A. Andrighetto, M. Manzolaro, *et al.*, "The SPES project at the INFN-Laboratori Nazionali di Legnaro," in *EPJ Web of Conferences*, EDP Sciences, vol. 66, 2014, p. 11 030.

[10] A. Galatà, L. Bellan, G. Bisoffi, *et al.*, "ADIGE: the radioactive ion beam injector of the SPES project," *Journal of Physics: Conference Series*, vol. 874, p. 012 052, Jul. 2017. doi: 10.1088/1742-6596/874/1/012052.

[11] *AGATA - Advanced GAmma Tracking Array*, AGATA Collaboration, 2024. [Online]. Available: https://www.agata.org/about (visited on 02/29/2024).

[12] G. Lilli, L. Centofante, M. Manzolaro, A. Monetti, R. Oboe, and A. Andrighetto, "Remote handling systems for the Selective Production of Exotic Species (SPES) facility," *Nuclear Engineering and Technology*, vol. 55, no. 1, pp. 378–390, 2023, issn: 1738-5733. doi: https://doi.org/10.1016/j.net.2022.08.034. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1738573322004326.

[13] D. Marcato, F. Gelain, G. Savarese, *et al.*, "Machine protection system design," *LNL Annual Report 2022*, p. 142, 2023.

[14] *EPICS Controls*, Argonne National Laboratory, 2024. [Online]. Available: https://epics-controls.org/ (visited on 02/28/2024).

[15] *EPICS Record Reference*, 2024. [Online]. Available: https://epics.anl.gov/base/R7-0/6-docs/RecordReference.html.

[16] Argonne National Laboratory, *Channel access protocol specification*, 2023. [Online]. Available: https://epics.anl.gov/docs/CAproto.html (visited on 03/02/2023).

[17] *Modicon M580, Hardware Reference Guide*, version 14, Schneider Electric, Dec. 1, 2023.

[18] M. Roetta, D. Bortolato, F. D'Agostini, and A. Andrighetto, "Coupling table system design," *LNL Annual Report 2018*, p. 27, 2019.

[19] G. Savarese, L. Antoniazzi, D. Bortolato, F. Gelain, D. Marcato, and C. Roncolato, "Vacuum control system upgrade for ALPI accelerator," Jul. 2022. doi: 10.18429/JACoW-IPAC2022-MOPOMS045.

Acronyms

AGATA Advanced Gamma Tracking Array. 7

ALPI Acceleratore Lineare Per Ioni. 5

CA Channel Access. 11, 13

CPTA Coupling Table. 1, 8, 17, 20–22, 39

CSS Control System Studio. 11, 25

CVCS Coupling-Vacuum Coordination System. 1, 21, 22, 25, 26, 32, 35, 39

EPICS Experimental Physics and Industrial Control System. 11, 25, 29

FE Front-End. 7, 17

FSM Finite State Machine. 1, 26, 36, 39

HHM Horizontal Handling Machine. 8, 18

INFN Italian Institute of Nuclear Physics. 3

IOC Input-Output Controller. 12, 25, 29

LNL National Laboratories of Legnaro. 1, 3

MPS Machine Protection System. 1, 8, 9, 17, 35, 37, 39

PLC Programmable Logic Controller. 9, 25

PPB Primary Protonic Beam. 5, 17

PV Process Variable. 12, 25, 30

RIB Radioactive Ion Beam. 1, 3, 5, 17

SPES Selective Production of Exotic Species. 1, 3, 4, 8, 39

TIS Target Ion-Source. 5, 7, 17, 20, 28, 39

VCS Vacuum Control System. 1, 17, 18, 20–22, 39