Automation of a High Voltage Laboratory

Report/Manual about the control system of the ML9901 High Voltage Laboratory of the ABB Corporate Research Center in Baden-Dättwil, Switzerland

Thesis submitted to:

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

Universitat Politècnica de Catalunya

Performed at the premises of:

Gas Circuit Breakers Group

ABB Corporate Research Center, Baden-Dättwil (Switzerland)

Supervised by:

Dr. Philipp Simka, ABB

Dr. Ricard Bosch Tous, UPC

Presented by:

Marc Julià Carrillo

Barcelona, May 2014

Abstract

This document is a report and manual on the work done by the author in the control system of a High Voltage Laboratory in the Corporate Research Center of ABB in Baden-Dättwil, Switzerland.

The project consisted in implementing the possibility to run automatic test sequences in the Lab, which was before just controlled manually through a Console. To do so, the author had to first think and implement a communication system between all the components in the Lab (the old and the new ones); to then program the Controller running the Lab and a Touch Panel so the future users could configure the test sequences easily from a convenient interface.

Regarding the document itself, first of all there is an introduction to the kind of experiments carried out in the Lab, like Lightning or Switching Impulse tests. Following, there is an overview and general explanation of the Lab: its whole functionality, how it is structured and the communication between the different parts, and possible circuits to build and ways to operate it. There is also a description of all the components in the Laboratory, explaining their main characteristics, how they work and what they are used for.

The main part of the thesis, however, is the detailed documentation of the codes programmed for the Controller and the Touch Panel, and the manual on how to operate the control system of the Lab. These parts want to be useful for people carrying experiments in the Lab, so they should be understandable for not initiated people; but also for future updates of it, as an structured base for more automated sequences to be added and more functionalities to be implemented as soon as they are needed.

Finally, there is a brief chapter dedicated to some little works done for improving parts of the Lab, like the implementation of a cooling system in the Control Rack of the Lab, or the study of the effects of the atmospheric conditions in some experiments carried out in the Laboratory.

Table of Contents

List of figures

List of Tables

Notation

The following technical notation is used throughout the document:

Fig.: Figure

HV: High Voltage

AC: Alternate Current.

DC: Direct Current.

LI: Lightning Impulse.

SI: Switching Impulse.

PC: Personal Computer.

MCU: Main Control Unit.

PLC: Programmable Logic Controller

SCU: Safety Control Unit.

NC: Normally Closed.

NO: Normally Open.

UI: User Interface.

HMI; Human Machine Interface.

SG: Spark Gap.

GS: Ground Switch.

PCB: Printed Circuit Board.

1. Preface

1.1. Project origin

After almost one year in Switzerland in the frame of the UNITECH program, doing an exchange in ETH Zurich first and then an internship in a small company in the field of optics, the author was looking for the possibility to carry out his Final Project (PFC) in Industrial Engineering in a company of this country, as it has many technological firms and a lot of work for engineers.

After applying to *ABB*, he got an offer to do an internship in the Corporate Research Center of the company in Switzerland, working in the automation of a High Voltage Lab for the Gas Circuit Breakers Group, under the supervision of Dr. Philipp Simka. Moreover, he was offered the possibility to write his PFC out of the project done in the internship.

1.2. Motivation

A first motivation was to get familiar with PLC programming and the basics of the system controls implemented in the Lab. Moreover, it was a great opportunity to learn about High Voltage, a field not very much covered during the years in university. Finally, there was a big motivation also for learning to carry on a big project over several months, with different phases and things to do and try.

Other not so academic motivations were the possibility to get to know the research environment of a leading company like *ABB*, and to work and live more time abroad for professional and personal growing.

1.3. Objectives of the project

The main goal of the project is to upgrade a High Voltage Lab so that it can run some test sequences by its own. To do so, the communication system between the components in the Lab has to be checked and the appropriate changes have to be made in order to implement a Touch Panel that will allow the user to program the automatic sequences, and to prepare

an Oscilloscope to record data from them automatically. Then, the Touch Panel and the control system of the Lab have to be programmed to work under the new premises and run the automated test sequences. And finally everything has to be built and set in their housings and locations in the Lab.

The other objective of the project is to get an overview of the functionality of the whole Lab and write a report explaining all of its components, the work done, and detailing how to operate it and how everything works for future users. This report is actually the document following these pages.

1.4. Scope of the project

The scope of the project is to leave the control system of the Lab ready to run some predefined automated sequences, with the softwares running the control unit and the Touch Panel ready to be used with the different modes implemented so far.

Moreover, all this has to be documented in a report/manual, where the whole functionality of the Lab and its main components is generally described. However, there is no need to get too detailed in some components that were already there before the arrival of the student and that won't be modified during his stay there. Examples of this are the Safety Circuit or most of the elements in the High Voltage Setup, that will be just described with a general explanation on how they work based on the reports and the work done by a previous intern.

On the other hand, the Control System and how to operate it will be explained in detail. This way, future users of the Lab will be able to refer to this document in order to learn how to operate it, fix possible bugs, modify options or implement new features.

In conclusion, this works has to serve as a basement for future projects and updates in the Lab, being mainly focused in its control system

2. Introduction

ABB is a world leader in Automation and Power technologies. The corporation (which has more than a hundred thousand employees worldwide) was born from the merge of the Swedish company Asea and the Swiss Brown Bovery, and settled its headquarters in Zurich (Switzerland).

The company has seven Corporate Research Centers around the world, where the cutting edge technologies and products are developed. In the one in Baden-Dättwil (Switzerland), there is a group conducting research about “Gas Circuit Breakers”. These devices are used to disconnect high voltage circuits, like the ones in power plants or big electric stations.

The work principle of a gas circuit breaker starts with separating two metal contacts, like in a normal interrupter. However, the high voltage between the contacts creates an electrical arc right after they separate, which has to be blown to interrupt the current flow. To do so, gas circuit breakers use a gas blast of sufficient intensity to cool down the arc and prevent it to be created again. The most common gas used nowadays is SF_6, due to its excellent dielectric properties.

Apart from research in the gas circuit breakers themselves, all kind of experiments that imply high voltage are carried out in the group also, like the ones to find the insulating capacity of a certain material or product.

In order to comply with the international standards and be safe against overvoltages or other distortions in the network, a material has to be able to withstand special circumstances in the voltage shapes. These special circumstances are represented by two types of voltage shapes:

- **Short-duration power-frequency voltages**: Voltages higher than the maximum rated for the device, but with a frequency similar to the one of the network, and applied during a short time (around 60 seconds).

- **Lightning Impulse (LI) and Switching Impulse (SI) voltages**: These voltages shapes are characterized by a voltage pulse with a double exponential form: a fast rise time (T_1) and a slower decay time to half-value (T_2), as it can be seen in **Fig. 1**. Depending on the values of T_1 and T_2, the impulse can represent different real circumstances experienced by insulating devices. The two most typical ones that have standardized values to be withstood according to the highest voltage for the equipment are the **Lightning Impulse** and the **Switching Impulse**.

The **Lightning Impulse** is a voltage pulse with a rise time of 1.2 μs and a time to half-value of 50 μs. This is the shape experimentally found to represent the effect of a real lightning coming from a storm. The **Switching Impulse** has a T_1 of 250 μs and a T_2 of 2500 μs, and represents the overvoltages appeared when interrupting a circuit.

In order to test the insulation level against **short-duration power-frequency voltages**, a normal Alternate Current (AC) or Direct Current (DC) circuit is used, applying a higher voltage than its theoretical maximum to the test object during a short time to check if there is breakdown.

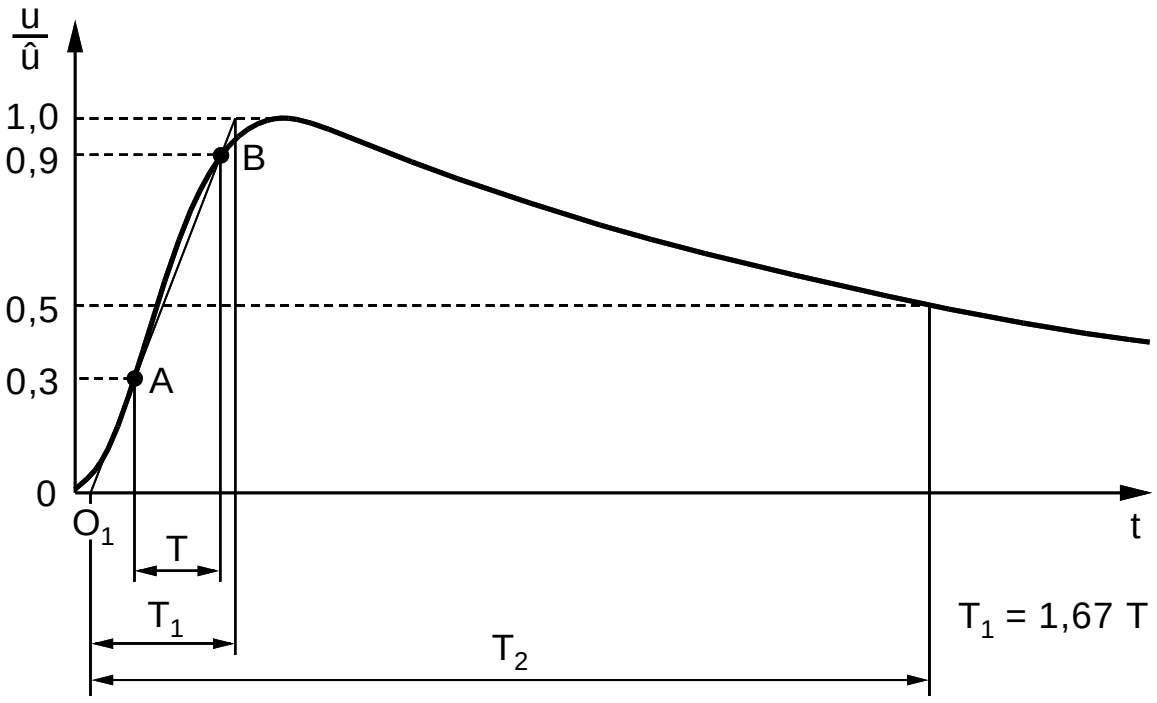


Fig. 1 Double exponential voltage form of the Lightning Impulse

To reproduce the double exponential voltage shape and so check the insulation level against **voltage impulses**, a group of capacitors is charged in parallel by a DC power source and then discharged in series on a circuit including the test object and a resistor in parallel.

3. Overall Concept

The High Voltage Laboratory ML9901 of ABB Corporate Research Center is able to test objects under Alternate Current (AC), Direct Current (DC) or different kinds of Impulses; and up to 420 kV depending on the mode. Half of the room consists in a faraday cage of about 150 cubic meters (around 5 x 10 x 3 meters), while the other half is used for storage, control system, etc. The High Voltage setup is installed inside the faraday Cage, in a space with 5.20 meters between the closest walls, and 3.3 meters between floor and ceiling.

Fig. 2 Interior of the faraday cage with a circuit and a test object, before the renovation of the Lab

This setup is controlled by a Programmable Logic Controller (PLC) that activates an AC Source, and supervised all the time by a safety circuit, which should prevent any dangerous situation. The user interface consists in a console with physical buttons, a touch panel to run the automated test sequences, and an oscilloscope to record the results.

3.1. Communication and signal flow

In order to have all the desired features and reinforce the security of the setup on top, it's very important to pay attention and correctly design the communication scheme between the different components of the lab. So that, the following graph [**Fig. 3**] shows the main connections between the whole system.

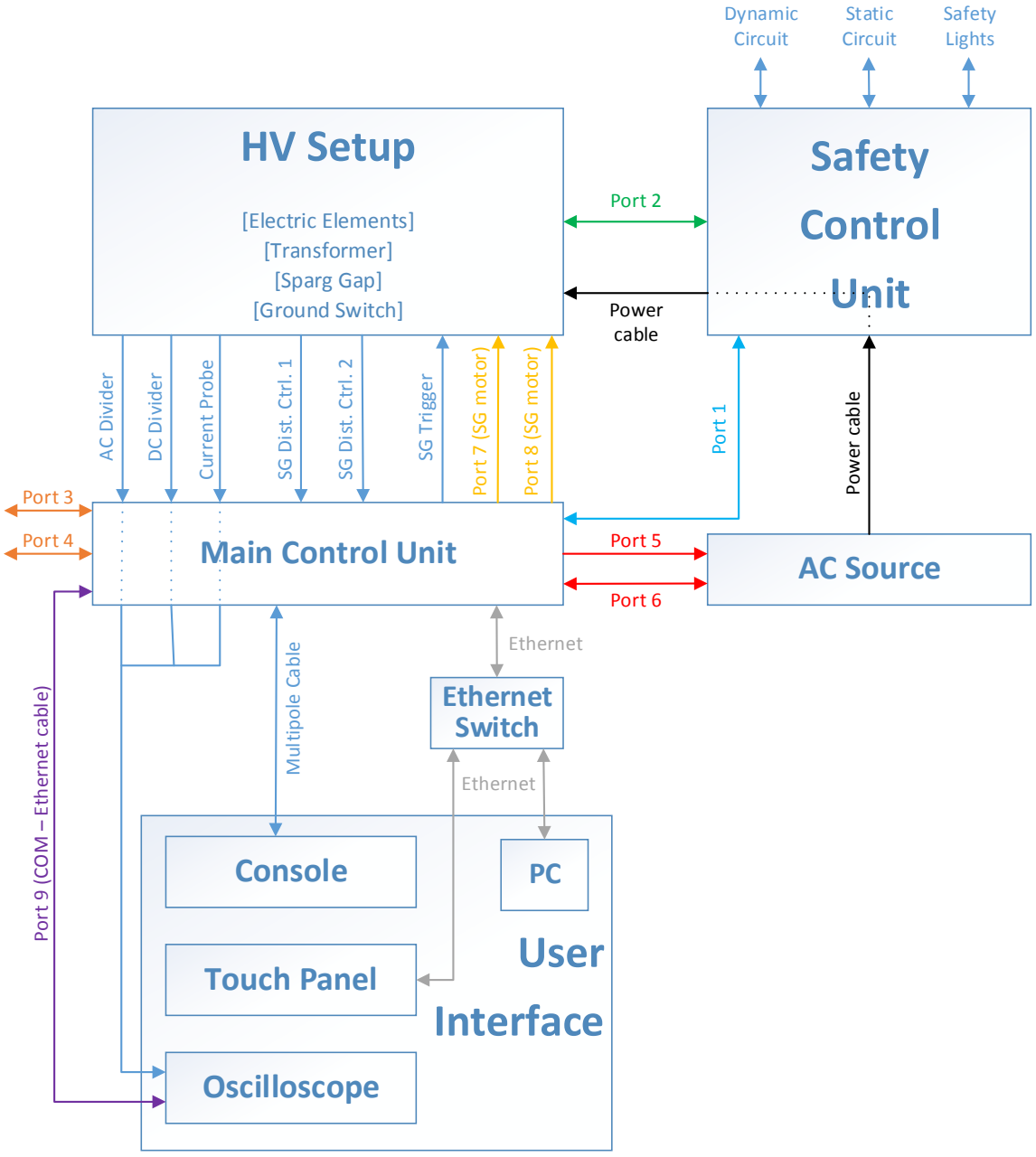


Fig. 3 Graph showing the connections between all the components in the Lab

At the top, the Safety Control Unit receives data from the Static and Dynamic Safety Circuits, controls the Safety Lights and interacts with the PLC through a cable connection. Moreover, it is also connected to the HV Setup.

The Main Control Unit (PLC) gets signals from the HV Setup through the BNC cables of the AC and DC voltage dividers (which are also sent to the Oscilloscope) and the current probe (in order to detect current going to the ground from the circuit); and through the optical fiber cables coming from the encoder in the Spark Gap motor (to control its distance). As outputs to the HV Setup it has an optical fiber cable to trigger the Spark Gap, and two power cables to control its motor. In order to control the AC Source, two more COM cables connect this one with the PLC. Finally, the regulated power from the AC Source goes through the Safety Control Unit (which will cut the power if the Safety Status is not right) and to the high voltage transformer inside the HV Setup.

All this automated system is managed from the different elements which compound the User Interface. The Console is connected to the Main Control Unit through a customized MultiPole cable. The Ethernet port from the PLC is plugged into a switch that makes possible to connect the PLC itself, the Touch Panel and a laptop (used to program the controller and the Touch Panel) at the same time through this network protocol. Finally, the oscilloscope is remotely controlled from the PLC through Serial connection.

3.2. Possible modes of operation

The ML9901 Lab can be operated either manually (using the Console) or automatically (using the Touch Panel).

When controlled through the Console, the user can select the desired voltage form (AC, DC or Impulse), set the voltage and frequency, and finally send the voltage to the circuit. This has to be done in real time, with no possibility to program it or, for example, rump up the voltage at a specific ratio.

To solve this and be able to run some automated test sequences, the Touch Panel has been implemented. As it is now, the Lab can run two types of sequences on its own:

- **Continuous Sequence**: It consists in an AC or DC voltage shape designed by the user through the Touch Panel, adding blocks of a certain duration where the voltage goes from a value to another. It has some more features too, like changing the frequency in each block, the possibility to repeat the same shape a certain amount of times, choosing the behavior after certain events like detecting a breakdown or finishing a repetition, or choosing the duration of the pause and if to ground the circuit or not between two repetitions.

- **Impulse sequence**: It consists in a set of voltage pulses that will be applied to the test object until one of the restrictions set by the user are reached. These restrictions can be the number of shots, number of breakdowns or a minimum or maximum voltage to reach. The voltage applied at every shot can be randomized or following a pattern (increasing it or decreasing it by a certain value depending on if a breakdown has been detected or not). It features also the possibility to choose the behavior after a shot or a breakdown, or to configure the duration of a pause between two pulses.

In all the automated modes there is also the possibility to automatically configure the Oscilloscope to adapt to the current voltage being applied, capture the wave and store the data recorded.

When the sequences are finished, the Touch Panel also shows some information about the experiment carried out, like the total duration, if it has been aborted before finishing, or the number of breakdowns (and when they happened) and repetitions or shots.

How to run all these modes is accurately explained later in **Point 5.2** of the document.

These are the automatic test sequences implemented so far. But, as the Touch Panel and the PLC are fully programmable from a PC, more automated modes can be added in the future according to the needs in the Lab. A detailed explanation of the code and how to program it is written in **Point 6** of the thesis.

3.3. Possible Circuits

Using the different electric components in the Lab (explained later in **Point 4.1**), many circuits can be built to apply different voltage forms to the test objects. Following, there is a brief description of the ones already prepared to be mounted in ML9901 Lab.

3.3.1. AC – 1 Stage

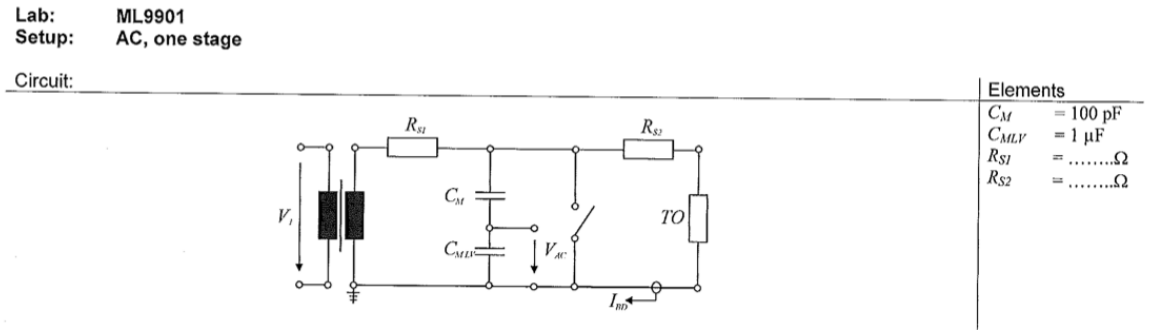


Fig. 4 AC - 1 Stage Circuit

This is a basic circuit to apply a maximum of 100 kV AC from the transformer directly to the test object. It features an AC voltage divider built with two capacitors that gives a ratio of 1 to 10.000; and the possibility to ground the setup with the Ground Switch.

3.3.2. DC – 1 Stage

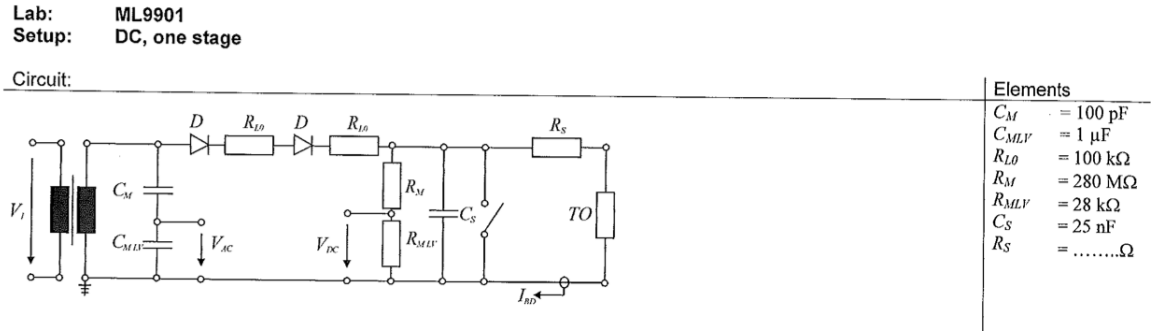


Fig. 5 DC - 1 Stage circuit

This circuit takes the AC voltage from the transformer and rectifies it with a pair of diodes and a capacitor in order to apply a maximum of 140 kV DC voltage to the test object. Apart from the AC divider next to the transformer, it also has a DC divider built with two resistors that gives a ratio of 1 to 10.000.

It is important to remember that this circuit will take some time to discharge after shutting down the transformer and stopping to apply voltage.

3.3.3. Arbitrary Impulse – 1 Stage

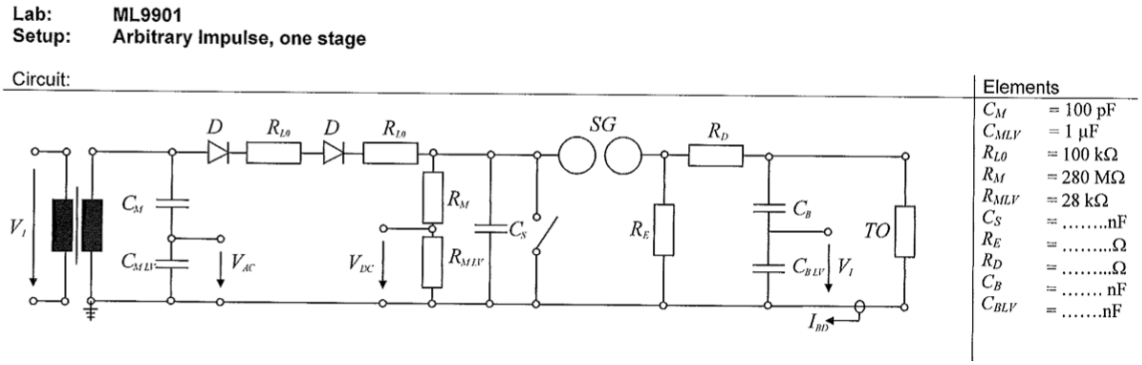


Fig. 6 Arbitrary Impulse - 1 Stage circuit

This circuit has a part with a similar logic to the DC one, with the diodes and capacitor to rectify the AC voltage from the transformer. But instead of applying this voltage to a test object, it charges the C_s capacitor, as it can be seen in **Fig. 6**.

A Spark Gap separates the charging part of the circuit from the one where the test object is. This way, as soon as the device is triggered, the circuit discharges the C_s capacitor through R_E and R_D, in order to apply a double exponential voltage impulse form of maximum 140 kV to the test object.

Changing the values of C_B, R_E and R_D it is possible to set the rise time and the half voltage decay time of the double exponential pulse form, in order to imitate real voltage shapes the test objects should be able to withstand.

Finally, C_B is also part of a voltage divider capable to measure the voltage applied to the test object in order to view the shape of the voltage form in the oscilloscope.

3.3.4. Lightning Impulse

The Lightning Impulse circuit is a specific form of the Arbitrary Impulse one, with specific values for the damping and discharge resistors in order to try to copy the voltage shape appeared when there is a real Lightning. In its one stage configuration, it can apply an impulse of maximum 140 kV.

It works exactly the same way as the Arbitrary Impulse circuit, with a rectifier after the transformer to charge the C_s capacitor, and the Spark Gap separating this part of the circuit from the test object and the discharge and damping resistors.

In ML9901 Lab this circuit can be built in its one, two or three stages versions, allowing to apply theoretical maximum Impulses of 140, 280 and 420 kV respectively. The two and three stages versions are built by adding one or two layers more of components above the basic circuit. This way, the transformer can charge a C_S capacitor in each stage, and these will later add up their voltages so the circuit can shoot an impulse of up to three times the voltage the transformer gives out.

Regardless of the number of stages in the circuit, the trigger is always done in the lower Spark Gap, moving this the potential up in the circuit and so automatically triggering the other Spark Gaps in order to bring the total addition of voltages to the top of the circuit and to the test object.

Following are the schematics for the three possible Lightning Impulse circuits:

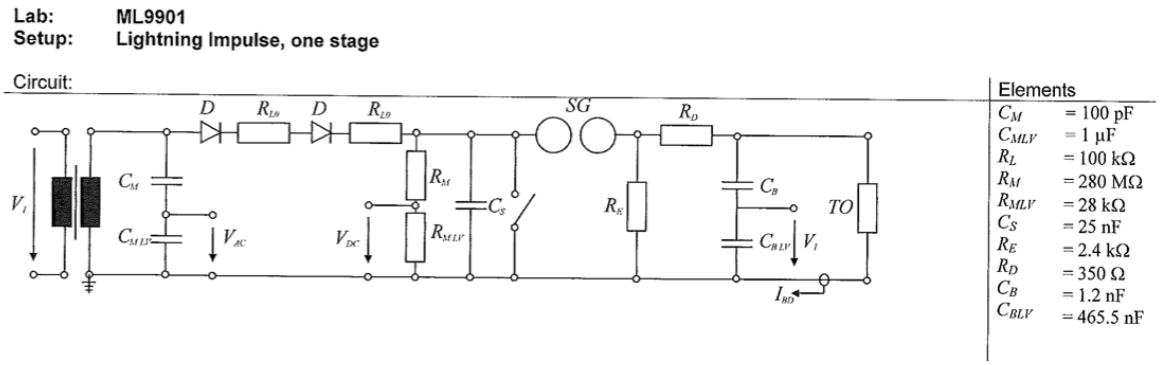
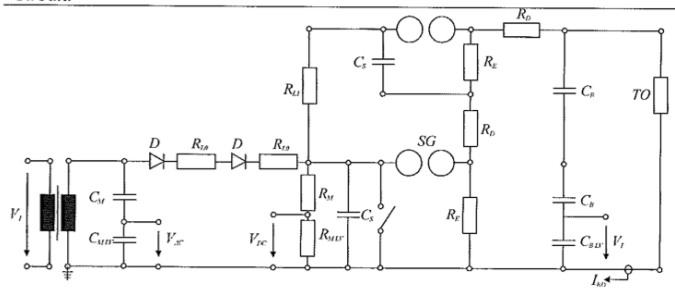


Fig. 7 Lightning Impulse - 1 Stage circuit

Lab: **ML9901**
Setup: **Lightning Impulse, two stage**

Circuit:

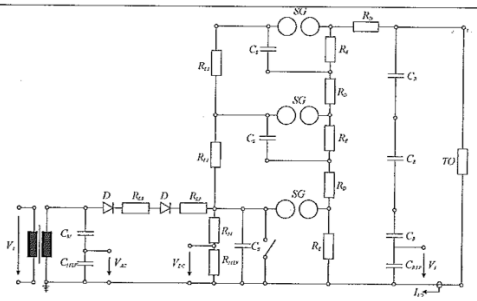


Elements	
C_M	= 100 pF
C_{MLV}	= 1 µF
R_L	= 100 kΩ
R_M	= 280 MΩ
R_{MLV}	= 28 kΩ
C_S	= 25 nF
R_E	= 2.4 kΩ
R_D	= 350 Ω
C_B	= 1.2 nF
C_{BLV}	= 501 nF

Fig. 8 Lightning Impulse - 2 Stages circuit

Lab: **ML9901**
Setup: **Lightning Impulse, three stages**

Circuit:



Elements	
C_M	= 100 pF
C_{MLV}	= 1 µF
R_{L0}	= 100 kΩ
R_{LI}	= 10 MΩ
R_M	= 280 MΩ
R_{MLV}	= 28 kΩ
C_S	= 25 nF
R_E	= 2.4 kΩ
R_D	= 350 Ω
C_B	= 1.2 nF
C_{BLV}	= 501 nF

Fig. 9 Lightning Impulse - 3 Stages circuit

3.3.5. Switching Impulse

This circuit has the same composition as the previous Impulse circuits, but with the specific combination of discharge and damping resistors in order to imitate the voltage shape appeared when switching a high voltage device.

As the Lightning Impulse circuit, it can be built also in one, two or three stages versions, in order to shoot up to 140, 280 and 420 kV respectively. Here are the schematics for each circuit:

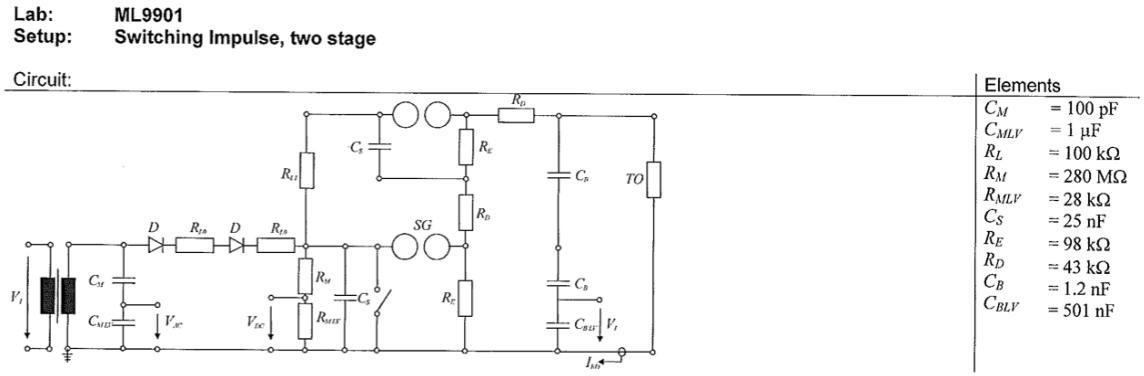


Fig. 10 Switching Impulse - 2 Stages circuit

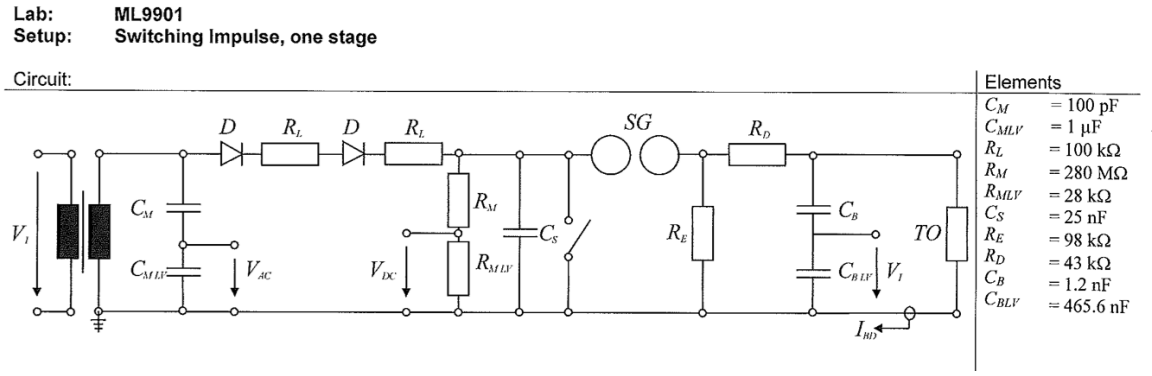
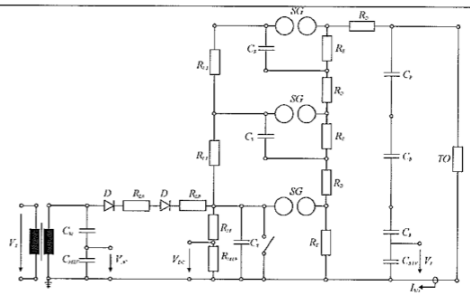


Fig. 11 Switching Impulse - 1 Stage circuit

Lab: **ML9901**
Setup: **Switching Impulse, three stages**

Circuit:



Elements	
C_M	= 100 pF
C_{MLV}	= 1 µF
R_{L0}	= 100 kΩ
R_{L1}	= 10 MΩ
R_M	= 280 MΩ
R_{MLV}	= 28 kΩ
C_S	= 25 nF
R_E	= 98 kΩ
R_D	= 43 kΩ
C_B	= 1.2 nF
C_{BLV}	= 501 nF

Fig. 12 Switching Impulse - 3 Stages circuit

4. Main Components of the Lab

4.1. High Voltage Setup

The HV Setup is a Construction Kit by W.S. Test Systems. This kit is formed by electric elements (resistors, capacitors, connections rods, insulators, etc.) that can be combined to build different circuits. All the insulating elements have similar shape and a length of 50 cm, and are rated to stand up to 140 kV across them. So that, they can be assembled and disassembled to create the circuits. Due to stability and insulating issues, the maximum stackable height in the circuits is 3 stages of components.

The high voltage source of the Setup is a transformer, model *TEO 100/10* also by *W.S. Test Systems*. This transformer can output up to 100 kV_{rms} from a 220 V_{rms} input, as it can be seen in the following key data:

- Rated power: 5kVA
- Rated Frequency: 50Hz
- Primary voltage: 220V rms
- Primary current: 22,8 A
- Secondary voltage: 100kV rms
- Secondary current: 0,05 A

4.1.1. Modules in the HV Setup

The elements in the HV Setup of the ML9901 Lab are labelled with the following format: one or two letters according to the German acronym for the type of element (discharge resistor, load resistor, capacitor…), a number to differentiate the identical elements, and two letters more when they are specific for a kind of circuit (*LI* for Lightning, *SI* for Switching or *PT* for PigTail).

Following there is a description of the different elements of the HV Construction kit that can be found In the Lab (they can be seen in **Fig. 13**):

- **Resistors**: Ranging from 133 Ω to 280 MΩ. They are labelled as *RL* (load resistors), *RE* (discharge resistors) and *RD* (dumping resistors). There is a fourth type of resistors labelled as *RM*, which have the possibility to attach another small resistor in series to use them as DC divider and measure the voltage in the setup.

- **Capacitors**: Ranging from 100 pF to more than 500 F. Labelled as *CB* (Charge Capacitors), *CS* (Storage Capacitors) and CM. *CB* and *CM* elements have the possibility to attach another capacitor and be used as dividers to measure AC voltage in the setup.

- **Diodes**: Consisting in a diode and a 100 kΩ resistor, and labelled as *D* and a number.

- **Spark Gap**: Consisting of two metal spheres with variable distance (set using a motor and an encoder), in order to ignite an arc at a specific voltage. One of them has a spark generator run by two 9V batteries in one of the spheres in order to ignite the spark that triggers the electric impulse. They are labelled as *FS* and a number.

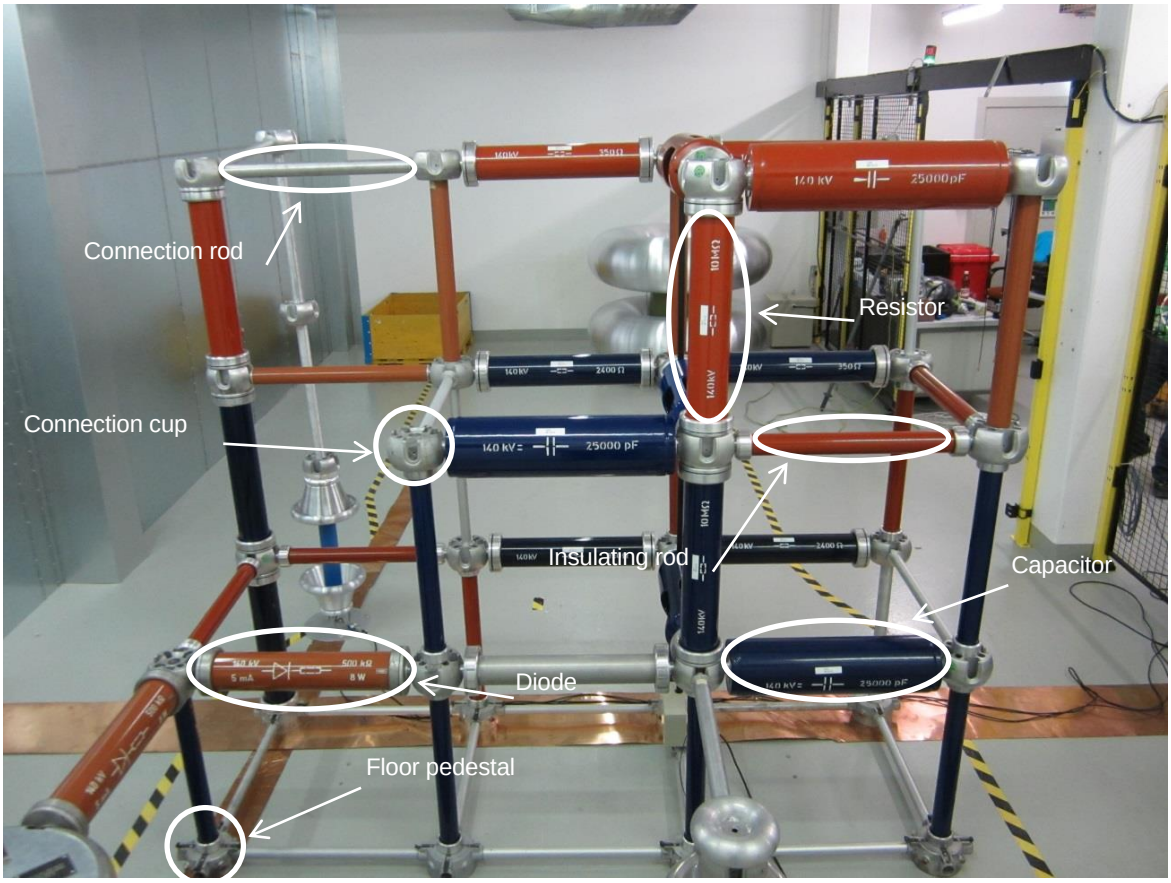


Fig. 13 Different electric elements to build the circuits

- **Connecting elements**: Like metal rods, connection cups and floor pedestals. They are used to connect the main elements of the circuit. They are not labelled.

- **Insulating rods**: To insulate some parts of the setup. They are thin blue or red rods which are not labelled.

- **Corona protector elements**: Consisting of toroids and rounded metal parts in order to cover the sharp edges in the circuit and avoid corona field and partial discharges. They are not labelled and there are toroids of different sizes in the lab.

A full inventory of all the electric elements of the HV Setup present in the Lab, together with their nominal and measured values and some more information, can be found in **Annex A: Electric Elements in ML9901 Lab**.

4.2. Safety Circuit (SC)

The safety circuit in ML9901 Lab is a hardware implementation in order to avoid any kind of voltage to be applied to the setup if all the safety measures are not fulfilled.

The Safety Circuit checks the following security conditions before allowing to turn the status to ON: the door of the faraday cage has to be closed; the ground stick has to be on its holder blocking access to the door; a key switch in the control rack has to be turned ON; and the emergency buttons to turn OFF the Safety Status must be reset. If these conditions are fulfilled, it will be possible to turn ON the Safety Status by pressing the specific button in the Console. This will modify the Safety Status in the PLC, activate the Output relays of the AC Source, open the Ground Switch in the HV Setup, and change the Safety Lights.

Fig. 14 Front of the Safety Control Unit, with the Key Switch, buttons to turn ON and OFF and to control the lights, connections to the different circuits, and two auxiliary power sockets

The SC is formed by a static and a dynamic circuit. The Static Circuit controls the Ground Stick holder and the Key Switch; while the Dynamic Circuit checks the Emergency buttons and the contacts in the door.

4.2.1. Safety Controller

Fig. 15 Safety Controller *Jokab Safety Vital 3*

The safety controller is stored in the Safety Control Unit (SCU) in the control rack. It is the model *Jokab Safety Vital 3*, a safety system by *Jokab Safety AG* (member of the ABB group). It supports two different kinds of safety sensors with different techniques; and complies with CRC safety concept and EN ISO 13849. This is a norm that says that every relay must be there twice, either in parallel for Normally Closed (NC) or in series for Normally Open (NO) modes, in order to ensure safety function at all time thanks to redundancy; therefore, it is ideal to use in the Lab. Moreover, in principle it is fully modular and can be fitted for almost any purpose.

The complete system is powered by an extra-low safety voltage (24V DC) source, including safety sensors to ensure no harm is done to operator while using any safety relevant parts.

Regarding the circuits, the Dynamic one is based on a single-channel safety concept where several safety components (e.g. sensors) can be connected in series and supervised with only one safety module. A dynamic signal is sent from Vital through all connected components, and then back to Vital, which evaluates the received signal. Since each safety component inverts the signal it is possible to detect short circuits or failures in a sensor.

The Static Safety Circuit is based on a clocked and a static (24V DC) signal, so normal components (like any switch, key or contactor with at least two contacts) can be integrated in the safety concept. Only if the clocked and the static signals are received from *Vital*, clearance for the static part is given. It is not possible for the *Vital* controller to determine whether there is a short circuit or not in the static part, but due to the two independent lines a fault can be detected (for example when a contact of the ground stick holder is not closed).

Vital 3 has three logic modes for the outputs to be set, as it can be seen in the following figures:

- **M1**: Output group one (controlled by the static part) and output group 2 (controlled by the dynamic part) are independent.

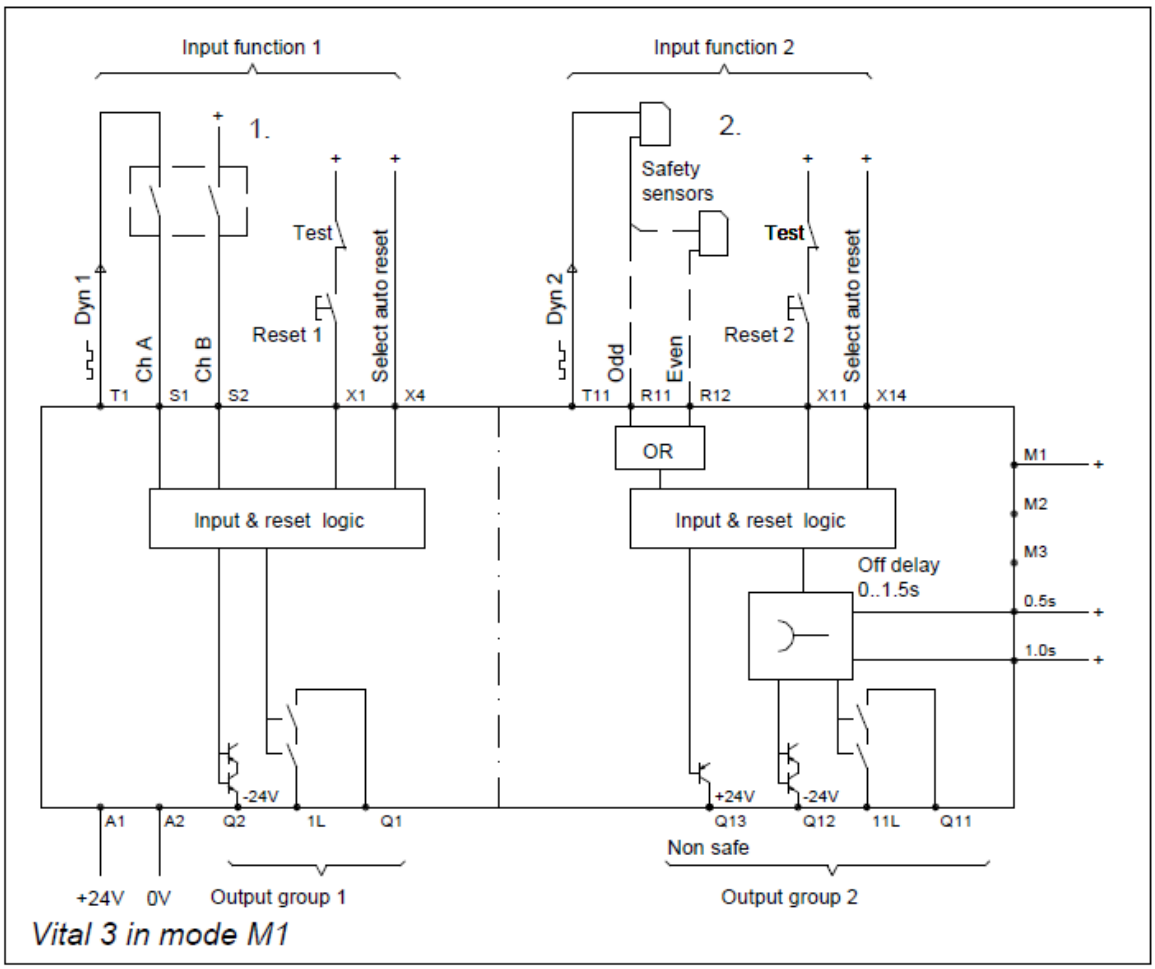


Fig. 16 *Vital 3* Safety Controller in M1 logic mode

- **M2**: Input group one has master control and input group two controls output group two only.

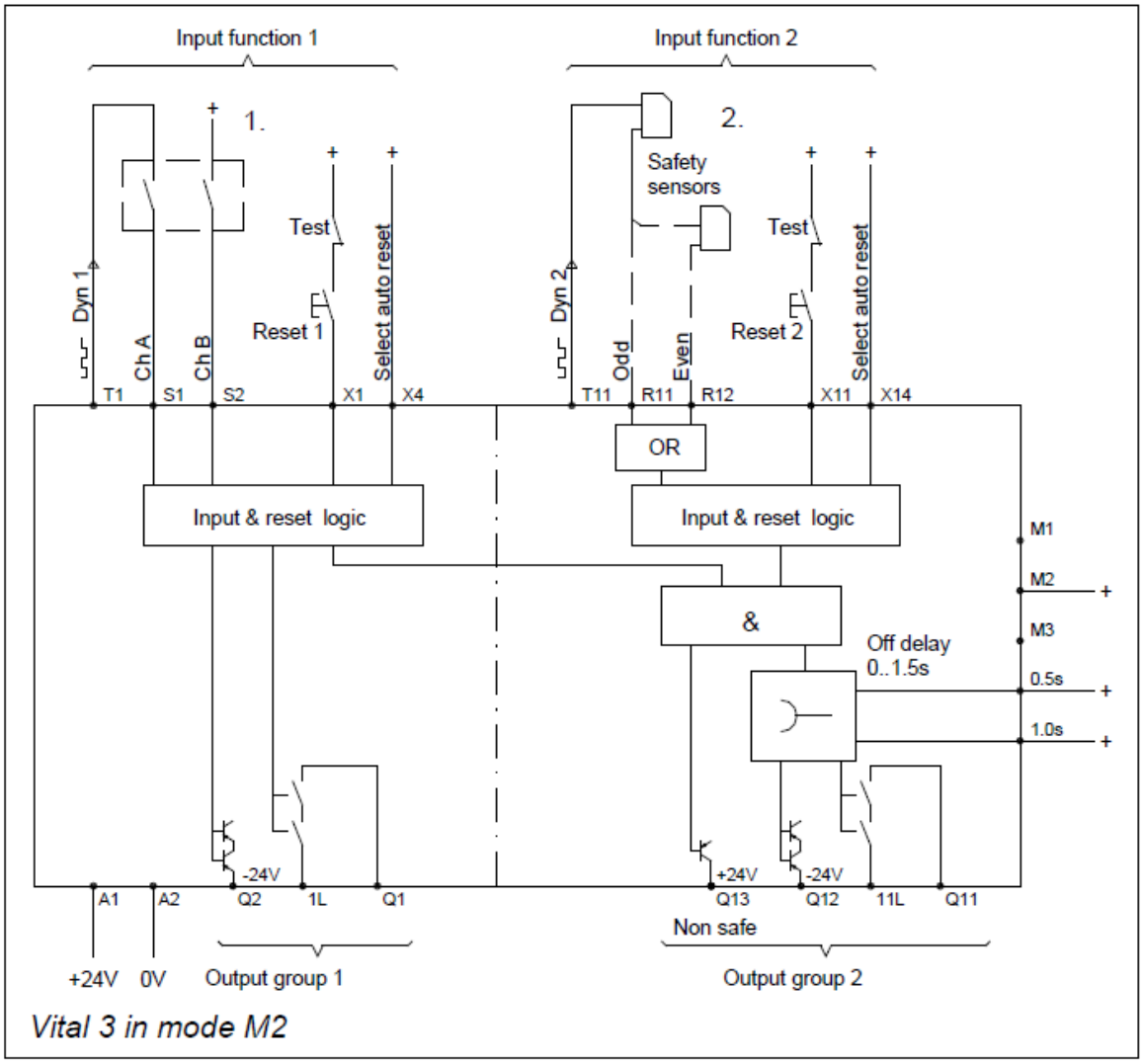


Fig. 17 *Vital 3* Safety Controller in M2 logic mode

- **M3**: Input group one and input group two are working in parallel and controlling both outputs.

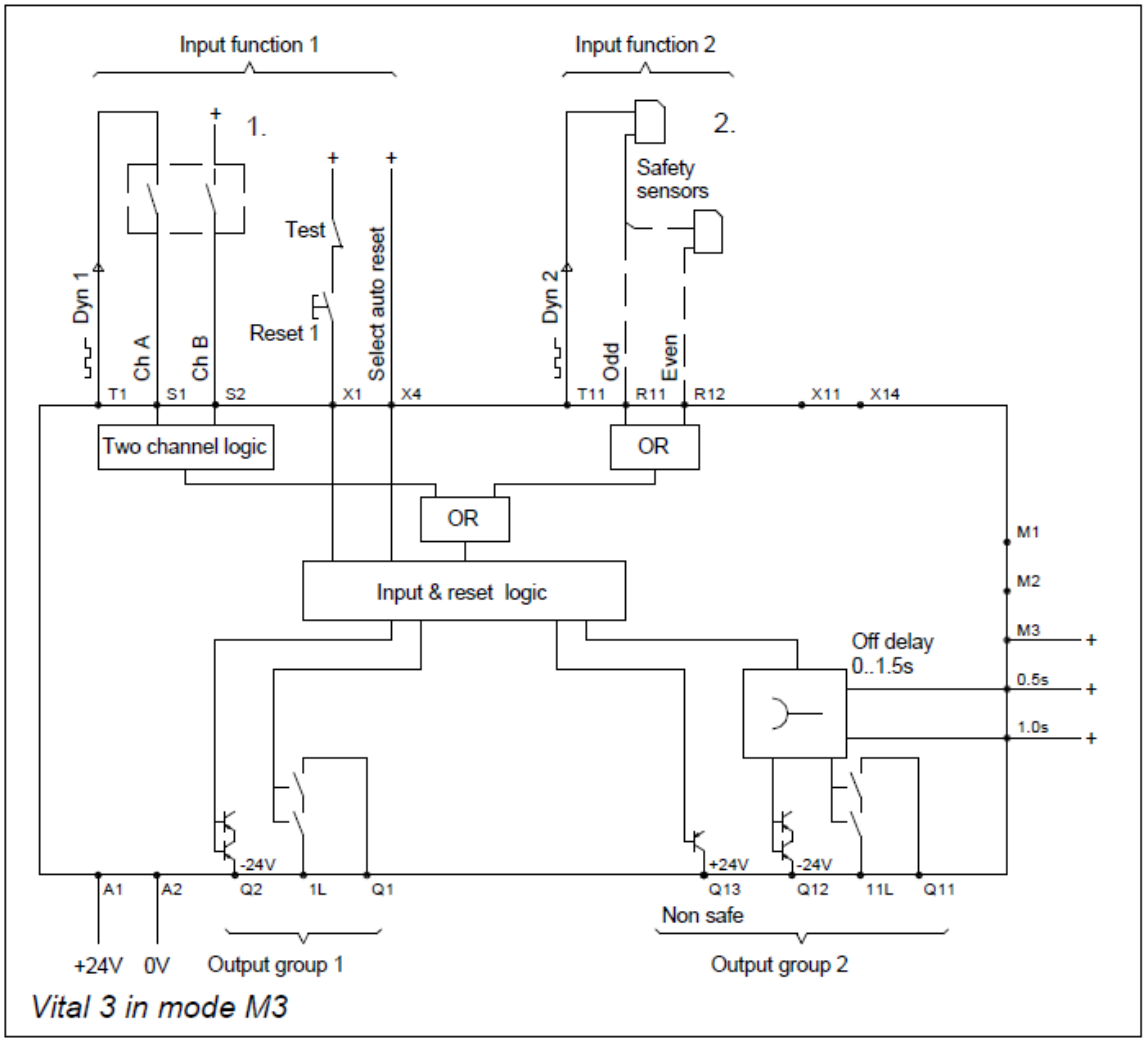


Fig. 18 *Vital 3* Safety Controller in M3 logic mode

Every safety sensor has to be able to shut down the system at any time, so the logical choice was to use mode **M2** for the new safety unit. Output group two was used to control all further safety functions.

Vital 3 controller was configured in auto reset mode. So that, after a safety feature is used (like for example emergency stop) it is possible to reclose the safety circuit without further notice. Therefore, X1, X4, X11 and X14 are connected to 24V.

Detailed information on the connections can be found in the complete schematics of the Safety Circuit in **Annex B: Schematics of the Safety Circuit**.

4.2.2. Safety relays

Safety relays are used to extend the existing outputs of the safety controller. Four kinds of different relays were built in:

- *Jokab Safety RT9*: Two NO contractors.
- *Jokab Safety RT6*: Three NO and one NC contractors.
- *ABB AL9-30-10*: Three NO and one auxiliary NO.
- *ABB AL40-30-10*: Three NO and one auxiliary NO.

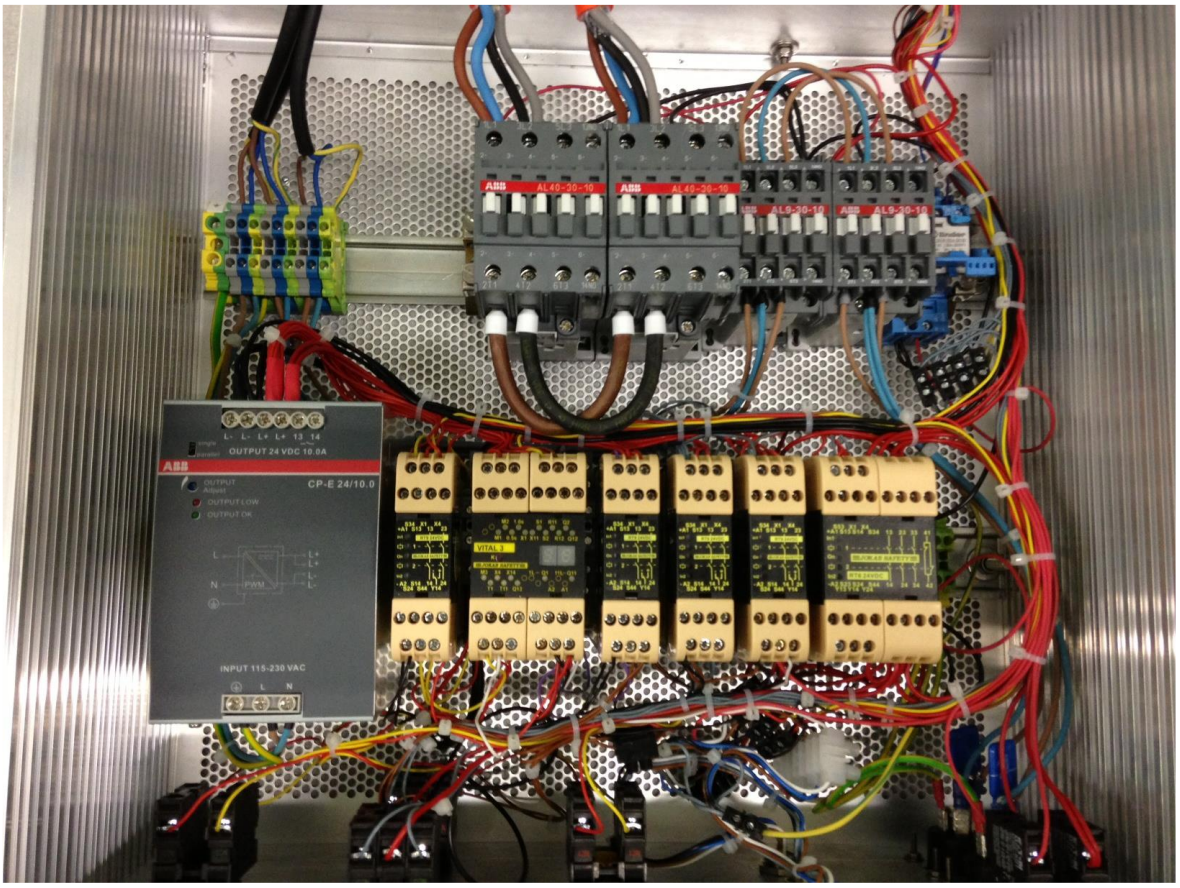


Fig. 19 Inner view of the Safety Control Unit

4.3. Main Control Unit (MCU)

The Main Control Unit (MCU) consists in a Programmable Logic Controller (PLC) that manages everything that can be regulated in the HV Setup in every moment: voltage, frequency, Spark Gap or Ground Switch. It is also in charge of detecting breakdowns, and has some safety features to disconnect the power source if anything goes wrong during an experiment. The PLC can set these parameters when received the specific commands through the Console; or it can also run some sequential experiments when ordered so through the Touch Panel, as explained before in **Point 3.2**.

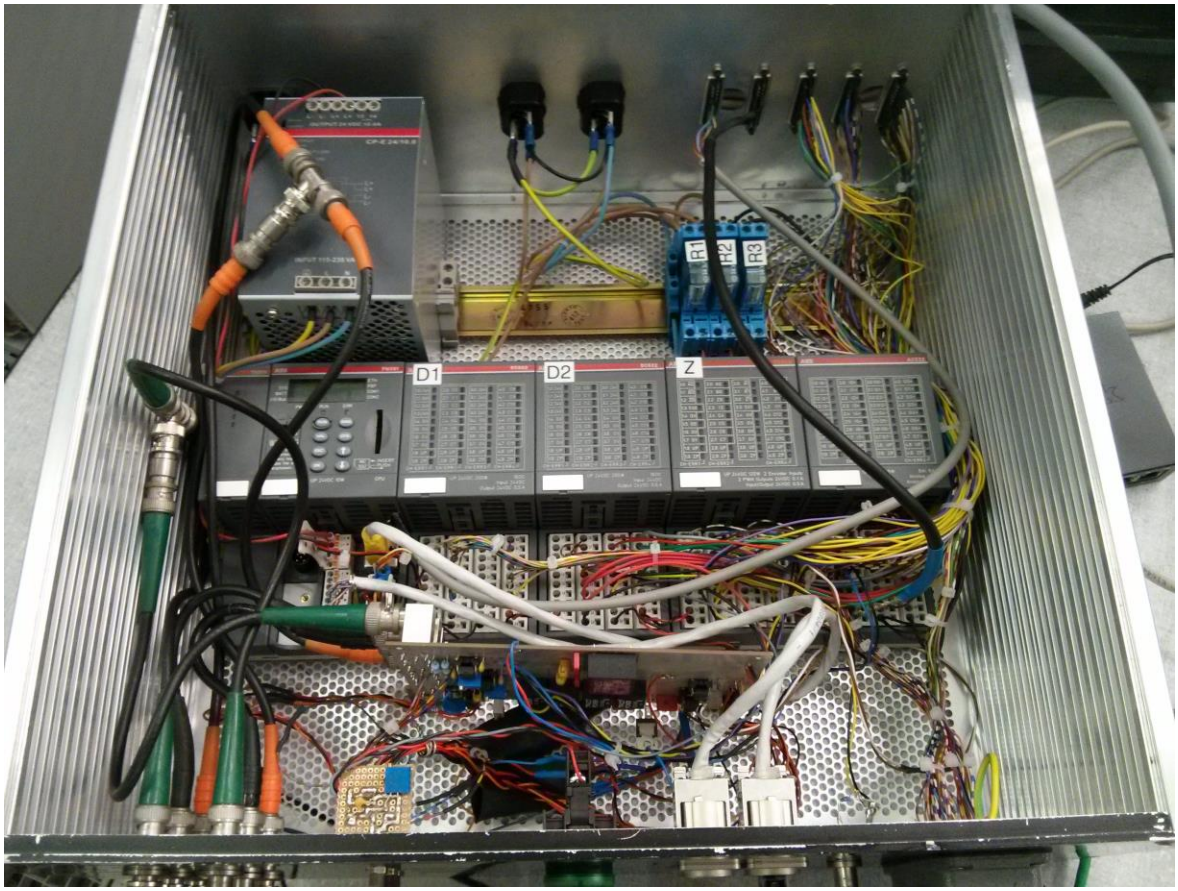


Fig. 20 Inner view of the MCU, with all the modules of the PLC, its power source and the three relays to control the Spark Gap motor

A PLC is a digital computer used in industries for the automation of processes due to its real-time features, robustness and resistance to electrical noise, vibration and impacts. It is a modular device, with a CPU expanded by modules with inputs and outputs connected to the user interface or to the system to be controlled. It is programmed from a computer, and

after downloading the code to its non-volatile memory, the CPU scans it cyclically, reading the inputs and updating the outputs accordingly.

When choosing the device to control the Lab, a PLC seemed the best option, beating other candidates like a microcontroller, a computer with *LabView* or a relay circuit. Among its ideal characteristics there were its fast reaction time, capacity to work with digital and analogic signals, modularity (easiness to replace or add parts), long life period or the fact of being a quality standard in industry. The fact that a PLC doesn't hang and doesn't allow user change (without connecting it to a PC to program it), made it definitely more convenient for the desired task than a more complex device like a computer.

The PLC used to control the lab is an ABB *AC500*, a scalable PLC which provides flexibility thanks to the different modules that can be chosen to complement the CPU and adapt to any specific needs. All the modules of the PLC are set in a rack inside the MCU metal case, and powered by a 24V DC power supply installed next to them. This PLC is programmed from a computer with a proprietary software from *ABB* but based in *CoDeSys*.

Fig. 21 Programmable Logic Controller ABB *AC500*

As CPU module for the PLC, *PM591-ETH* was chosen. It's one of the fastest CPUs for the *AC500* (minimum cycle time per instruction between 2 and 4 nanoseconds), and its main

characteristics are 4 MB of memory for the User Program, slot for SD Card, 2 serial ports, *FieldBusPlug* (FBP), and on-board Ethernet connection.

For the inputs and outputs, four modules were added to the CPU: two digital ones, one analogic, and an encoder module. The digital ones are two pieces of the model ABB *DC522*, consisting each of them in 16 digital configurable IN/OUTs of 24V DC and 0.5A, and labelled as *D1* and *D2*. The analogical one is an ABB *AX522*, with 8 inputs and 8 outputs configurable in different ranges of voltage or current, labelled as *A*. Finally, the encoder module is an ABB *CD522*, which features two encoders, two PWM outputs, and 8 24V DC digital IN/OUTs, and is labelled with the letter *Z*.

Apart from the PLC, inside the housing of the Main Control Unit there are also three relays, labelled as *R1*, *R2* and *R3*, as it can also be seen in **Fig. 20**. They are in charge of controlling the Ground Switch and turning the motor of the Spark Gap to one direction or the other, when activated by the PLC. There are also some PCBs in order to pre-process some signals from the HV Setup before entering them into the PLC or sending them out.

Fig. 22 Front panel of the MCU, with the connections from the HV Setup and to the devices forming the User Interface, together with the LED lighting when the PLC is on

The MCU has connections in both its front and back panels. In the front one there are, as it can be seen in **Fig. 22**, the inputs for the AC and DC Dividers and the Current Probe, together with its outputs to send the signals to the Oscilloscope; the optical fiber cables to control the motor of the Spark Gap and trigger it; the Ethernet ports to connect to the Ethernet Switch and to control the Oscilloscope; an external trigger; and a MultiPole cable connection to the Console.

In the back panel there are the different ports connecting with the Safety Circuit, AC Source and the HV Setup, that will be explained in detail later in **Point 4.7**.

4.4. User Interface

The user interface is composed by three elements. Two of them, the Console and the Touch Panel, are the Human Machine Interface (HMI) to control the PLC and, through it, the voltage applied to the HV Setup. The third one is a digital oscilloscope that helps to monitor, record and analyze the data from the experiments.

4.4.1. Console

The Console is the custom-made device allowing to set the parameters of the PLC manually and so control the HV Setup and run the experiments in the Lab. It is built with pulse buttons, displays, a three-state selector and a turning wheel. All this is connected with the PLC and Safety Circuit through a self-made multipole cable that goes to the front of the MCU.

Fig. 23 The Console to manually control the Lab

The pulse buttons are used to activate and deactivate digital inputs in the PLC, and to close or open the safety circuit. With the LEDs they have inside, they can also show the state of

the safety circuit and some digital outputs from the PLC. The displays show the desired or current parameters from the HV Setup, as voltage or frequency. The three-state selector sets two digital variables that then select the number of stages of the setup through a piece of code in the PLC. And, finally, the turning of the selection wheel is captured by the encoder in the PLC and then used to set the different parameters like voltage and frequency.

Later, in **Point 5.1**, it will be explained in more detail what every button does and how to use the Console to control the Lab.

4.4.2. Touch Panel

The Operator Terminal or Touch Panel installed to run the automated test sequences is an ABB *CP435T-ETH*. It features a 7 inches and 64k colors TFT touch screen, with a resolution of 800 x 480 pixels. Among other characteristics it has 8 MB of flash memory, an integrated Real Time Clock (RTC), and possibility to store data in a Compact Flash card. Regarding communication interfaces with the controllers, it can be connected via RS232, RS485, RS422 and Ethernet (which is the one chosen in the lab for the communications with the PLC and also for programming it from the computer). It also has 3 USB ports (2 hosts and 1 device), to attach a printer or a USB drive, for example. It works with 24V DC of power supply.

Fig. 24 Touch Panel ABB *CP435T-ETH*

This touch panel is prepared to directly import and read the variables from a big amount of PLCs in the market, and it features Macros and Ladder programming also, allowing to run some parts of code directly on it instead of having to do it from the controller. The programming of the panel is done from a PC through a specific proprietary software called *CP400Soft*, which includes many different fonts, designs and GIFs to personalize the appearance of the screens.

4.4.3. Oscilloscope

The oscilloscope installed in the lab is a digital model from LeCroy, the *Waverunner LT364*. Among its main technical specifications, it can display four external channels and has the ability to reach a sample rate of 1 GS/s and a memory length up to 500 kpoints per channel. It also features different modes for triggering, including edge, window or smart triggers; or the possibility to divide the screen in different sections for a sequence of captures.

Another important characteristic for the application in the Lab is the possibility to be totally remotely controlled through GPIB or RS232. Almost all the options that can be set with a button in the front panel have an equivalent command to be set remotely.

Finally, regarding storage, it has the possibility to print the displayed waveforms, and also to store the displayed data in a Compact Flash Card of maximum 512 MB.

Fig. 25 Front of the Oscilloscope LeCroy *Waverunner LT364*

Fig. 26 Front of the Oscilloscope LeCroy *Waverunner LT364*

4.5. AC Power Source

The device present in the Lab is a pulse width modulated AC Power Source by the company Pacific Power Source, more exactly the model *160ASX*. It has a rated power of 6000 VA and a frequency range of 15 to 1200 Hz. Its maximum output values are 132 V_{rms} and 48 A_{rms} from line to neutral, and 264 V_{rms} with 16 A_{rms} between line and line.

However, in the Lab it is limited to set a frequency between 40 and 300 Hz, and with the current limited to 45 A. Between the AC Source and the HV Setup there is also a separation transformer with a 1:2 ratio, as a safety feature in order to completely separate the HV Setup from the control rack.

Like the oscilloscope, it can be controlled manually through the front panel, but also remotely through GPIB or RS232. This last mode plus the analogue input are the systems used to control it from the PLC.

Fig. 27 Pacific Source *160ASX* AC Power Source

4.6. Grounding and shielding

In the last months, a renovation of the lab has been carried out in order to fully shield it and so increase the safety when reaching the highest possible voltages and, at the same time, reduce the noise in the signals to the lowest value.

To do so, the floor and walls of the lab were covered with 1 mm thick galvanized steel panels, all connected between them and to a metal grid installed to cover the ceiling and doors. Even some plates to cover the windows were built, so they can be opened and closed at the user's will. This converted the lab in a totally metal surrounded Faraday cage, connected to the Earth pole in the lab.

Apart from the separation transformer between the AC Source and the HV Setup for Safety reasons, another one was set between the power line and the control rack (connected to the earth pole in the Lab) in order to avoid ground loops. Finally, to reduce electromagnetic disturbance in the control rack and noise in the signals, a filter was set in the power input of the rack and all the cables going in and out of it were fully shielded.

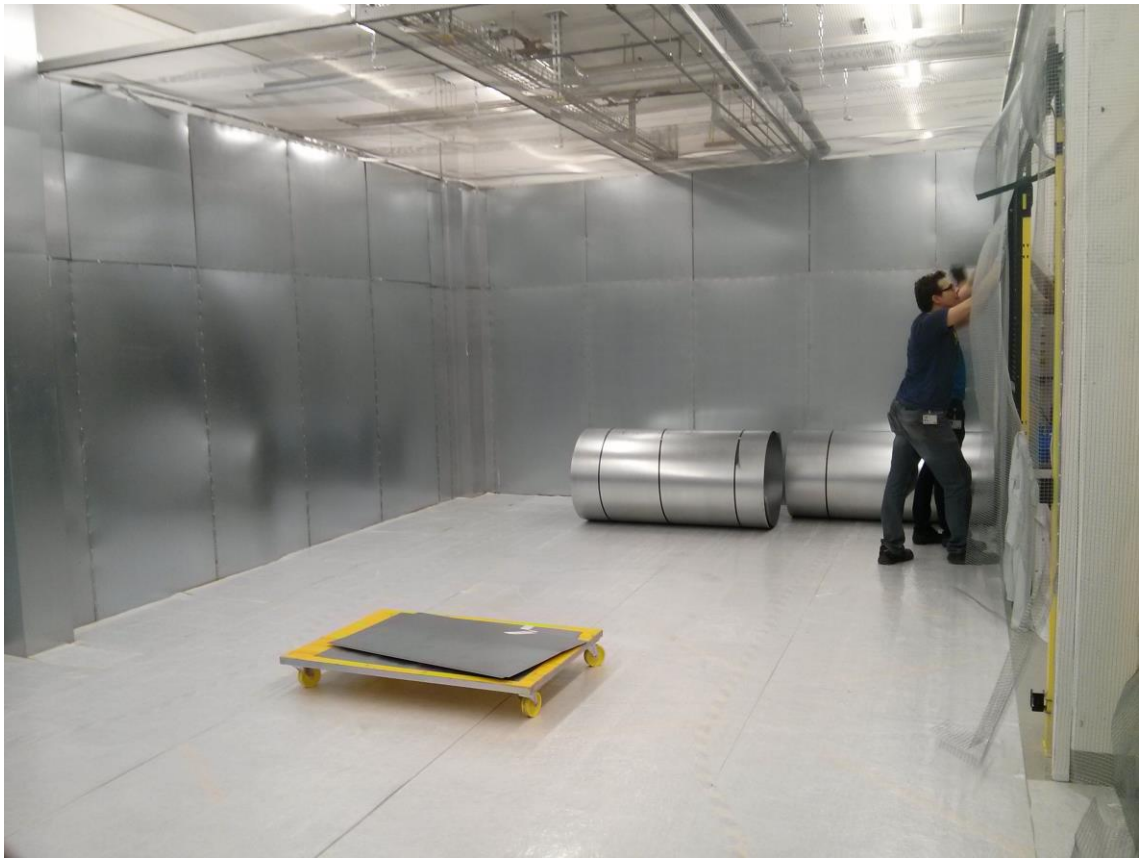


Fig. 28 Work to upgrade ML9901 Lab and fully shield it

4.7. Cables and connection ports

Many of the cables and connections in the Lab are self-made, in order to get the desired signals to be carried to a certain place and some specific features like shielding or full safety even if a cable is connected in a wrong port. So that, following there are the descriptions of the different customized ports and cables present in the Lab.

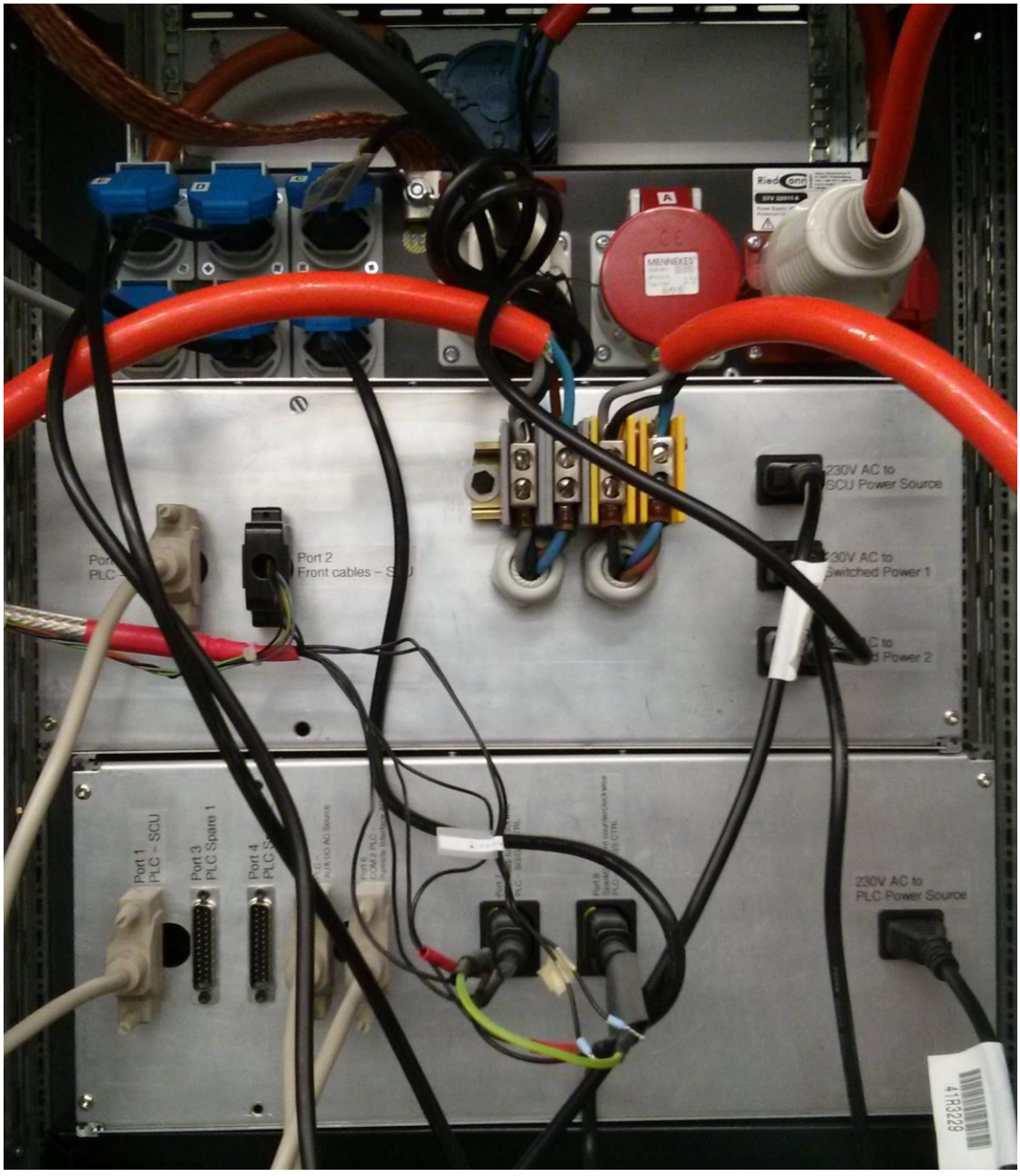


Fig. 29 Back panels of the SCU (top) and MCU (bottom)

4.7.1. Connection Ports in the Control Rack

The Control Rack has a set of ports labelled from 1 to 9, mainly coming from the Main Control Unit or the Safety Circuit, as it can be seen in **Fig. 29** Back panels of the SCU (top) and MCU (bottom), showing the back of the MCU and the SCU.

Ports 1 to 6 are D-Sub 25 connectors. Port 1 goes from the back of the MCU to the back of the SCU, carrying signals from the PLC and the multipole cable coming from the Console. Port number 2 brings some cables from the front of the Rack to the back of the SCU. Ports 3 and 4 go out of the back of the MCU containing the spare inputs and outputs not currently used in the PLC. In order to control the AC Source, ports 5 and 6 connect the remote control ports in the back of this one with the PLC through the back of the MCU.

Ports 7 and 8 are two 3-pin power connectors in the back of the MCU that go to the HV Setup in order to power and control the motor of the Spark Gap. Finally, port number 9 is the only one going out of the MCU through its front panel. It is an Ethernet type connection coming from the COM1 port in the PLC and going to the Remote Interface port of the Oscilloscope, in order to control this one from the PLC through RS232 interface.

The tables in the following pages describe the pin assignation in each of the ports labelled from 1 to 9 inside the control rack. They show the function of each specific pin, where does it come from, where does it go, and the colors of the cables inside. For some specific locations inside the MCU it may be useful to go back to **Fig. 20** , where the different modules of the PLC are shown.

Table 1 Control Rack Port 1 (MCU to SCU)

Port 1 - MCU to SCU (D-Sub 25 male back of MCU / D-Sub 25 female back of SCU)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color SC side**	**Going to**
1.1	GND for Ground Switch 2 (connected to GND)	R3 (Pin 24 NO)	Black	Black	S.C.
	GND	PLC A_ZP	Black	Black	Port 2.5 (S.C. side)
1.2	Safety light Green	Multipolecable Pin 4	Blue	Red	S.C.
	Safety light Monitoring Green	PLC D2_C3	Green		
1.3	Safety light Red	Multipolecable Pin 5	Pink	Red	S.C.
	Safety light Monitoring Red	PLC D2_C4	Red		
1.4	PLC Failsafe 1	PLC D1_C15	Yellow	Yellow	S.C.
1.5	PLC Failsafe 2	PLC D2_C15	Yellow	Yellow	S.C.
1.6	Safety Status	PLC D2_C2	White	White	S.C.
1.7	Safety Button Green Return 1	Multipolecable Pin 8	Green	Brown	S.C.
1.8	Safety Button Green Send 2	Multipolecable Pin 7	Yellow	Brown	S.C.
1.9	Safety Button Green Send 1	Multipolecable Pin 6	Grey	Brown	S.C.
1.10	Safety Button Red Send 1	Multipolecable Pin 10	White	Grey	S.C.
1.11	Safety Button Green Return 2	Multipolecable Pin 9	Brown	Brown	S.C.
1.12	Safety Button Red Return 1	Multipolecable Pin 12	White-Pink	Grey	S.C.
1.13	Safety Button Red Send 2	Multipolecable Pin 11	White-Blue	Grey	S.C.
1.14	Safety Button Red Return 2	Multipolecable Pin 13	White-Grey	Grey	S.C.
1.15	Static Safety Circuit Return 24V+	Multipolecable Pin 17	Green-Black	Red	S.C.
1.16	Static Safety Circuit Return	Multipolecable Pin 16	Grey-Red	Yellow	S.C.
1.17	Static Safety Circuit Send 24V+	Multipolecable Pin 15	Yellow-Red	Red	S.C.
1.18	Static Safety Circuit Send	Multipolecable Pin 14	Green-Red	Yellow	S.C.
1.19	Grounding return 2	R3 (Pin 21 COM)	Black	Black	Port 2.6 (S.C. side)

Table 2 Control Rack Port 2 (Front cables to SCU)

Port 2 - Front cables to SCU (D-Sub 25 male from LED Stripe and GS/SG CTRL / D-Sub 25 female back of SCU)					
Port & Pin	**Function**	**Coming from**	**Cable Color LED Stripe & GS/SG CTRL side**	**Cable Color SC side**	**Going to**
2.1	Safety-Ground-Switch Connection	GS/SG CTRL plug	Black	Red	S.C.
2.2	LED-Stripe 24V+	Front LED stripe	Grey	Red	S.C.
2.3	LED-Stripe Green return	Front LED stripe	Green	Purple	S.C.
2.4	LED-Stripe Red return	Front LED stripe	Red	Blue	S.C.
2.5	GND	Front LED stripe	Black	Black	Port 1.1 (S.C. Side)
2.6	Grounding return 2	GS/SG CTRL plug	Black	Black	Port 1.19 (S.C. Side)

Table 3 Control Rack Port 3 (Spare pins from PLC 1)

Port 3 - Spare pins from PLC 1: IN/OUT from digital D2 and analog A modules (D-Sub 25 male from back of MCU)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color out side**	**Going to**
3.1	PLC digital IN/OUT	PLC D2_C8	Grey	-	Free
3.2	PLC digital IN/OUT	PLC D2_C9	Green	-	Free
3.3	PLC digital IN/OUT	PLC D2_C10	White	-	Free
3.4	PLC digital IN/OUT	PLC D2_C11	Purple	-	Free
3.5	PLC analog IN	PLC A_I3 -	Pink-Black	-	Free
3.6	PLC analog IN	PLC A_I3 +	Red-White	-	Free
3.7	PLC analog IN	PLC A_I4 -	Pink-Blue	-	Free
3.8	PLC analog IN	PLC A_I4 +	Black-White	-	Free
3.9	PLC analog IN	PLC A_I5 -	Pink-Red	-	Free
3.10	PLC analog IN	PLC A_I5 +	Grey-Blue	-	Free
3.11	PLC analog IN	PLC A_I6 -	Brown-Grey	-	Free
3.12	PLC analog IN	PLC A_I6 +	Brown-Green	-	Free
3.13	PLC analog IN	PLC A_I7 -	Black-Red	-	Free
3.14	PLC analog IN	PLC A_I7 +	Yellow-Black	-	Free
3.15	PLC analog OUT	PLC A_O3 -	Yellow-Brown	-	Free
3.16	PLC analog OUT	PLC A_O3 +	Yellow-White	-	Free
3.17	PLC analog OUT	PLC A_O4 -	Pink-Brown	-	Free
3.18	PLC analog OUT	PLC A_O4 +	Green-White	-	Free
3.19	PLC analog OUT	PLC A_O5 -	Pink-Green	-	Free
3.20	PLC analog OUT	PLC A_O5 +	Pink-Grey	-	Free
3.21	PLC analog OUT	PLC A_O7 -	Brown-Blue	-	Free
3.22	PLC analog OUT	PLC A_O7 +	Yellow-Pink	-	Free
3.23	PLC 0V	PLC D2_0V (3.5)	Black	-	Free
3.24	PLC 24V	PLC D2_24V	Red	-	Free
3.25	PLC 5V	PLC Z_5V1	Blue	-	Free

Table 4 Control Rack Port 4 (Spare pins from PLC 2)

Port 4 - Spare pins from PLC 2: IN/OUT from encoder Z module (D-Sub 25 male from PLC)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color out side**	**Going to**
4.1	Aneg IN encoder 0	PLC Z_A0neg	White-Black-Green	-	Free
4.2	Bneg IN encoder 0	PLC Z_B0neg	Yellow-Green-black	-	Free
4.3	Zneg IN encoder 0	PLC Z_Z0neg	Blue-Red-Black	-	Free
4.4	Z IN encoder 0	PLC Z_Z0	Brown-Red	-	Free
4.5	I IN encoder 0	PLC Z_I3	Green-Blue	-	Free
4.6	Aneg IN encoder 1	PLC Z_A1neg	Brown-White-Black	-	Free
4.7	Bneg IN encoder 1	PLC Z_B1neg	Green-Brown-Black	-	Free
4.8	Zneg IN encoder 1	PLC Z_Z1neg	Grey-Pink-Black	-	Free
4.9	Z IN encoder 1	PLC Z_Z1	Brown-Black	-	Free
4.10	I IN encoder 1	PLC Z_I11	White	-	Free
4.11	PLC digital IN/OUT	PLC Z_C4	Yellow	-	Free
4.12	PLC digital IN/OUT	PLC Z_C5	Yellow	-	Free
4.13	PLC digital IN/OUT	PLC Z_C6	Yellow	-	Free
4.14	PLC digital IN/OUT	PLC Z_C7	Yellow	-	Free
4.15	PLC digital IN/OUT	PLC Z_C12	Yellow	-	Free
4.16	PLC digital IN/OUT	PLC Z_C13	Yellow	-	Free
4.17	PLC digital IN/OUT	PLC Z_C14	Yellow	-	Free
4.18	PLC digital IN/OUT	PLC Z_C15	Yellow	-	Free
4.19	PWM/Pulse OUT	PLC Z_O0	Purple	-	Free
4.20	PWM/Pulse OUT	PLC Z_O1	Purple	-	Free
4.21	External Trigger +	PLC D2_C12	Green	-	Free
4.22	PLC 0V	PLC Z_0V	Black	-	Free
4.23	PLC 24V	PLC D2_24V	Red	-	Free
4.24	PLC 5V	PLC Z_5V1	Blue	-	Free

Table 6 Control Rack Port 5 (MCU to AUX I/O AC Source)

Port 5 - MCU to AUX I/O AC Source (D-Sub 25 male from back of MCU)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color AC Source side**	**Going to**
5.8	REGELSPANNUNG_ANALOG +	PLC A_O6+	Black	-	AUX I/O AC Source
5.9					AUX I/O AC Source
5.10					AUX I/O AC Source
5.20	REGELSPANNUNG_ANALOG -	PLC A_O6-	Black	-	AUX I/O AC Source
5.21					AUX I/O AC Source
5.22					AUX I/O AC Source

Table 5 Control Rack Port 6 (COM2 to Remote Interface AC Source)

Port 6 - COM2 PLC to Remote Interface AC Source (D-Sub 25 male from back of MCU					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color AC Source side**	**Going to**
6.2	PLC COM2 TxD	PLC COM2 Pin 2	Red	-	Remote Interface AC Source
6.3	PLC COM2 RxD	PLC COM2 Pin 7	Purple / Orange	-	Remote Interface AC Source
6.4	COM2 GND	PLC D1_ZP (3.9)	Black / Pink	-	Remote Interface AC Source
6.5			Black / Light Blue	-	Remote Interface AC Source
6.6			Black / Light Green	-	Remote Interface AC Source
6.7	PLC COM2 SGND	PLC COM2 Pin 5	Green / Yellow	-	Remote Interface AC Source
6.8	COM2 GND	PLC D1_ZP	Black / Brown	-	Remote Interface AC Source
6.20	COM2 GND	PLC D1_ZP	Black / Dark Blue	-	Remote Interface AC Source
6.22			Black / Dark Green	-	AUX I/O AC Source

Table 7 Control Rack Port 7 (SG motor turn clockwise)

Port 7 – Spark Gap motor turn clockwise (3-pin female from back of MCU / 3-pin male from SG/GS CTRL Plug)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color SG/GS CTRL side**	**Going to**
7.1 (L)	230V clockwise Spark Gap motor	R1 (Pin 24 NO)	Brown	Black	GS/SG CTRL Plug
7.2 (N)	Neutral	24V Power Supply N	Blue	Black	GS/SG CTRL Plug
7.3 (GND)	GND	24V Power Supply GND (through 8.3)	Black / Black	Black	GS/SG CTRL Plug

Table 8 Control Rack Port 8 (SG motor turn counterclockwise)

Port 8 - Spark-Gap motor turn counter-clockwise (3-pin female from back of MCU / 3-pin male from SG/GS CTRL Plug)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color SG/GS CTRL side**	**Going to**
8.1 (L)	230V counter-clockwise Spark Gap motor	R2 (Pin 24 NO)	Brown	Black	GS/SG CTRL Plug
8.2 (N)	Neutral	24V Power Supply N (through 7.2)	Blue / Black	Black	GS/SG CTRL Plug
8.3 (GND)	GND	24V Power Supply GND	Black	Black	GS/SG CTRL Plug

Table 9 Control Rack Port 9 (COM1 PLC to Remote Interface Oscilloscope)

Port 9 - COM1 PLC to Remote Interface Oscilloscope (Ethernet Cat-5 Cross-Over female from PLC)					
Port & Pin	**Function**	**Coming from**	**Cable Color PLC side**	**Cable Color Oscilloscope side**	**Going to**
Not Connected	PLC COM1 Term. P (RS-485)	PLC COM1 Pin 1	Not Connected	-	-
9.1	PLC COM1 RxD/TxD-P (RS-485)	PLC COM1 Pin 2	Green-White	Green-Black	-
9.2	PLC COM1 RxD/TxD-N (RS-485)	PLC COM1 Pin 3	Green	Green	-
9.3	PLC COM1 Term. N (RS-485)	PLC COM1 Pin 4	Orange-White	Orange-Black	-
9.4	PLC COM1 RTS (RS-232)	PLC COM1 Pin 5	Blue	Blue	Pin 8 RS-232C Oscilloscope (CTS)
9.5	PLC COM1 TxD (RS-232)	PLC COM1 Pin 6	Blue-White	Blue-Black	Pin 2 RS-232C Oscilloscope (RxD)
9.6	PLC COM1 SGND	PLC COM1 Pin 7	Orange	Orange	Pin 5 RS-232C Oscilloscope (GND)
		PLC D1_ZP	Black		
9.7	PLC COM1 RxD (RS-232)	PLC COM1 Pin 8	Brown-White	Brown-Black	Pin 3 RS-232C Oscilloscope (TxD)
9.8	PLC COM1 CTS (RS-232)	PLC COM1 Pin 9	Brown	Brown	Pin 7 RS-232C Oscilloscope (RTS)

4.7.2. Speakon cable

Connections to bring power and some control functions to the Oscilloscope, Spark Gap motor, Ground Switches and Transformer are done through a customizable cable model *Speakon* from the company Neutrik. This connector is used with the following pin assignments depending on the size of it:

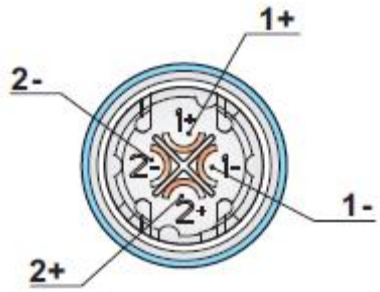


Fig. 31 Speakon cable in its 4 Pole version

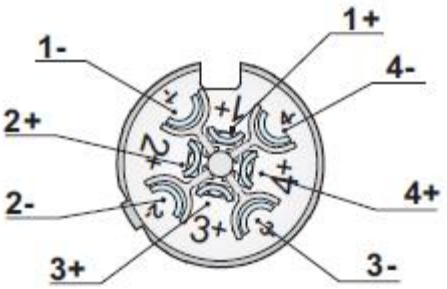


Fig. 31 Speakon cable in its 8 Pole version

Table 10 Pin assignment for the 8 Pole Function Speakon Connector

Pole	Color	Assignment
1+	Red	24V+
1-	Brown	230V right-hand rotation FS
2+	Black	Return Channel Grounding 1
2-	Blue	Neutral
3+	Green	Position Ground Switch 2
3-	Brown	230V left-hand rotation FS
4+	Grey	Position Ground Switch 1
4-	Black	Return Channel Grounding 2

Table 11 Pole assignment for the 4 Pole Safety Circuit Speakon connector

Pole	Color	Assignment
1+		GND
1-		N
2+	-	N
2-	Brown	230V

Table 12 Pole assignment for the 4 Pole Oscilloscope Speakon Connector

Pole	Color	Assignment
1+		N
1-	Brown	GND
2+	Blue	N
2-	-	230V

Table 13 Pole assignment for the 4 Pole Transformer Speakon Connector

Pole	Color	Assignment
1+	Brown	AC-Source OUT L
1-	Blue	AC-Source OUT L
2+	Black	AC-Source OUT N
2-	Grey	AC-Source OUT N

Remarks:

1) The shell is connected to GND and respectively connected to the connector shell (clamped).

2) Both the plug and sockets are held in the *Neutrik STX* version (Electronic parameters according to data sheet).

4.7.3. M12 cable for Safety Circuit

The Safety Circuit uses a specific self-made cable called *M12*, with the following pole assignation depending on being for the Static, Dynamic or Lighting circuit:

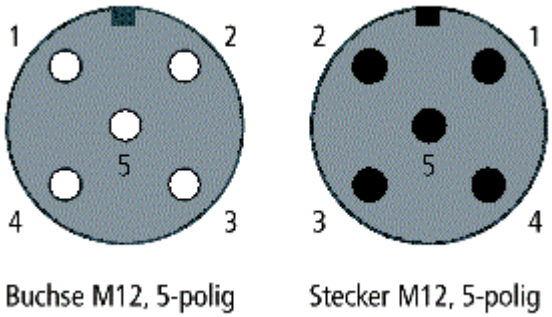


Fig. 32 M12 cable for the Safety Circuit

Table 14 M12 Plug/Socket assignment in the Safety Circuit

	Color in the Housing	Dynamic circuit	Static circuit	Lighting circuit
1	Brown	24V	24V	Green
2	White	SEND	SEND	Red
3	Blue	0V	RETURN 24V	0V
4	Black	RETURN	RETURN	N.C.
5	Grey	GND	GND	GND

5. How to operate the Lab

5.1. Manual mode

The manual mode of the lab is controlled through the black console seen in **Fig. 33**. The buttons are labelled with stickers showing what they are used for, but following there is a short explanation on the recommendable steps in order to use the lab in Manual mode:

Fig. 33 Console to control the Lab in Manual mode

1) First of all, after turning on the whole control rack and the PLC, the voltage form and number of stages in the circuit should be chosen. To do so, one has to press one of the buttons in the second column from the left, choosing either *AC*, *DC* or *Li/Si*; and move the stages selector under them to the desired number.

2) After the voltage form has been selected, the desired voltage and frequency can be set using the yellow buttons in the third column and the steering wheel below them. The top button (*10x*) is used to change the voltage or frequency in steps of 10, if pressed while adjusting them. The voltage set can be seen in the top left display, and the frequency in the top right one. These adjustments can be done at any other time later on as well.

3) Once the voltage form and frequency and voltage values have been set, is necessary to turn the Safety Status on before being able to apply voltage to the circuit. To do so, the safety circuit must be closed (including door of the Faraday Cage closed, ground stick on the holder of the door, and Safety key turned on in the control rack), before one can effectively press the *ON* red button in the first column from the left of the Console.

4) As soon as Safety Light is red, the user can start the operation by pressing the red *ON* button in the fourth column. At that moment, the desired voltage will be applied to the circuit, showing the real measured value in the top center display.

5) If in Impulse mode, as soon as the nominal voltage is reached, the yellow *Trigger* button in the right part of the Console will light up. From that moment it is possible to press it in order to trigger the Spark Gap, which should have moved automatically to the right distance as soon as *Li/Si* mode was selected and the nominal voltage was set.

This is the recommended sequence to start using the lab in Manual mode. Later on, the order can be altered, modifying voltage or frequency while Safety or Operation are on. To change the voltage form or the number of stages, however, Operation and Safety must be turned off.

To switch off the operation in any moment, *OFF* green button in the fourth column can be used. The green *OFF* button in the left side of the Console turns off the Safety Status, automatically turning off also the Operation and the AC Source and closing the ground switch (this can be dangerous when there is still a high voltage in the circuit, leading to a fast discharge through the ground switch with a loud noise). The big red button in the top left corner is also used to turn off the Safety Status, preventing it to be turned on again unless the button is set to its original position.

Finally, there is also a way implemented to move the Spark Gap manually using the Console. To do so, Impulse mode has to be selected but the setup must not be in Operation yet. Then, pressing at the same time *Trigger* and *10x* button the Spark Gap will move to the left; and pressing *Trigger* and *Set Frequency* it will move to the right.

5.2. Auto mode

The auto mode of the Lab is controlled through the touch panel. When in *Auto* mode, most of the buttons in the Console will be inactive, even if their LEDs will keep showing their status when needed. However, the buttons to set the Safety Status will keep reacting when pushed, as they are directly connected to the Safety Circuit and they are actually the only way to change the Safety Status (it is physically impossible to do it from the touch screen). The green button to turn off Operation is also implemented to keep on working, in order to be able to stop the sequence from there instead of the equivalent buttons in the touch panel.

All the screens in the panel follow the same graphical structure, to make it user friendly and easy to understand. As an example to explain this structure, the following screen will be used:

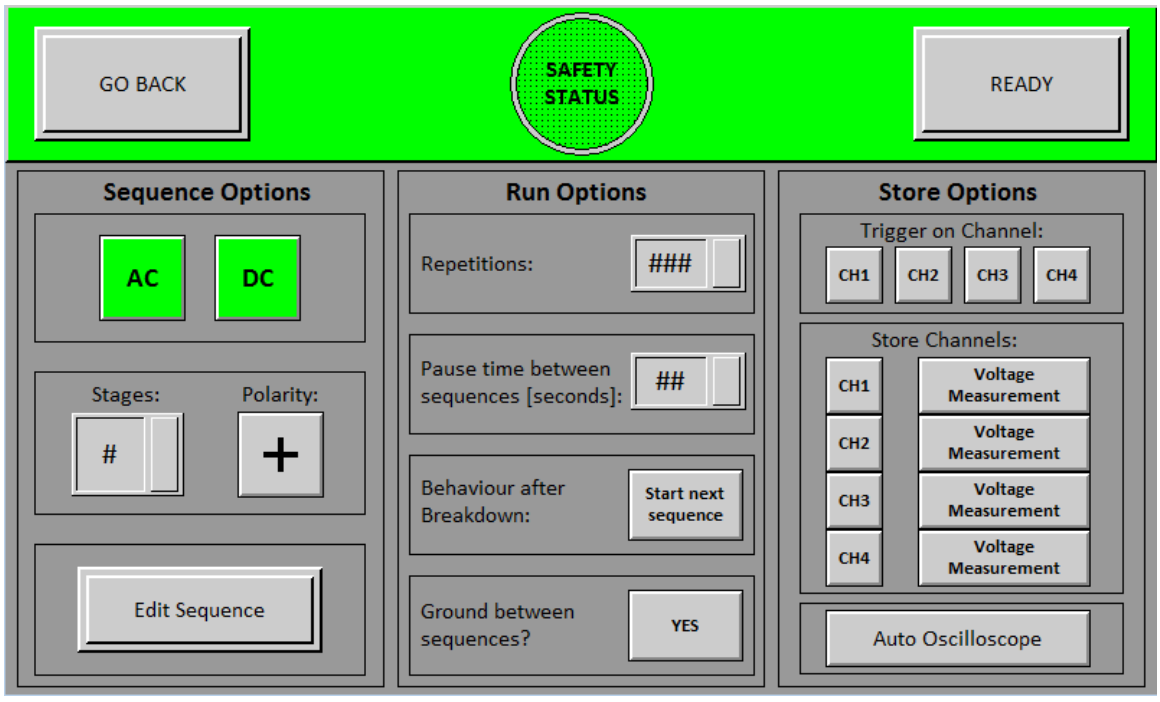


Fig. 34 Example of the interface in the Touch Panel

The screen is usually divided in the main grey area where the specific buttons and actions of each screen are performed; and a top layer in green or red. This top layer is present in almost all the screens in the touch panel, and gives information about the Operation Status of the Setup following the color code of the Lab: while it's green, the setup is not on Operation; but as soon as it becomes red it means Operation is on and so there is voltage

being applied to the setup. It also shows always the Safety Status (green for off and red for on). Depending on the screen there can be also more banners showing the status parameters of the sequence. Finally, in the top right corner there is usually the button used to go forward in the control; and in the top left the one used to go back to the previous screen.

Regarding the main grey area of the screen, it contains the specific parameters of each window, either to set or to show variables. It is structured in rectangles separating the parameters on the same field.

The shape of the buttons follows also a predefined pattern. The displays and numeric entries are done in the same layout, but with a pushable rectangle in the right part for the ones where a number can be introduced. The buttons with text inside can also be of two different types: buttons to select or set two-states parameters, which are represented by a rectangle that, when pushed, changes its text or color (usually grey when they are off and green when they are on); and buttons to perform an action or change the screen, represented by a grey rectangle getting out of a grey frame.

These are the main common parameters for all the screens, but each screen has its specific functionality. Following, there is a brief description of each of them, with explanations on what every button does and how to correctly use the touch panel to operate the lab.

5.2.1. "Manual/Auto" Screen

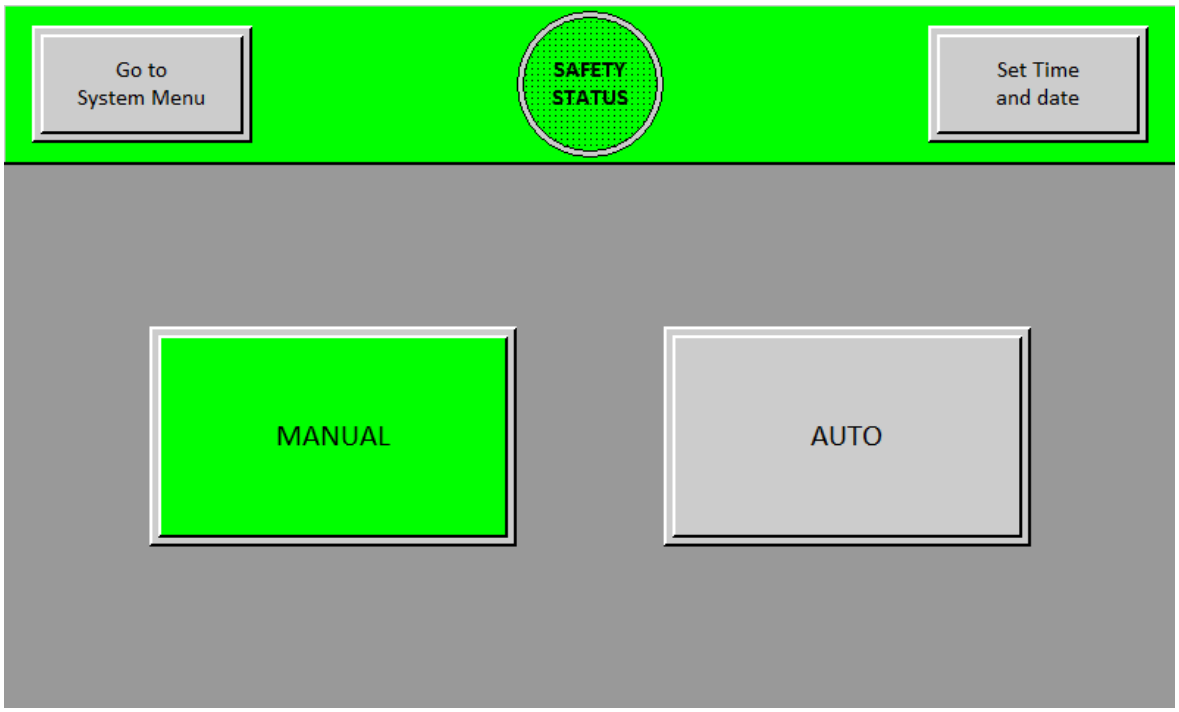


Fig. 35 TP screen to choose between Manual and Auto modes

This is the first screen to appear when the touch panel is switched on. It basically shows the common top bar with Safety and Operation Status, and it allows the user to choose between Manual and Automatic mode.

By default it is set in Manual, so if the user doesn't press anything, the Console can be used to control the HV Setup. If the user presses *Manual*, it will jump to the "Manual" screen; if *Auto* is pressed, it will set the Automatic mode and jump to another screen to start setting the automating sequence.

There are two more buttons in the top. *Go to System Menu* to go to the general settings of the touch panel (there one can choose the IP and other parameters for the Ethernet connection, the default port to download new applications, etc.). And *Set Time and date*, to set the time and date of the touch panel.

5.2.2. “Manual” screen

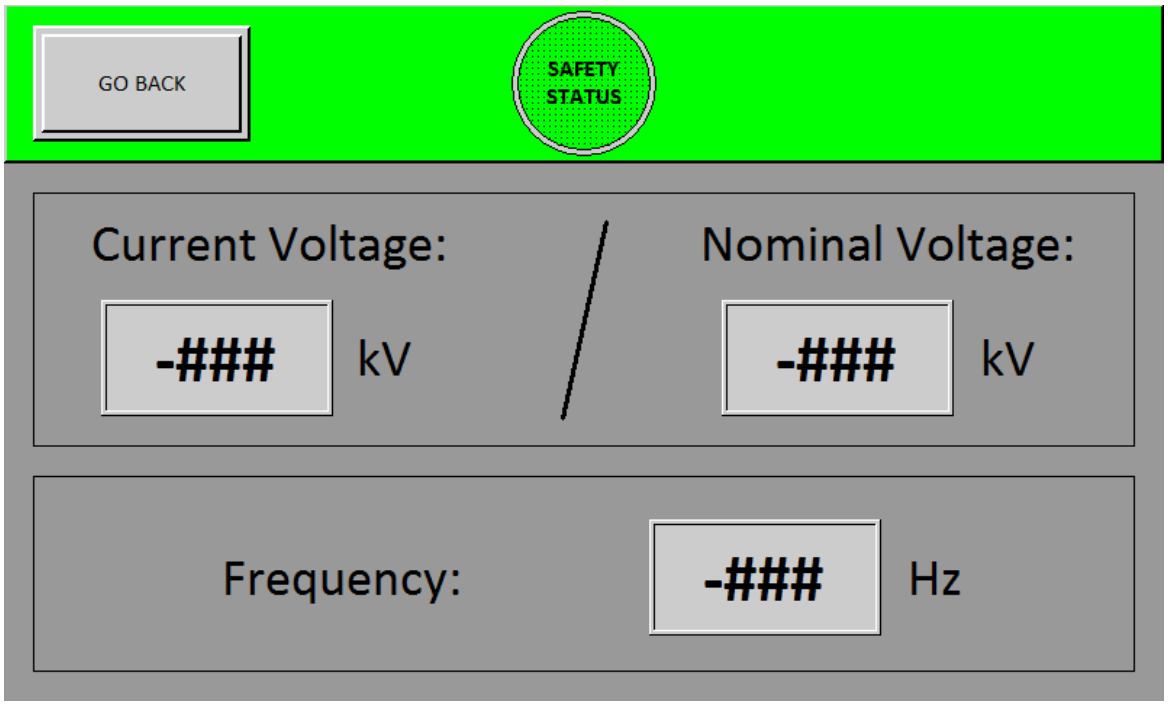


Fig. 36 TP screen for the Manual mode

This screen shows the voltage and the frequency in the circuit, apart from the Safety and Operation Statuses.

It has only the option to go back, as the touch panel is not used while in Manual mode. Everything is controlled through the Console.

5.2.3. "Auto Continuous/Impulse" screen

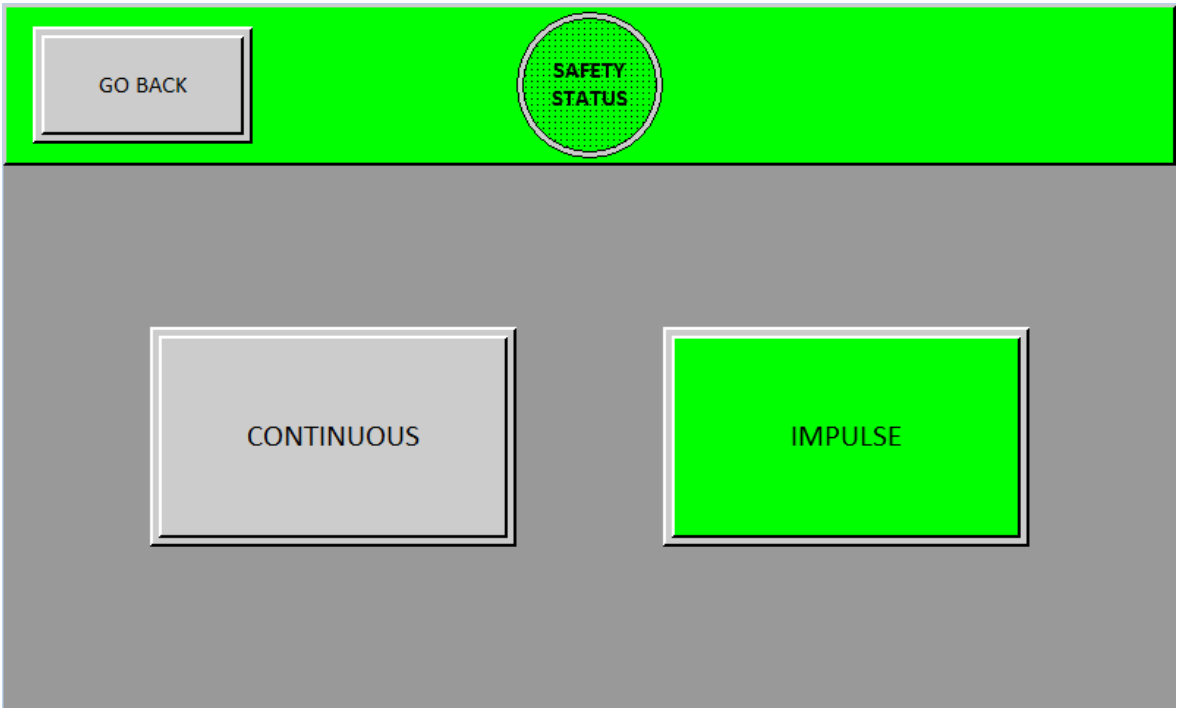


Fig. 37 TP screen to choose between Continuous and Impulse auto modes

This screen appears after pressing *Auto* in the first screen. It allows the user to choose between *Continuous* or *Impulse* automatic modes.

5.2.4. “Continuous Preparation” screen

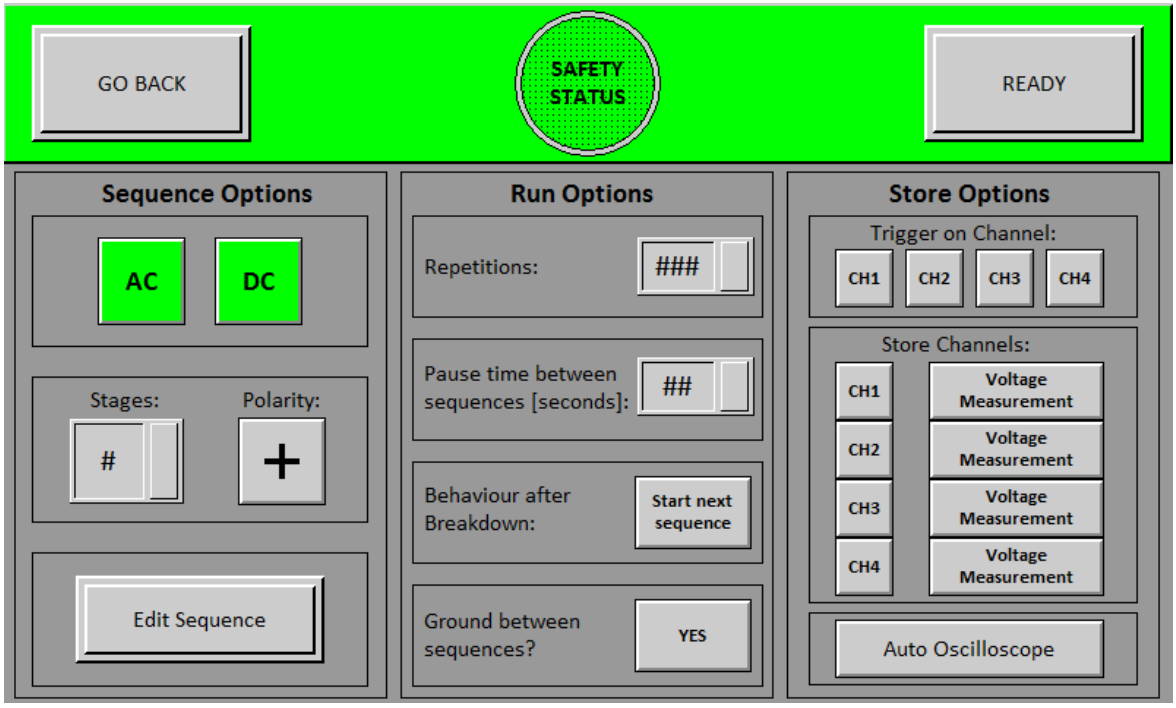


Fig. 38 TP screen to prepare the Auto Continuous mode

This is the screen that appears when *Continuous* mode chosen, and it is used to define the parameters for an Automatic Continuous sequence.

In the left part it allows to choose the options of the sequence, like *AC* or *DC* modes, or the number of stages and polarity of the circuit. The button *Edit Sequence* jumps to the “Sequence definition” screen, where one can set the shape of the sequence.

The options in the center of the screen set the options for how to run the sequence. This includes the number of repetitions intended of the defined sequence, the time to wait between the end of a repetition and starting the following one, the behavior after a breakdown, and if the setup needs to be grounded or not between two repetitions.

In the right side of the screen the options for the oscilloscope can be selected. First of all, the channel where to trigger has to be set. Then, one can enable or disable any of the four channels in the oscilloscope by pressing in the respective button; and select if the signal attached to it is a voltage or a current measurement by pressing the corresponding button next to each channel (to set different parameters when configuring the oscilloscope automatically). Finally, in the bottom, there is a button to select if the oscilloscope has to be set automatically (according to the parameters chosen above, and to the voltage form,

polarity and maximum voltage to be applied in the sequence); or if it has to be left to the user to set it.

When everything is correctly set, one can press the button *Ready* in the top right corner in order to set the oscilloscope and go to the screen to run the Automatic Continuous sequence.

5.2.5. “Edit Sequence” screen

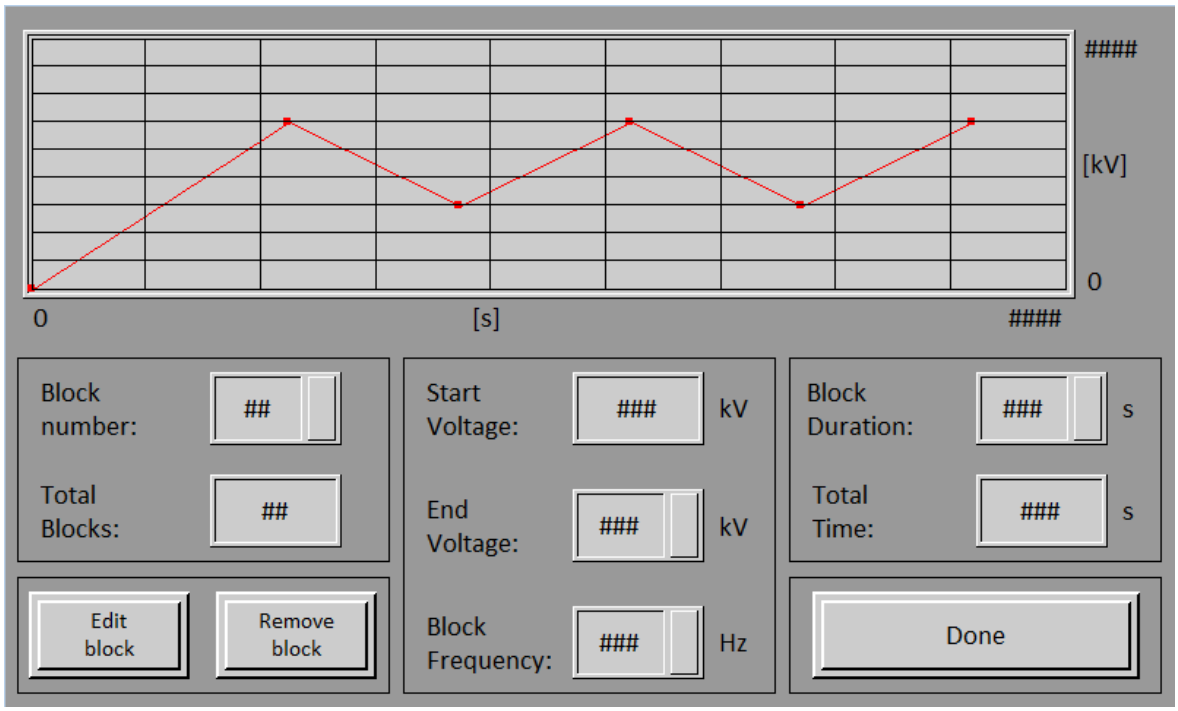


Fig. 39 TP screen to configure a Continuous sequence

This screen allows the user to define the sequence to be done in the lab when in an Automatic Continuous mode. This sequence is defined by a succession of blocks. Each block is specified by and end voltage, a frequency and a duration. The program will go from the end voltage of the previous block to the end voltage of the current block in the specified duration time.

The first time this screen is opened it allows to edit block number 1, with a start voltage of 0 kV. The user can change the end voltage, the block frequency (predefined to 50 Hz if not changed) and the duration of the block by pressing on the corresponding buttons. To store the changes and create the block, the button *Edit block* in the bottom left corner of the screen has to be pressed. Then the graph will be updated to show the defined sequence, and the values in the screen will be changed to allow the user to add the second block. The same procedure has to be repeated to add more blocks until the sequence is completely defined.

To edit an already defined block, the same procedure has to be followed but previously selecting the number of the block in *Block number* field. After changing the desired parameters, *Edit block* has to be pressed in order to change the stored values for the stated block, and adapt the graph to them.

To remove a block, the number of it has to be selected in *Block number*, and *Remove block* has to be pressed. This will erase the stored values for that block and move one position back the values of the already defined following blocks in order to adapt the sequence.

Finally, when one is satisfied with the created sequence, *Done* button in the bottom left corner of the screen has to be pressed in order to go back to the “Continuous Preparation” screen.

It is noticeable to know that the graph escalates itself every time according to the maximum voltage of the sequence and the total time. So that, the values in the high part of time and voltage scales are changed every time a block is added, modified or deleted.

5.2.6. “Continuous Running” screen

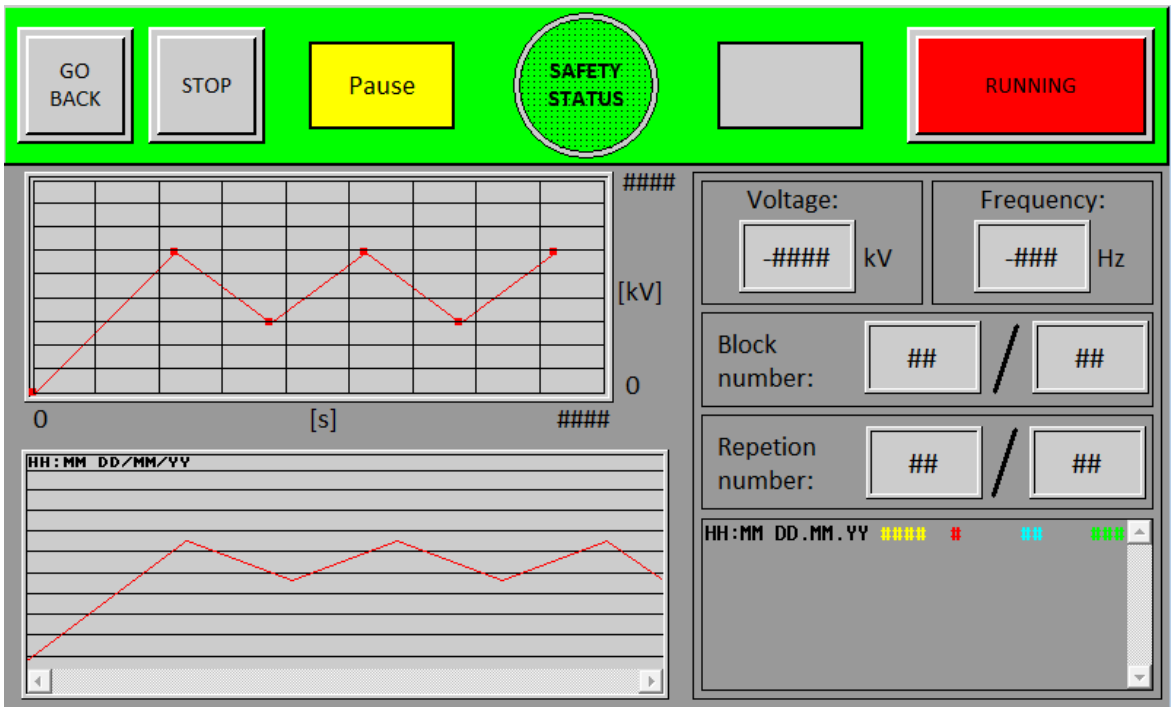


Fig. 40 TP screen to run a Continuous sequence

This screen appears after pressing *Ready* in the “Continuous Preparation” screen. It is used to start running the sequence and to show its status while running.

As soon as Safety Status has been turned on through the button in the Console, *Run* button in the top right of the screen will be clickable. If pressed, it will change to running and the sequence will start. As soon as voltage starts to be applied to the setup, Operation Status will turn on, turning the top green bar to red. After a block is ended, the system will stop Operation Status and wait until the voltage is 0 again to enter the *Pause* state (shown in a yellow banner left of *Safety Status* indicator), or *Grounding Pause* (yellow banner right of *Safety Status* indicator).

The main grey part of the screen is divided in two sides. The left part is occupied by two graphs. The top one shows the defined sequence, actually it is the same graph that can be seen in “Edit sequence” screen. The bottom one is a life graph showing the value of the measured voltage every one second, and can be moved to the sides through a lower bar in order to show all the data when it doesn’t fit in the screen. Both of them are escalated according to the maximum voltage expected in the sequence.

In the right part of the main grey screen, the current values of Voltage and Frequency can be seen on the top. Under them one can keep track on the current block number out of the total amount of blocks that define a sequence; or the repetition number out of the total number of repetitions.

In the bottom right part there is a table recording the Breakdowns and giving information on them. Every time there is a Breakdown or a repetition ends without interruption, a new entry in the table is registered. Firsts two columns from the left show the time and date of the entry, followed by four columns displaying the number of the repetition, a 1 or a 0 indicating if a breakdown has been detected or not (respectively), the block number the sequence was in case a Breakdown has been detected, and the voltage applied when there was the breakdown. If there was no breakdown in that repetition, each of the three columns in the right will show a 0.

Finally, while the sequence is running, *Stop* button in the top left of the panel will be available. Pressing this button will stop the sequence and bring the touch panel to the "Continuous finished" screen. The same will happen if the green *Operation OFF* button in the Console is pressed.

Regarding the *Go Back* button in the top left corner of the screen, which brings the panel back to "Continuous Preparation" screen, it is important to note that it is only available to be pressed while the sequence is not running. As soon as run is pressed, it gets inactive and only *Stop* can be pressed.

5.2.7. “Continuous Finished” screen

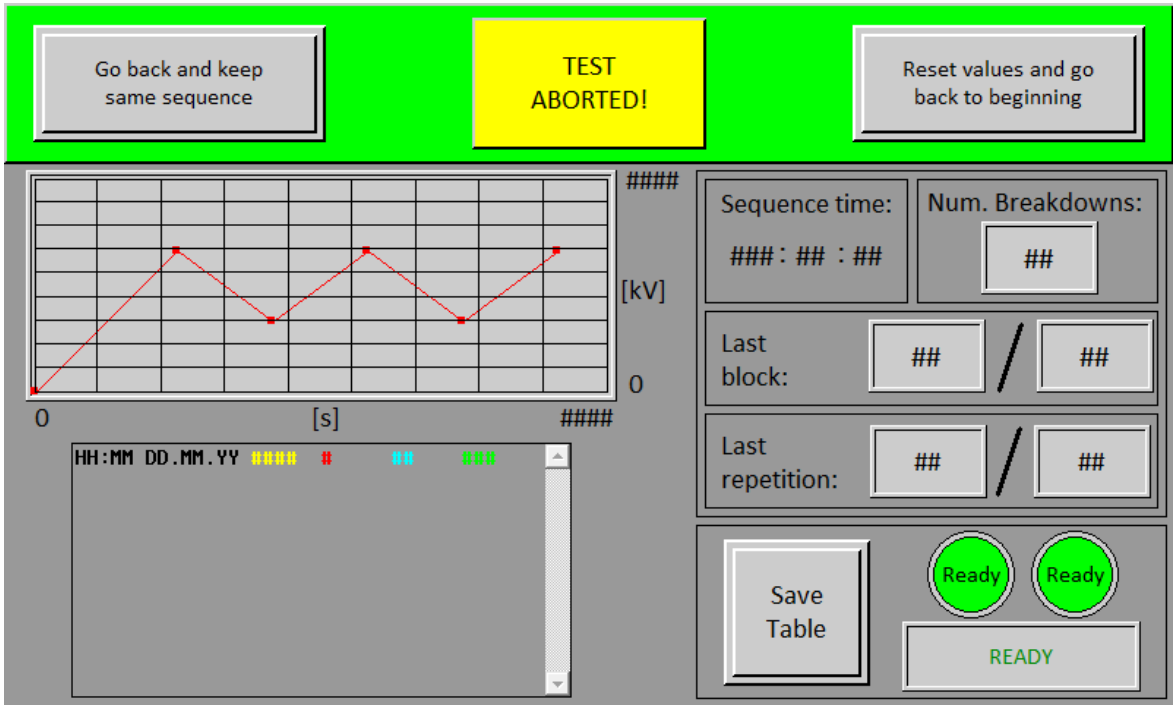


Fig. 41 TP screen appearing when a Continuous sequence has finished

“Continuous Finished” screen appears when an automatic continuous sequence ends, either because it reached the end or because of any other kind of interruption.

It basically tries to keep the structure and information appearing in “Continuous Running” screen, showing the intended shape of the sequence in a graph in the left, and the table registering the breakdowns right under it.

On the right side of the panel the last block and repetition out of the total that should have been done are shown. And right over them the total time of the sequence and the number of Breakdowns are also displayed. In the bottom right corner there is a button to save the table registering the breakdowns in a USB Pendrive attached to the panel. As soon as it is pressed, it changes a value in a register that urges the panel to copy the value of the Logging buffer where the data is stored to the USB port. If everything is right, when the button will be released the banners in the right will show a message announcing it.

In the top Status bar, this time there is a banner in the middle substituting *Safety Status*. This banner shows when the process has ended correctly (*Test finished!* message in green), or when it has been aborted manually or by any malfunction or safety feature (*Test aborted!* message in yellow).

After all the values have been reviewed, the user can choose either to go back to “Continuous Preparation” screen without erasing the already defined sequence parameters but resetting the ones changed during the execution of the experiment, in order to try to run the same again (button in top left corner); or to go back to the beginning in the *Manual/Auto* selection screen and delete the configured sequence while resetting all the variables.

5.2.8. "Impulse Preparation" screen

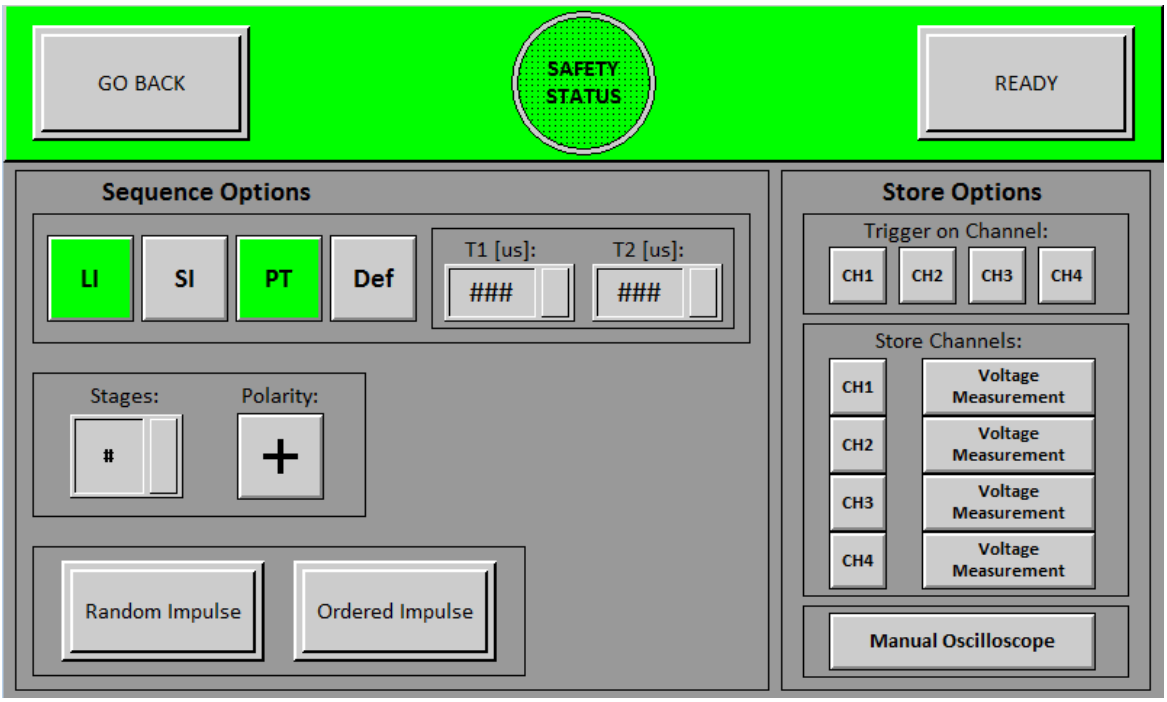


Fig. 42 TP screen to prepare the Auto Impulse mode

This screen appears after pressing *Impulse* in "Auto Continuous/Impulse" screen. Here one can set the parameters for an automated Impulse sequence.

It has a layout similar to the "Continuous Preparation" screen, with exactly the same buttons to configure the oscilloscope in the right part.

In the left, the kind of Impulse circuit can be chosen between *LI* (Lightning), *SI* (Switching), *PT* (PigTail) or *Def* (defined by the user through the T1 and T2 values of the double exponential voltage shape in microseconds at its right). The selection of these parameters will affect the times per division set in the oscilloscope. Right under this, number of stages and polarity of the circuit can be selected. And finally at the bottom, two buttons allow to choose between a Random sequence or an Ordered one. When any of the buttons is pressed, the screen will change to a new one where the specific options and limits for the sequence can be set.

When everything is set, pressing *Ready* in the top right corner will set the oscilloscope (if configured to be set automatically) and change the screen to the "Impulse Running" one.

5.2.9. "Random Impulse" screen

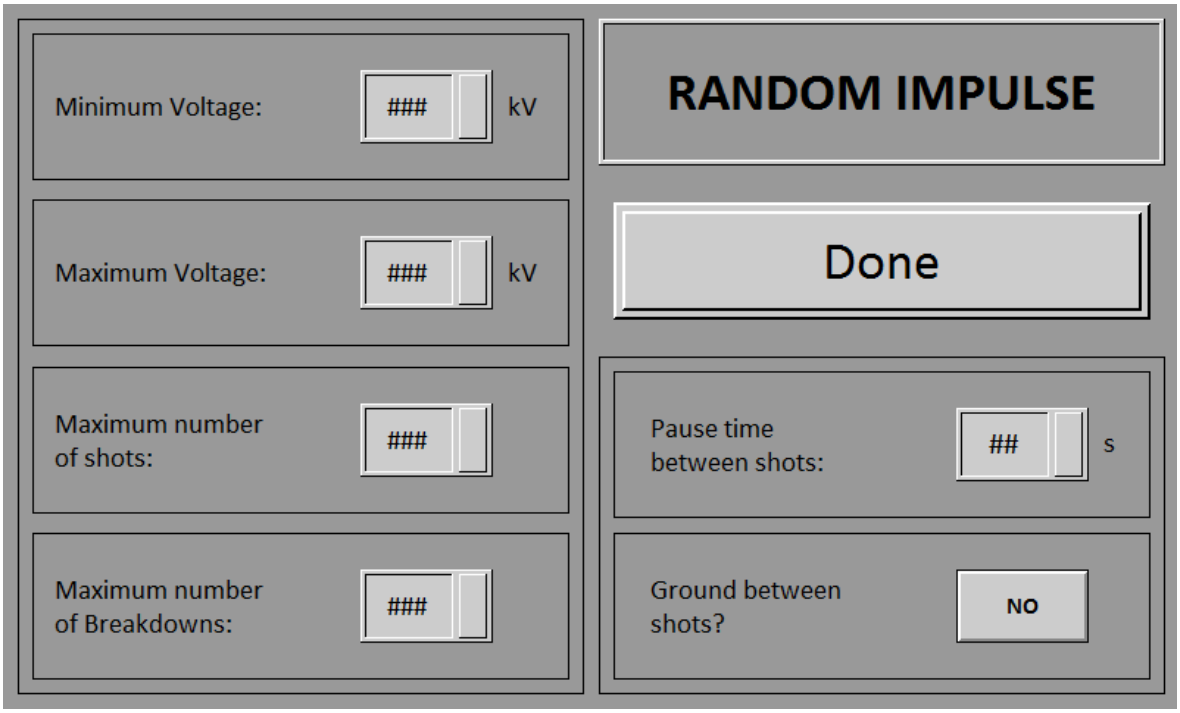


Fig. 43 TP screen to configure a random Impulse sequence

This screen is used to configure a randomized Impulse sequence.

In the left part of the screen it allows to choose the parameters of the sequence, like the minimum and maximum voltages that the random values can take; or the maximum number of shots and breakdowns, so the sequence stops as soon as one of them is reached.

In the right part of the screen, the time in seconds to rest between the shots and if ground or not between them can be chosen.

When all the parameters have been chosen, *Done* button brings the user back to "Impulse Preparation" screen.

5.2.10. “Ordered Impulse” screen

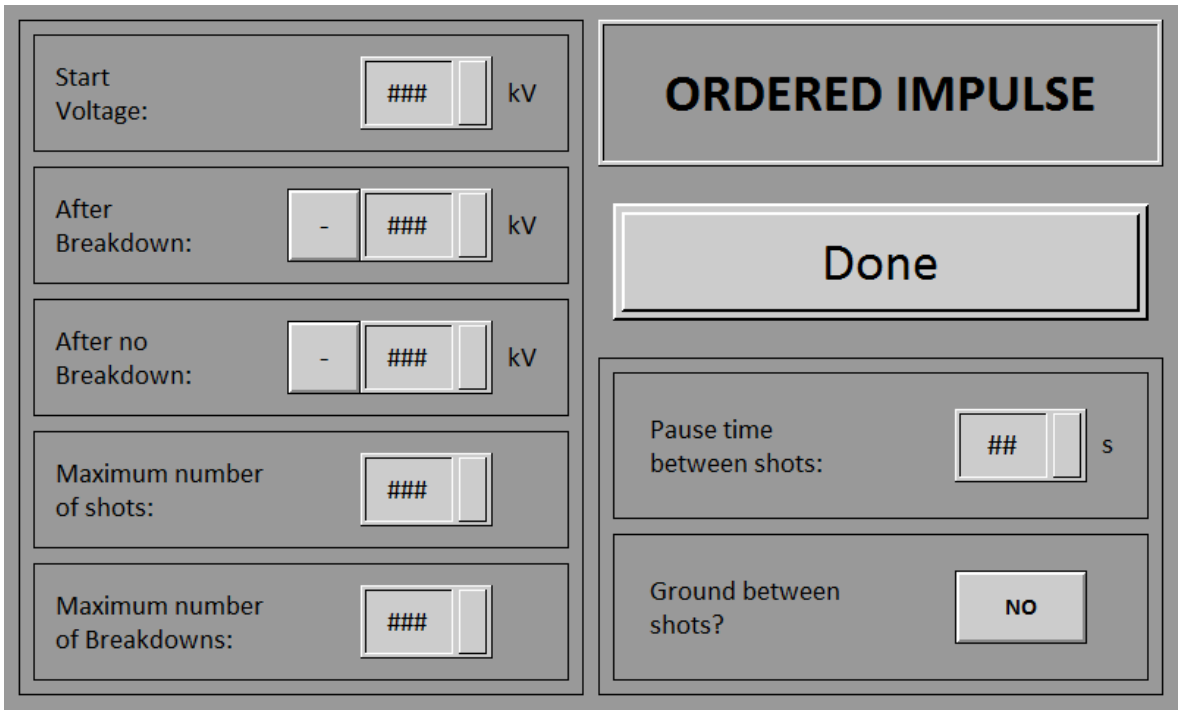


Fig. 44 TP screen to configure an ordered Impulse sequence

This screen allows the user to set the parameters for an ordered sequence of Impulse shots.

Similar to the “Random Impulse” screen, in the right side it allows to choose the pause time between shots and if to ground or not between them.

In the left part of the screen, the first voltage to be shot has to be chosen. The two parameters right below it set how will be the following shot in case there has been a breakdown or not. A specific voltage has to be chosen to be added (positive sign) or subtracted (negative sign) in each case. Finally, the user can also define the limit of shots and breakdowns to stop the sequence.

As soon as everything is configured as desired, pressing *Done* will bring the user back to “Impulse Preparation” screen.

5.2.11. “Impulse Running” screen

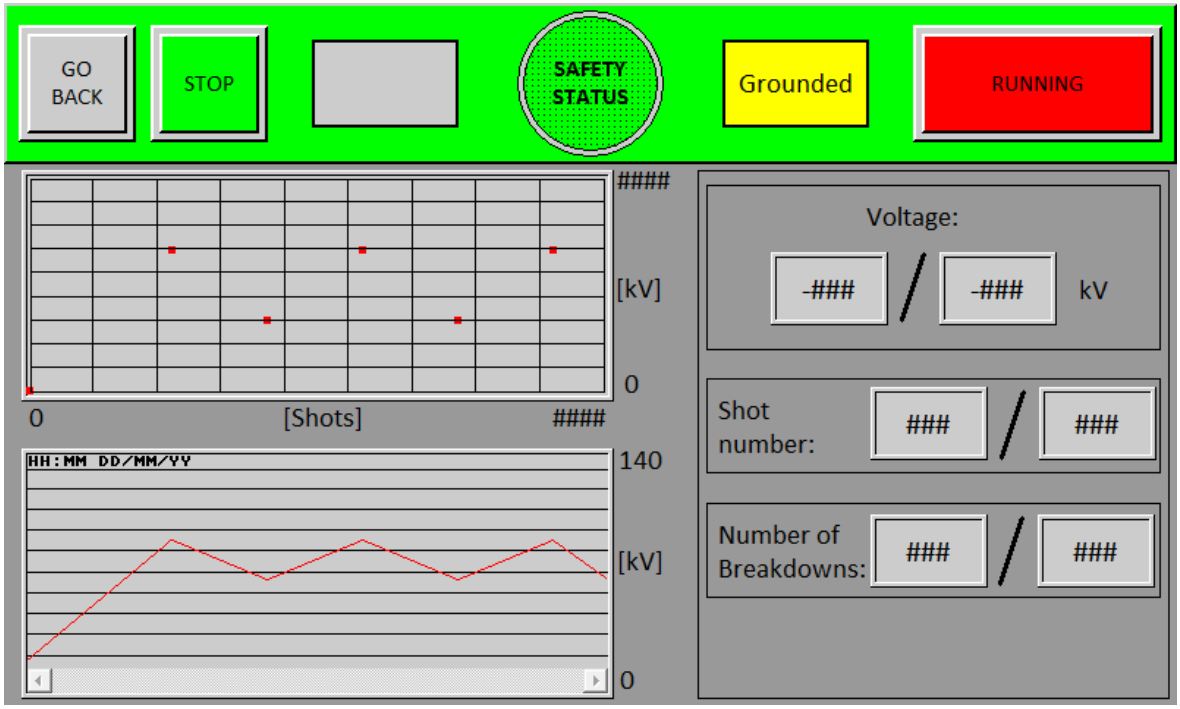


Fig. 45 TP screen to run an Impulse sequence

This screen is displayed after pressing *Ready* in “Impulse Preparation” screen, and allows to start running the sequence through the button in the top right corner, as soon as there is Safety Status (set through the corresponding button in the Console).

With a similar layout to “Continuous Running” screen, it will show status information in the top, with the yellow banners indicating when the sequence is paused or grounded between the shots, the *Run* button showing *Running* as soon as pressed and the sequence is working, and the green top bar becoming red when voltage is being applied to the setup.

The main area of the screen shows information about the sequence running. In the left, two graphs record the voltages being shot every time. The top one adapts every time to the maximum voltage and number of shots present in the sequence, and escalates the values and the high ranges of the scales in order to display the sequence in the nicest way. The lower one is just recording the voltage applied right before each shot, without the ability to adapt the size of the screen, but with the possibility to store the values of the sequence in a USB drive or a Compact Flash card, as they are registered in a Logging Buffer in the memory of the touch panel.

In order to make this last graph not very difficult to read when there are low voltages, three identical screens were created with different top voltage values for the scale in the graph (140, 280 and 420 kV), and the panel goes automatically to one of them according to the number of stages selected.

In the right part of the screen, the current measured voltage versus the nominal that is desired to be applied are shown; followed by the number of shots and breakdowns out of the maximums configured.

Finally, in the top left of the screen there are the buttons to go back when the sequence has not started yet, or to stop when it's already running.

5.2.12. “Impulse Finished” screen

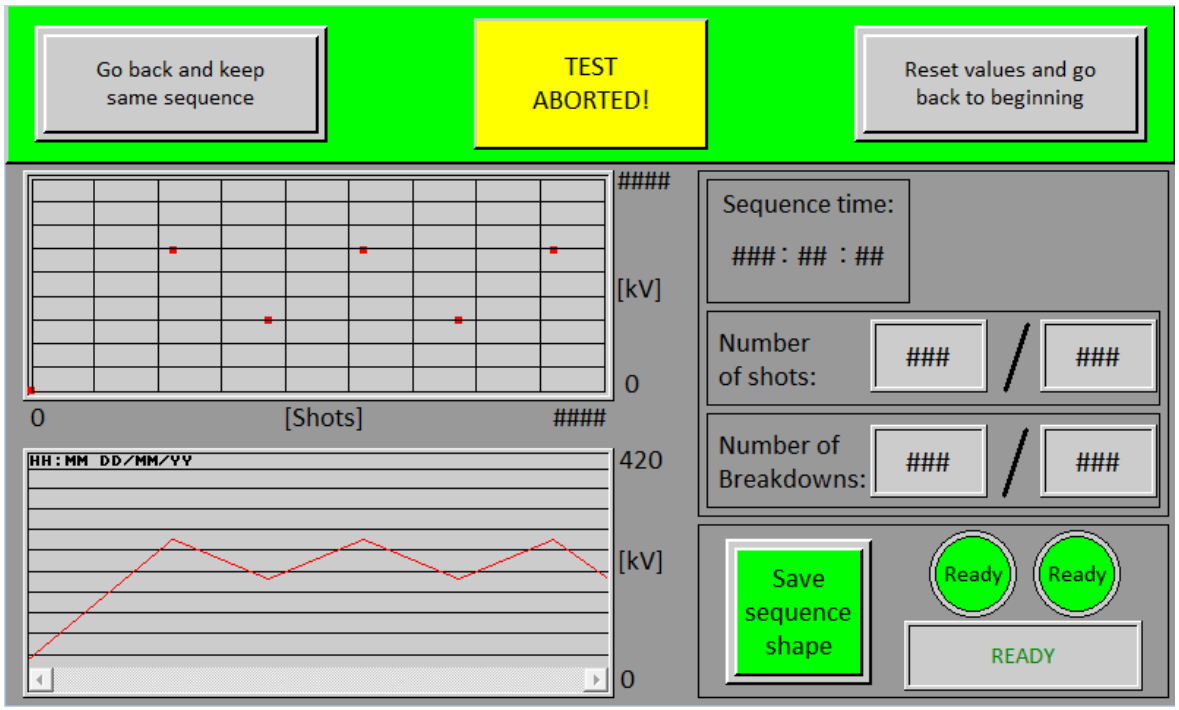


Fig. 46 TP screen appearing when an Impulse sequence has finished

“Impulse Finished” screen appears when an Impulse sequence ends, either by reaching any of the stated limits or aborted manually or by any safety feature.

It keeps the structure of “Impulse Running” screen, with the two graphs in the left and the number of shots and breakdowns in the right. There is also the total time spent by the sequence right above the number of shots. And, in the bottom right, a button to save the sequence shape stored in a logging buffer in the touch panel and shown in the lower graph in the screen. As the save button in “Continuous Finished” screen, it has some banners right of it showing if the data has been correctly stored.

The top bar is similar to the one of “Continuous Finished” screen, with a banner in the middle indicating if the sequence has finished correctly or has been aborted before. And there are also the buttons to go back and keep the values to repeat the sequence in the left; and to delete and reset all the parameters and start again from the beginning in the right.

As the “Impulse Running” screen, this one has also three almost identical screens for the different number of stages, adapting the scale and maximum voltage displayable in the lower graph.

6. Code structure and detailed explanation

6.1. Basics of AC500 programming

The application running the PLC was programmed using *ABB*'s software *PS501 Control Builder Plus* in its version 2.2.0, which is based on *CoDeSys version 2.3.9.34 (Build Mar 5 2012)*, a development environment for PLC programming by the company *3S Systems*.

The CPU in the Lab is an old model with the firmware version 1.2 (new ones have 2.x firmware versions). This means all the coding and parameters configuration is directly done in *CoDeSys*. In the "Resources" tab, the right CPU has to be selected under "Target Settings", and the "PLC Configuration" must be defined, adding all the modules present in the device. A task to run the main program is also defined there, with the possibility to adjust the cycle time of the PLC in order to make it faster or slower. In CPUs with firmware newer than 2.x all this configuration is done through a tool directly in *PS501* software, out of the *CoDeSys* environment.

CoDeSys is a complete development environment for PLCs. Each project is organized in POUs (Program Organization Units). The first one to be created will be named PLC_PRG, and it will be run cyclically by the PLC, so all other ones should be called from there. The body of each POU can be written in any of the following IEC programming languages:

- **Instruction List (IL)**: Textual low level programming language which resembles assembly. It consists of a series of instructions, each one beginning in a new line and containing an operator and one or more operands.

- **Structured Text (ST)**: Textual high level programming language that resembles Pascal or C. It also consists of a series of instructions, but it allows complex statements like *DO-WHILE* or *IF-THEN-ELSE*.

- **Sequential Function Chart (SFC)**: Graphical programming language based on GRAFCET. It is structured in steps with associated actions, linked between them by transitions with associated logic conditions.

- **Function Block Diagram (FBD)**: Graphical programming language based in networks. Each network contains a structure that represents a logical or arithmetic expression between input variables (in the left) and output variables (in the right), the call of a function block, etc.

- **Ladder Diagram (LD)**: Graphical programming language based on representing the structure of an electric circuit. It consists on a series of networks with a circuit made of contacts, coils and connecting lines, that transmit from left to right the condition "ON" "OFF" equivalent to the Boolean values "TRUE" "FALSE". Its name comes from its resemblance with a ladder, with two vertical rails on the sides and horizontal rungs between them.

- **Continuous Function Chart (CFC)**: Graphical language very similar to Function Block Diagram. However, in SFC elements are positioned freely in the space instead of using networks as in FBD. This allows the use of feedback, for example.

CoDeSys supports all the IEC standard POUs, included in the library Standard.lib. Moreover, other libraries can be included and more POUs can be written. When creating a new POU, this can be of three different types:

- **Program (PRG)**: This kind of POU is recognized globally throughout the whole project, and keeps all the values from the last time it was run until the next one, regardless it's called from different places.

- **Function Block (FB)**: This type of POU also retains the values from the last run until the next time is called, but only if it is called from the same instance. That means the Function Block allows copies of it to be made and called from different parts of the code. This is very useful to avoid having to create different POUs with the same structure many times.

- **Function (FUN)**: This kind of POU always returns a variable, being this one of a previously defined type. It doesn't keep the values from one run to the following, so it can be called from different places in the code with different results depending on the input variables.

Regarding the variables, these can be either local (can only be accessed from inside the POU where they were declared) or global (can be accessed from anywhere in the program). In order to interact with the inputs and outputs in the PLC and with the touch panel, this variables have to be set in a physical address. This addressing is done in the declaration of the variables, adding the line *"AT %..."*, followed by some letters that identify if the variable is an Input/Output of the PLC or a Memory address to be read or written from the touch panel; the type of variable (Boolean, Word, etc.); and finally some numbers addressing it to a unique physical address. Apart from the standard data types, other specific ones can also be created in the "Data Types" tab of CoDeSys.

In “Visualizations” tab, a basic interface can be created, in order to see more clearly what the variables are doing when trying the program. To help in debugging, CoDeSys has also different features, like offline and online simulation modes, and the possibility to add breakpoints where the process will stop, allowing to see the values of the variables in that exact moment.

These are the main things to know about the programming environment for the PLC running the lab. For deeper information, one should refer to the *User Manual for PLC Programming with CoDeSys 2.3* **[1]** by *3S-Smart Software Solutions GmbH*, available from the internet in PDF.

6.2. Basics for programming the Touch Panel

The *ABB CP435T-ETH* touch panel present in the lab was programmed using ABB's *CP400 Soft version 1.2.0 (Build 186)*.

The first thing to be done is to define the basic parameters under "Workstation Setup". Most important ones are the model of the touch panel and PLC, and the type of communication between them. With this, the touch panel will be able to interact with the controller by changing the variables stored at *%M* addresses in the PLC. Addresses of the Control and Status blocks are also defined there, and the Logging buffers have to be activated there too.

The **Control Block** is a set of registers in the Controller. It allows this last one to control actions in the operator terminal by changing some variables. Some of the typical operations include changing the screen, triggering the collection of data for some graphs, or commanding the logging buffers. The **Status Block**, on the other hand, shows information about the status of the operator terminal, like the current displayed screen or the state of the logging buffers.

Logging Buffers are used to store, in the touch panel RAM, the different values a variable took during a certain amount of time. The data acquisition can be triggered either by a timer (so data is acquired every certain time), or by the Controller (changing the state of a bit in the Control Block). This data can then be displayed in a *Historical Display* object (which includes *Historical Trend Graph*, *Historical Data Table* or *Historical Event Table*), and stored to a CF Card or USB Memory Stick.

The main programming of the interface in the touch panel is quite simple. It consists in creating different screens that can be filled with different kinds of objects. Each object can interact with one or more variables (either internal or from the PLC), doing something different depending on the kind of object: from reading a bit, to writing some characters or displaying a bunch of values in a graph. The user can also define the shape, size and general appearance of every object.

To make the assignation of the variables to the objects easy and not having to point to its physical address every time, a list of global variables can be exported from the PLC and imported to the Operator Terminal under *Tag Table* window. This will allow the programmer of the touch panel to refer to the memory addresses in the PLC directly by the name of the variable assigned there, making the process more intuitive and user-friendly.

Apart from modifying variable through objects in the screens, the touch panel has also the feature to run some *Macros*. These are little pieces of code used to complement the actions of some buttons and reduce the program size and optimize controller efficiency. There are different types of Macros depending on when they are executed:

- ***Application Macros***: they are recognized through the whole application. It includes the *Initial Macro* (executed only once when the application is started), *Background Macro* (executed cyclically while the application is running) and *Clock Macro* (executed every 500 ms while the application is running).

- ***Screen Macros***: These macros are linked to the screen they are defined to. It includes the *Open Macro* (executed when the screen is commanded to be opened), *Close Macro* (executed when the screen is commanded to be closed), and *Cyclic Macro* (executed cyclically while the screen is displayed).

- ***ON/OFF Macros***: They are defined in push-button objects. If active, the *ON Macro* is executed once when the button is pressed and the bit assigned to it is set to ON. The *OFF Macro*, on the other hand, is executed when the button is released and the bit set to OFF.

- ***Sub-Macros***: It is a sub-command defined from the object menu, which can be called by the operator terminal with the CALL command.

These are the most relevant things to know about the programming environment for the touch panel used in the lab. For further information and deeper instructions on how to program each object and functionality, people should refer to the manuals by *ABB* available online in PDF format. Mainly *CP400Soft Technical Manual* ***[2]***, and *Control panel CP400 series First steps* ***[3]***.

6.3. Downloading the programs to the PLC and Touch Panel

Once the codes for the touch panel and PLC have been written, they have to be downloaded to the devices. To do so, they are connected with the laptop in the lab through Ethernet cables and an Ethernet Switch. As it is set at the moment, they have the following IP addresses:

- Laptop: 192.168.0.11.
- PLC: 192.168.0.10.
- Touch Panel: 192.168.0.9.

6.3.1. Downloading code to the PLC

If the IP addresses are set correctly, so the PLC and the laptop can see each other in the same network, downloading the code is very easy. In the top menu in CoDeSys, one must go under “Online” and select “Login”. The software will automatically compile the code and, if no errors are found, it will tell the user that changes have been made to the program the PLC had stored, and ask if it has to update just the changes or “Load all” the code again. Experience shows it is recommendable to press in “Load All”. After the download is finished, to make sure the project stays in the PLC even after rebooting it the user should press “Create Boot Project”, also under the “Online” tab in top menu.

In case the laptop can’t login to the PLC, is most likely the Ethernet Switch or any of the cables are disconnected. If the physical connection is right, it might be a problem with the IP addresses. To change and set a new IP address to the PLC, the following manual (available in PDF from ABB) should be consulted: *AC500 Ethernet Configuration* ***[4]*****.**

6.3.2. Downloading code to the Touch Panel

When the IP addresses are well configured, the user must go to top menu “Application” → “Compile”. After the project is compiled without errors, one has to go again to “Application” and select “Download Application”. If the program is running, a dialog will appear in the touch panel asking to confirm the download of a new application. After pressing on “Yes”, the new code will be flashed in the ROM memory of the panel.

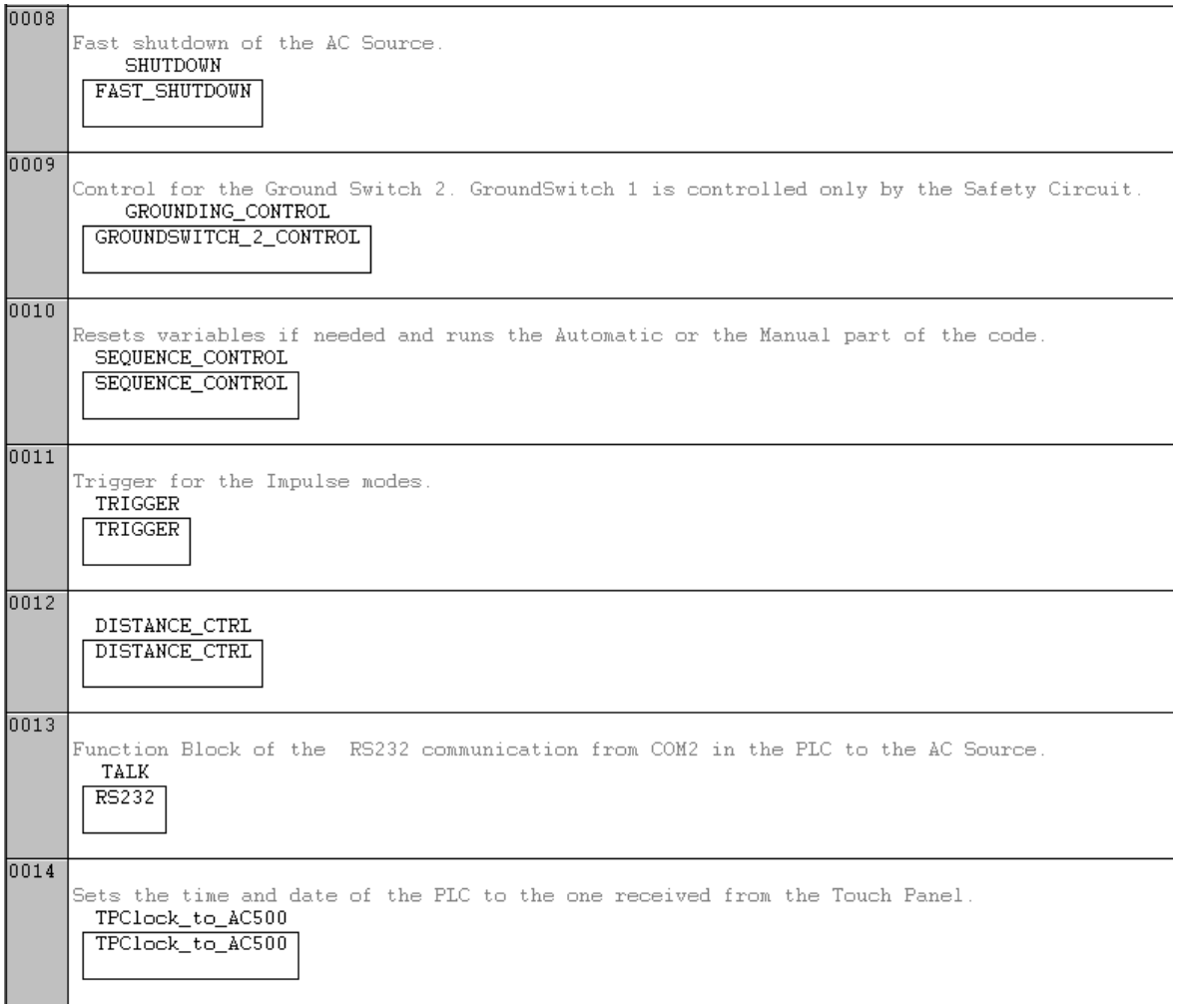
In case the communication doesn’t work because something is not well configured, one just has to go under “Options” → “Communication parameters”, where the communication method can be selected (Ethernet, USB, COM1, COM2…), together with choosing the IP address the touch panel has or the required parameters for each mode.

6.4. Detailed explanation of the code

In the following pages, each POU in the code will be represented and explained in detail.

6.4.1. PLC_PRG (PRG)

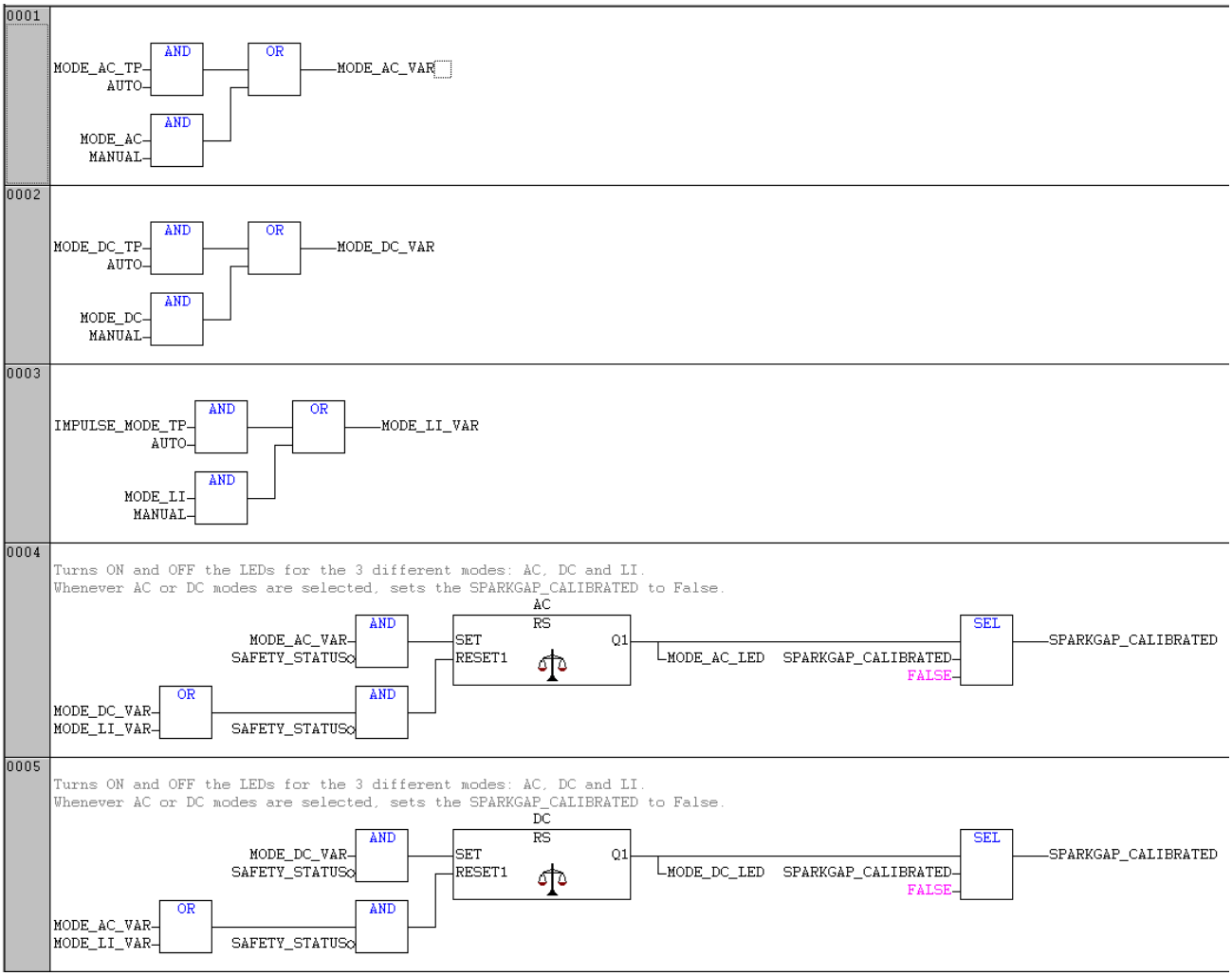
The first one is the main program, called **PLC_PRG**. This program is executed cyclically by the PLC, and all the rest of the code is called from there.



As it can be seen, it first sets some variables related to the Safety Status, and then it calls other function blocks in the application. FBs called in this first layer of code have the highest priority, and they will always be executed in every cycle, regardless being in Manual or Auto modes.

The order between some of them is not random. For example, it is important that when a breakdown is detected in BREAKDOWN_DETECTION function block, the PLC disconnects the voltage supply in CONTROLLER and FAST_SHUTDOWN function blocks before entering the SEQUENCE_CONTROL and continuing with the automatic functions.

6.4.2. VOLTAGE_FORM (FB)

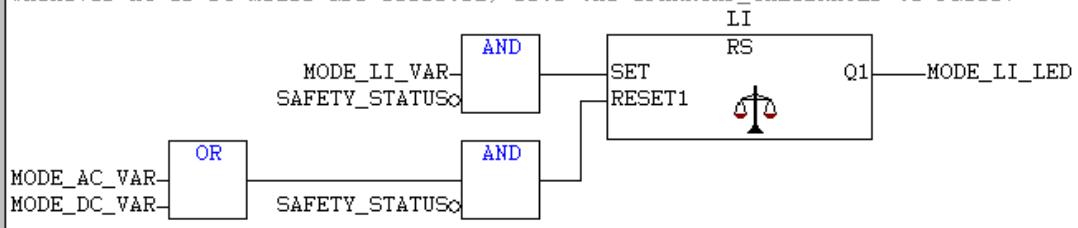


This function block choses the voltage mode between AC, DC or Impulse. To do so, it reads the corresponding variables depending if Manual or Auto mode is selected, stores a string containing ‘AC’, ‘DC’ or ‘LI’ in VOLTAGEFORM variable, and switches ON the LED of the chosen mode in the Console.

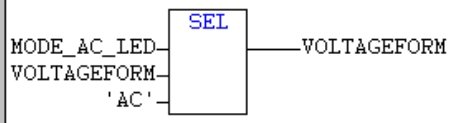
It also resets the variable SPARKGAP_CALIBRATED as soon as AC or DC mode are selected, so to calibrate the SparkGap again the following time Impulse mode is selected. Finally, when a mode is chosen, it sets the variable MODE_SET to true.

0006

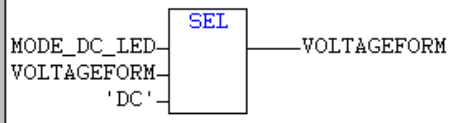
Turns ON and OFF the LEDs for the 3 different modes: AC, DC and LI.
Whenever AC or DC modes are selected, sets the SPARKGAP_CALIBRATED to False.



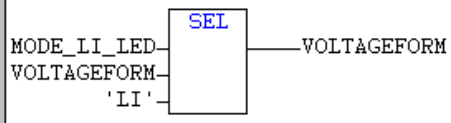
0007



0008

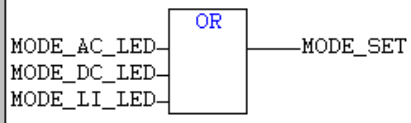


0009

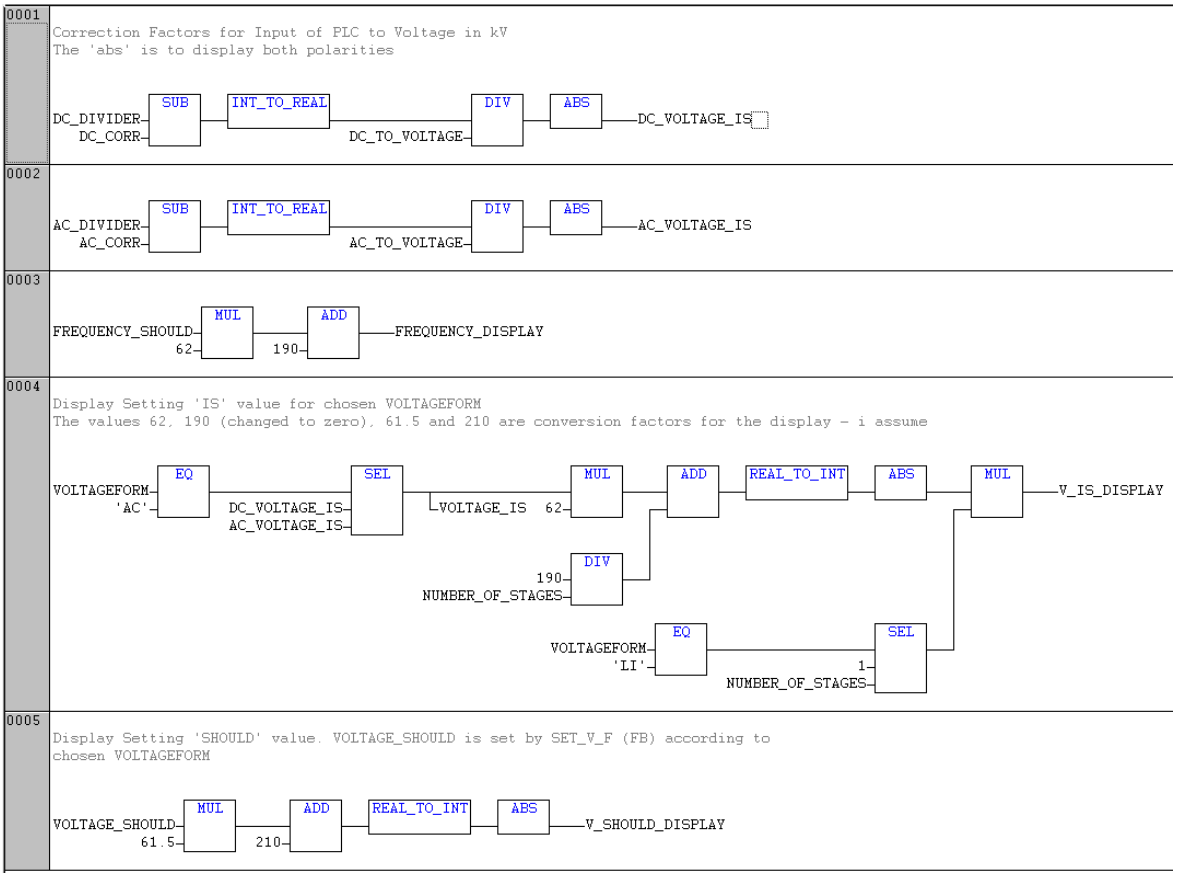


0010

The bool "Mode_Set" is true when one of the LED is on
This bool is used in FB Operation to enable turning on



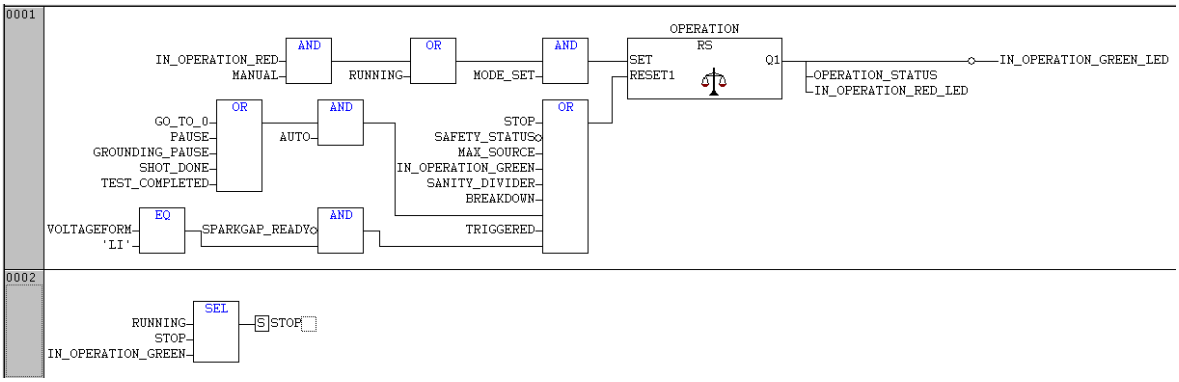
6.4.3. DISPLAY (FB)



This function block is in charge of controlling the Voltages and Frequency displays in the Console. To do so, it first takes the data from the AC and DC dividers in the HV Setup, converts them to voltages and makes the absolute values of them in order to take both polarities (networks 1 and 2). Later, in network 4, choses the voltage to display according to the mode set and the number of stages, and converts it to a value that the LCD display can read and represent as the stated the voltage.

In networks 3 and 5 it takes the values for nominal voltage and frequency and converts them too so the display can represent them.

6.4.4. OPERATION (FB)



OPERATION is one of the most important function blocks in the program, as it controls the resetting of the variable OPERATION_STATUS, and so if there is voltage being applied to the circuit or not. It also controls the operation red and green LEDs, that show to the user the operation status in an easy way.

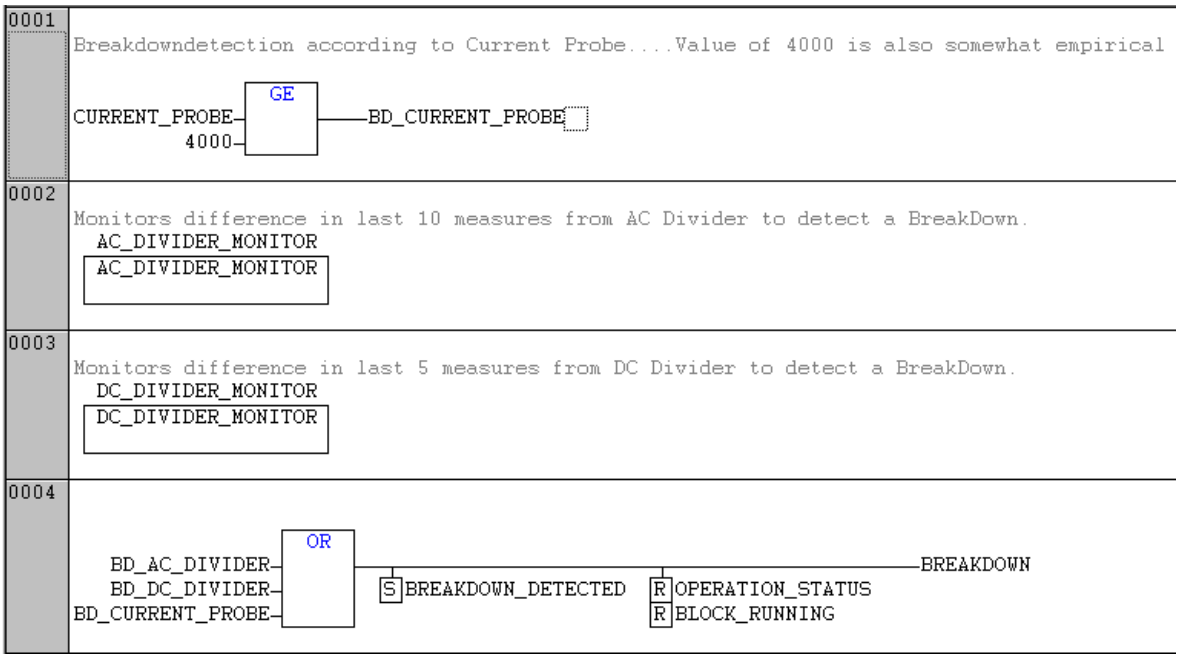
It consists basically in an RS (Resetting Bistable) block, a basic POU that sets its Q1 output as true when its RESET1 input is false and either the other input SET or the output Q1 are true. So, after being set once, keeps the state until it is reset.

In the application in the lab, it is activated by the operation ON button in the Console when in Manual mode or by RUNNING variable for Auto mode, always assuming a voltage mode has been set.

In the reset field, all the cases where the power supply has to be stopped are considered. From the basic safety ones as the Safety Circuit not being closed, the Sanity Divider, Breakdown detected, AC Source set to the maximum or Impulse mode selected and Spark Gap not ready; to the manual ones like STOP variable or the operation OFF button in the console; passing by the different stages without voltage in the Auto mode (waiting for the circuit to discharge to 0 voltage, pause, grounded pause, etc.).

In network number 2, it sets true the variable STOP when pressing the operation OFF button in the console and an auto sequence is running. That allows to stop the automatic sequence directly from this button in the console, instead of having to use the Touch Panel.

6.4.5. BREAKDOWN_DETECTION (FB)



This function block is in charge of detecting breakdowns in the circuit and setting the corresponding variables to stop the power source and respond accordingly in the automatic mode. Breakdowns can be detected in different sources:

- A **Current Probe** that measures the current that goes to the ground when there is a breakdown. This method has to be further tested until it works correctly, as the pulse generated by the probe seems to be too fast for the PLC to detect it.

- Two function blocks that monitor the **AC** and **DC Dividers**, checking continuously their last measures in order to detect big changes on them that indicate a breakdown is happening.

6.4.6. AC_DIVIDER_MONITOR (FB)

```
--------

V1 := V2;
V2 := V3;
V3 := V4;
V4 := V5;
V5 := V6;
V6 := V7;
V7 := V8;
V8 := V9;
V9 := V10;
V10 := AC_VOLTAGE_IS;

DERIVAT_AC := V10 - V1;

IF ABS(DERIVAT_AC) > 2 THEN
        BD_AC_DIVIDER := TRUE;
ELSE
        BD_AC_DIVIDER := FALSE;
END_IF;


IF DERIVAT_AC > max_value_ac THEN
        max_value_ac := DERIVAT_AC;
END_IF;

IF DERIVAT_AC < min_value_ac THEN
        min_value_ac := DERIVAT_AC;
END_IF;

--------
```

As commented before, the AC Divider Monitor stores the last 10 values from the AC voltage divider and compares the last one with the 10^{th} last one. When the difference in absolute value is bigger than 2, considers a Breakdown has been detected.

The threshold of 2 was evaluated empirically so that the function can withstand a normal regulation of voltage without interpreting there is a breakdown.

The last lines of the code store the maximum and minimum value for the evaluated difference. This was used to set the explained threshold of 2, and can be removed for future use.

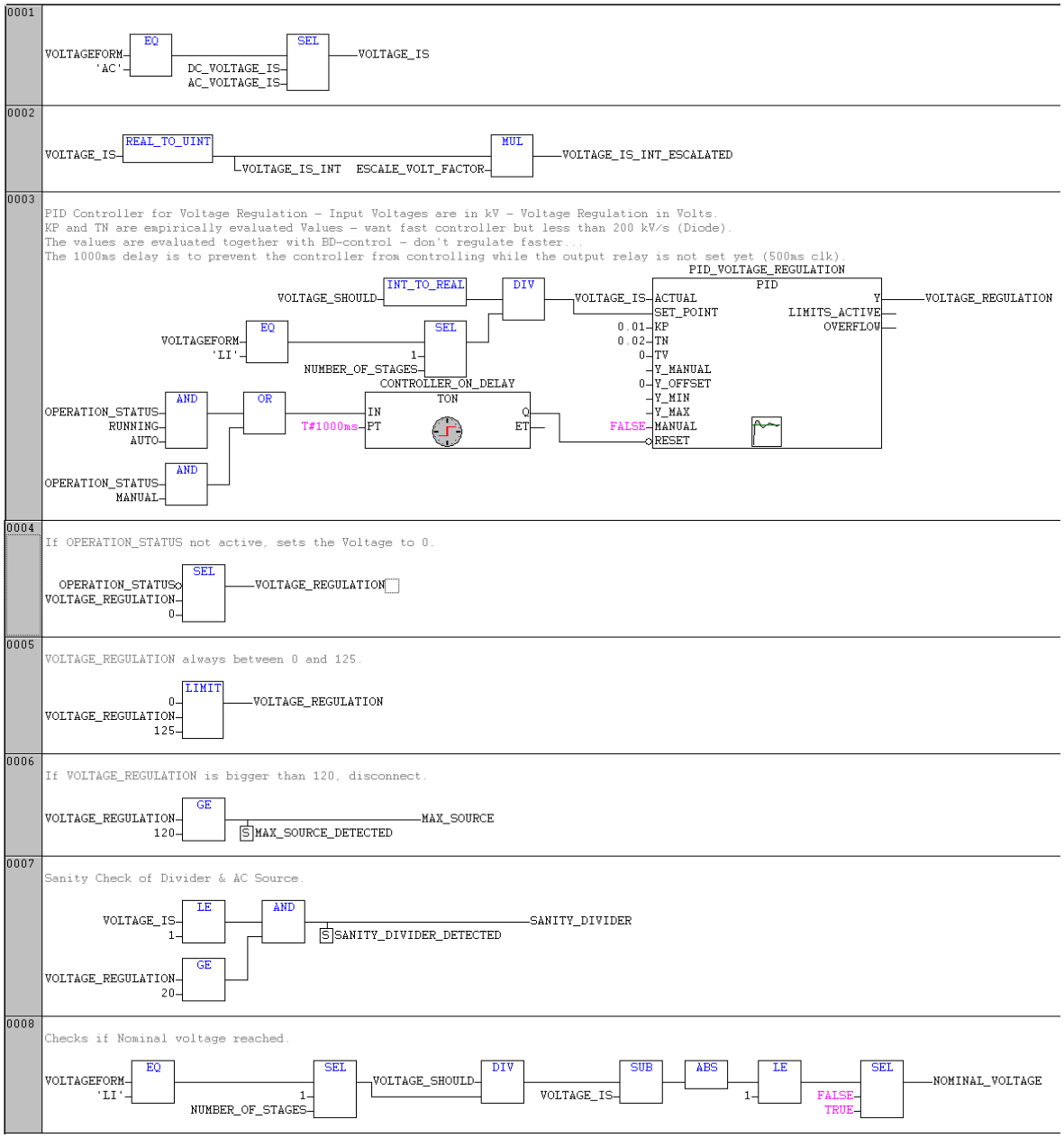
6.4.7. DC_DIVIDER_MONITOR (FB)

```
--------

V1 := V2;
V2 := V3;
V3 := V4;
V4 := V5;
V5 := DC_VOLTAGE_IS;

DERIVAT_DC := V5 - V1;

IF ABS(DERIVAT_DC) > 5 THEN
        BD_DC_DIVIDER := TRUE;
ELSE
        BD_DC_DIVIDER := FALSE;
END_IF;


IF DERIVAT_DC > max_value THEN
        max_value := DERIVAT_DC;
END_IF;

IF DERIVAT_DC < min_value THEN
        min_value := DERIVAT_DC;
END_IF;

--------
```

The DC Divider Monitor works the same way as the AC one, with the only differences of taking the last 5 values instead of 10, and having a threshold of 5 instead of 2 (also empirically evaluated).

Taking 5 values instead of 10 makes the detection faster. The voltage from the AC divider, however, is post processed by a circuit board to get the *rms* value, delaying this the arrival of the measures and bringing noise to them. This noise proved to be bigger than the differences between the last 5 values, that's why more values had to be considered to correctly detect the breakdowns when checking the AC divider.

6.4.8. CONTROLLER (FB)



This function block is mainly in charge of regulating the AC Source through a PID to reach the nominal voltage in a controlled way.

To do so, it first reads the current voltage from the divider according to the mode selected. Then, the PID compares it with the nominal voltage we want to reach, and selects a voltage regulation according to the parameters set (*Kp*, *Tn*…). These values are selected in order not to regulate faster than 200 kV/s, and so never go over the rated 20 mA maximum current through the diodes in the Lab. The reset field of the PID block is controlled by OPERATION_STATUS, only allowing the PID to regulate the voltage when it's true.

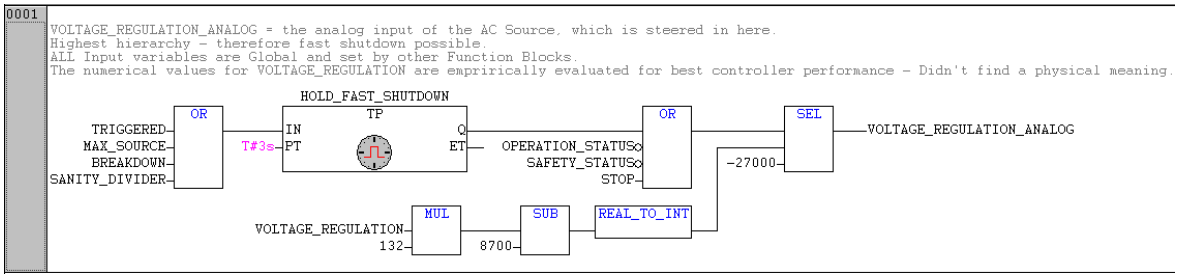
Moreover, there is a delay of 1 second in order to avoid the PID to start controlling while the output relay is still not set (it is set by a 500 ms clock). Network number 5 sets limits to the voltage regulation; in the other hand, when there is no operation status network number 4 sets the voltage regulation back to 0.

Apart from the PID controller, this function block has some more features. In network number 6, the safety feature MAX_SOURCE is implemented, setting a variable to stop operating if the voltage regulation goes to the maximum of the AC Source. The 7th network implement the SANITY_DIVIDER, a safety feature that stops Operation if the controller is trying to give more than 20 kV while the actual measured value is still under 1 kV (this might indicate some malfunctioning in the circuit or AC Source).

Finally, the last network compares the voltage set to be reached with the actual measured value, to check if nominal voltage has been reached.

It’s important to note that, in Impulse mode, the voltage is only measured in the first stage, regardless the setup has more or not. This makes necessary to adapt, in some parts of the code, the nominal voltage or the measured one according to the number of stages.

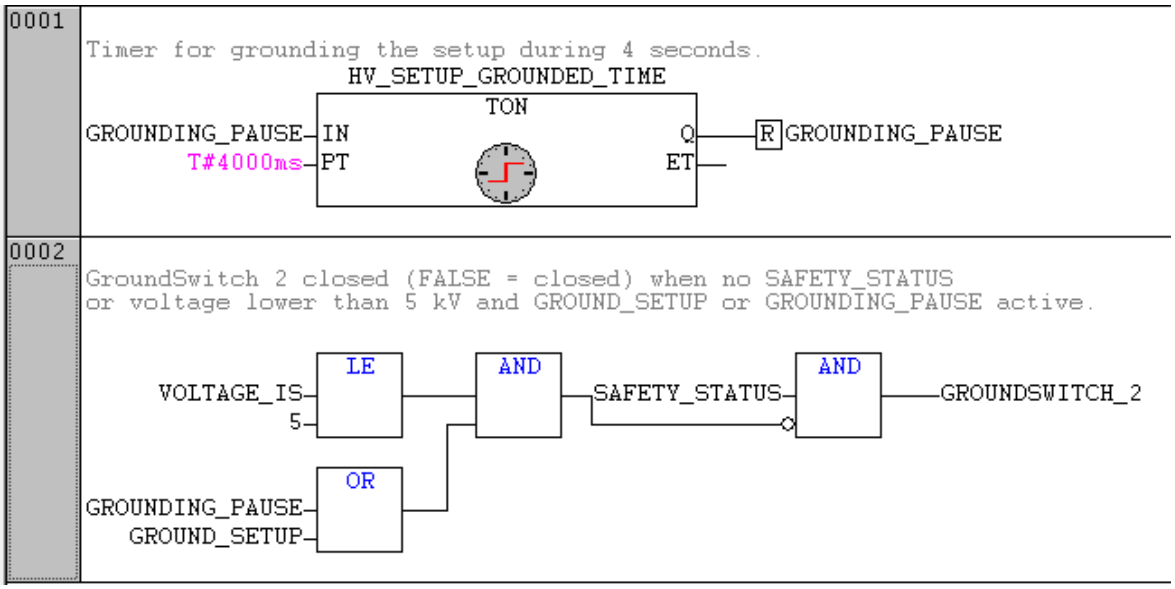
6.4.9. FAST_SHUTDOWN (FB)



This function block acts directly to the analog input of the AC Source, allowing a fast shutdown when necessary.

When any of the safety features that require the power to be disconnected is active, it sets the analog input of the AC Source directly to -27000, the lowest value. In the other hand, while any of these variables is active and there is Operation Status, it converts the value in VOLTAGE_REGULATION (calculated in CONTROLLER function block) to the analog value necessary for the AC Source to set the desired voltage. The values used for the conversion were evaluated empirically to give the best performance.

6.4.10. GROUNDSWITCH_2_CONTROL (FB)



This block is in charge of controlling one of the two Ground Switches (the other one is directly controlled by the Safety Circuit).

It is used in the automatic mode to ground the setup between two shots or repetitions, or when the sequence is ended. GROUNDSWITCH_2 is an output variable that acts directly to the Ground Switch, closing it (setup grounded) when false.

The block will ground the circuit as soon as there is no safety status, or when it's commanded to be grounded and the voltage is smaller than 5 kV. If the grounding command comes from a Grounding Pause, it will also set a timer to count 4 seconds, so to open again the circuit after the time has passed.

6.4.11. SEQUENCE_CONTROL (FB)

```
--------

IF NOT VARIABLES_INITIALIZED THEN
        INITIALIZE_VARIABLES;
        VARIABLES_INITIALIZED := TRUE;
END_IF;

IF NOT SEQUENCE_RESET THEN
        PARTIALLY_RESET_VARS;
        SEQUENCE_RESET := TRUE;
END_IF;

IF AUTO THEN
        NOT_RUNNING := NOT RUNNING;

        IF MODE_AC_TP THEN
                MAX_VOLT_BLOCK := NUMBER_OF_STAGES * 100;
        ELSIF MODE_DC_TP OR IMPULSE_MODE_TP THEN
                MAX_VOLT_BLOCK := NUMBER_OF_STAGES * 140;
        END_IF;
        VOLT_LIMIT_1 := MAX_VOLT_BLOCK;
        VOLT_LIMIT_2 := MAX_VOLT_BLOCK;
        VOLT_LIMIT_3 := MAX_VOLT_BLOCK;
        VOLT_LIMIT_4 := MAX_VOLT_BLOCK;
        VOLT_LIMIT_5 := MAX_VOLT_BLOCK;

        PAUSE_BETWEEN_SHOTS;

        IF MODE_AC_TP OR MODE_DC_TP THEN
                BLOCK_TIMER;

                IF NOT RUNNING THEN
                        PREPARE_AUTO_CONTINUOUS;
                ELSIF RUNNING THEN
                        RUN_AUTO_CONTINUOUS;
                END_IF;

        ELSIF IMPULSE_MODE_TP THEN
                IF NOT RUNNING AND READY THEN
                        PREPARE_IMPULSE;
                ELSIF RUNNING THEN
                        RUN_IMPULSE;
                END_IF;
        END_IF;

        GRAPHS_UPDATE;

        TOTAL_TIME_COUNTER;

ELSE
        SET_V_F;
        NUMBER_OF_STAGES_SELECT;
END_IF

--------
```

This is the function block controlling the sequence mode. It runs a part of the code or another depending if the Auto or Manual mode is selected.

First of all, if needed, reinitializes all the variables or only the ones related to a run sequence. Then, if Auto mode is selected, it sets the maximum possible voltage according to the voltage mode and the number of stages. This value is copied to different variables that will limit the possible inputs when setting parameters from the touch panel.

The function block PAUSE_BETWEEN_SHOTS is called just before splitting between a part of the code for the Continuous modes (AC and DC), and the Impulse. If a Continuous mode is selected, it calls BLOCK_TIMER and splits again between calling PREPARE_AUTO_CONTINUOUS (in case RUNNING is false) or RUN_AUTO_CONTINUOUS (if the command to “Run” has been received from the touch panel).

If Impulse mode was selected, it also splits between not yet running (then it calls PREPARE_IMPULSE) or sequence started (RUN_IMPULSE is called). Still in the Auto mode, it calls GRAPHS_UPDATE to refresh the data in the graphs and tables in the touch panel; and TOTAL_TIME_COUNTER to count the duration of the sequence.

Finally, when Auto mode is not selected the program assumes it is in Manual and calls SET_VF and NUMBER_OF_STAGES_SELECT, in order to activate the buttons in the Console.

6.4.12. PARTIALLY_RESET_VARS (FB)

```
--------

NOMINAL_VOLTAGE := FALSE;
TRIGGERED := FALSE;
BREAKDOWN := FALSE;

SPARKGAP_READY := FALSE;
SPARKGAP_CALIBRATED := FALSE;

VARIABLES_PREPARED := FALSE;
TIME_DIV_CONFIGURED := FALSE;
OSCI_CHANNELS_CONFIGURED := FALSE;
SEQUENCE_MODE_CONFIGURED := FALSE;
OSCI_READY := FALSE;
OSCI_DATA_STORED := FALSE;
OSCI_ARMED := FALSE;

GO_TO_0 := FALSE;

BLOCK_MSECONDS := T#0s;

OSCI_CHANNEL := 1;

FREQUENCY_SHOULD := 50;
VOLTAGE_SHOULD := 0;

VOLTAGE_REGULATION := 0;

OPERATION_STATUS := FALSE;

RUNNING := FALSE;
STOP := FALSE;
PAUSE := FALSE;
READY := FALSE;

BREAKDOWN_DETECTED := FALSE;
GROUNDING_PAUSE := FALSE;
TEST_COMPLETED := FALSE;

SHOT_DONE := FALSE;

BLOCK_RUNNING := FALSE;
START_BLOCK := FALSE;

READ_BLOCK := FALSE;
EDIT_BLOCK := FALSE;
REMOVE_BLOCK := FALSE;

GROUND_SETUP := FALSE;

VOLTAGE_SELECTED := FALSE;

NUM_SHOTS := 0;
NUM_BD := 0;

CURRENT_BLOCK := 1;
CURRENT_REPETITION := 1;

TOTAL_TIME_SECONDS := 0;
```

```
TOTAL_TIME_MINUTES := 0;
TOTAL_TIME_HOURS := 0;

I := 0;
WHILE I < 100 DO
        BLOCK_ARRAY_ESCALATED[I].ACUM_TIME := 0;
        BLOCK_ARRAY_ESCALATED[I].VOLT := 0;
        SHOTS_ARRAY[I].ACUM_TIME := 0;
        SHOTS_ARRAY[I].VOLT := 0;
        I := I + 1;
END_WHILE;

RESET_LOGGING_BUFFERS := TRUE;

--------
```

As it can be seen, this function block basically sets the variables used while running a sequence to their original values. However, it doesn't change the parameters set to define the sequence. Thanks to this the sequence can be executed again from the beginning.

6.4.13.

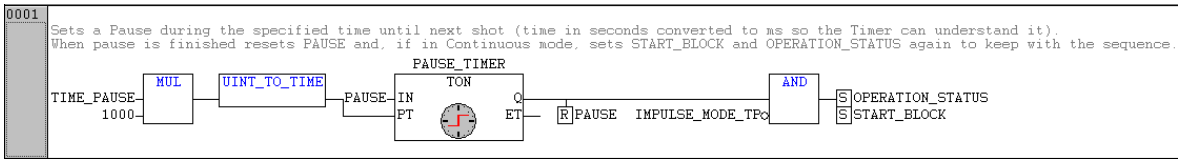
INITIALIZE_VARIABLES (FB)

```
--------

PARTIALLY_RESET_VARS;

MODE_SET := FALSE;

MODE_DC_TP := FALSE;
MODE_AC_TP := FALSE;
IMPULSE_MODE_TP := FALSE;

AUTO := FALSE;
MANUAL := TRUE;

SIGN_AFTER_BD := FALSE;
SIGN_AFTER_NO_BD := TRUE;

LI_MODE := TRUE;
SI_MODE := FALSE;
PT_MODE := FALSE;
DEF_MODE := FALSE;
POLARITY := TRUE;
RANDOM_IMPULSE := TRUE;
ORDERED_IMPULSE := FALSE;

CONTINUE_AFTER_BD := FALSE;
PREPARE_OSCI := FALSE;

TIME_DIVISION := 0;
VOLT_DIVISION := 0;
VOLT_DIVISION_AC := 0;
VOLT_DIVISION_CP := 0;
OFFSET := 0;
OFFSET_AC := 0;

BLOCK_NUM := 0;
TOTAL_BLOCKS_PLUS1 := 1;

MAX_VOLT_SEQUENCE := 0;


START_VOLT := 0;
MIN_VOLT := 0;
MAX_VOLT := 420;
AFTER_BD := 10;
AFTER_NO_BD := 10;
MAX_SHOTS := 100;
MAX_BD := 100;
T1 := 0;
T2 := 0;


BLOCK_FREQUENCY := 50;
START_VOLT_BLOCK := 0;
END_VOLT_BLOCK := 0;
MIN_VOLT_BLOCK := 0;
MAX_VOLT_BLOCK := 140;
MAX_VOLT_SEQUENCE := 0;

VOLT_LIMIT_1 := 140;
VOLT_LIMIT_2 := 140;
```

```
VOLT_LIMIT_3 := 140;
VOLT_LIMIT_4 := 140;
VOLT_LIMIT_5 := 140;

TOTAL_TIME := 0;
DURATION_BLOCK := 0;
REPETITIONS := 1;


NUMBER_OF_STAGES := 1;

TRIGGER_ON_CHANNEL := 1;
TIME_PAUSE := 5;

TOP_VOLT_AXIS := 0;
TOP_TIME_AXIS := 0;


I := 0;
WHILE I < 100 DO
        BLOCK_TIME_ARRAY[I] := 0;
        BLOCK_FREQUENCY_ARRAY[I] := 50;
        BLOCK_ARRAY[I].ACUM_TIME := 0;
        BLOCK_ARRAY[I].VOLT := 0;
        I := I + 1;
END_WHILE;


I := 1;
WHILE I <= 4 DO
OSCI_CH_ARRAY[I] := 1;
OSCI_CH_CP_ARRAY[I] := 1;
I := I + 1;
END_WHILE;

--------
```

This function block sets all the variables for the Automatic mode back to their original values, so a new sequence can be defined again. To do so, it first calls the function block PARTIALLY_RESET_VARS, and then resets all the remaining variables one by one.

6.4.14. PAUSE_BETWEEN_SHOTS (FB)



This block is in charge of controlling the pauses between shots or repetitions in the automatic modes. As soon as PAUSE is enabled, it starts a timer that reads the defined time duration for the pause, and resets PAUSE as soon as it is finished.

When the sequence run is in the continuous mode, it also sets the variables OPERATION_STATUS and START_BLOCK again in order to start the following block of the sequence.

6.4.15. ESCALATE_GRAPH (FB)

```
--------

IF MAX_VOLT_SEQUENCE > 0 AND TOTAL_TIME > 0 AND NOT IMPULSE_MODE_TP THEN
        ESCALE_VOLT_FACTOR := 450 / MAX_VOLT_SEQUENCE;
        ESCALE_TIME_FACTOR := 500 / TOTAL_TIME;

        TOP_VOLT_AXIS := 450 / ESCALE_VOLT_FACTOR;
        TOP_TIME_AXIS := 500 / ESCALE_TIME_FACTOR;

        I := 1;
        WHILE I < TOTAL_BLOCKS_PLUS1 DO
                BLOCK_ARRAY_ESCALATED[I].VOLT := BLOCK_ARRAY[I].VOLT *
                ESCALE_VOLT_FACTOR;
                BLOCK_ARRAY_ESCALATED[I].ACUM_TIME := BLOCK_ARRAY[I].ACUM_TIME *
                ESCALE_TIME_FACTOR;
                I := I + 1;
        END_WHILE;

ELSIF MAX_VOLT_SEQUENCE > 0 AND NUM_SHOTS > 0 AND IMPULSE_MODE_TP THEN
        ESCALE_VOLT_FACTOR := 450 / MAX_VOLT_SEQUENCE;
        ESCALE_TIME_FACTOR := 500 / NUM_SHOTS;

        TOP_VOLT_AXIS := 450 / ESCALE_VOLT_FACTOR;
        TOP_TIME_AXIS := 500 / ESCALE_TIME_FACTOR;

        I := 1;
        WHILE I <= NUM_SHOTS DO
                BLOCK_ARRAY_ESCALATED[I].VOLT := SHOTS_ARRAY[I].VOLT *
                ESCALE_VOLT_FACTOR;
                BLOCK_ARRAY_ESCALATED[I].ACUM_TIME := SHOTS_ARRAY[I].ACUM_TIME *
                ESCALE_TIME_FACTOR;
                I := I + 1;
        END_WHILE;

END_IF

UPDATE_GRAPH := TRUE;

--------
```

This block is used in the automatic modes to adapt the voltage and time values of each block or shot in order to fill the graphs in the touch panel, as they have a fixed range. To do so, it looks for the highest voltage (for the Y axis) and time or number of shots (for the X axis) in the sequence, and multiplies all the values by an escalation factor that brings the highest value close to the edge of the graph.

Looking deeper into the code of the block, it can be seen that there are two parts: one for the Continuous mode and the other for the Impulse mode. They both work the same way, being the only difference the value used for the X axis: the accumulated time for Continuous mode, and the number of shots for the Impulse mode.

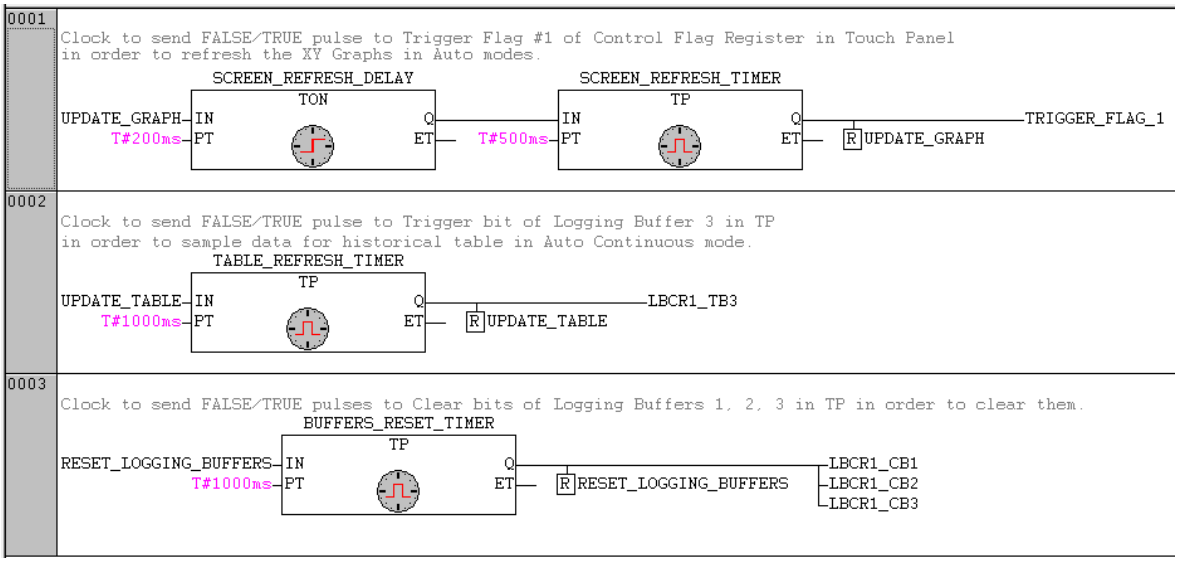
The code will only run if the maximum voltage and the total time or number of shots are bigger than zero, because if they are zero would mean that any block is defined. In case there are blocks to escalate, it first calculates the escalation factors dividing the maximum values defined for each axis in the graphs in the touch panel by the maximum voltage and time or number of shots.

Then, it divides back the fixed maximum ranges in the graphs by the calculated escalation factors. These two operations should theoretically bring back to the maximum values of the sequence, but as the variables are of type *INT*, the results of the divisions will be always truncated, so the final values after these two divisions will be either the same or smaller than in the beginning.

Finally, it multiplies all the used values in the arrays of voltage and accumulated time by the escalation factors, so the touch panel can read them and represent them nicely in the graphs.

After having escalated all the necessary values, it sets the variable UPDATE_GRAPH to true, so the graphs will be refreshed with the new data.

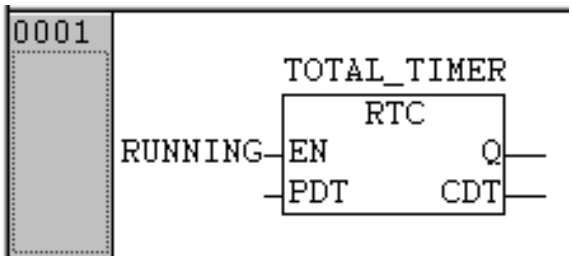
6.4.16. GRAPHS_UPDATE (FB)



This block is in charge of setting the necessary bits to refresh and clear data in graphs and logging buffers in the touch panel. Its basic implementation is with timers that, when activated, enable the desired bit to true for half or one second, enough time for the touch panel to detect the FALSE/TRUE edge needed to do the task.

In the first network, UPDATE_GRAPH activates the bit to refresh the data in the *XY Graphs*, the ones were the escalated values with the shape of the sequence are represented. In the second one, a new set of values in Logging Buffer 3 are collected when UPDATE_TABLE is true (this logging buffer stores, and then shows in a table, if there has been a breakdown and where has it been after each repetition in the Continuous mode). Finally, the third network resets the three used logging buffers when the corresponding variable activated to do so.

6.4.17. TOTAL_TIME_COUNTER (FB)



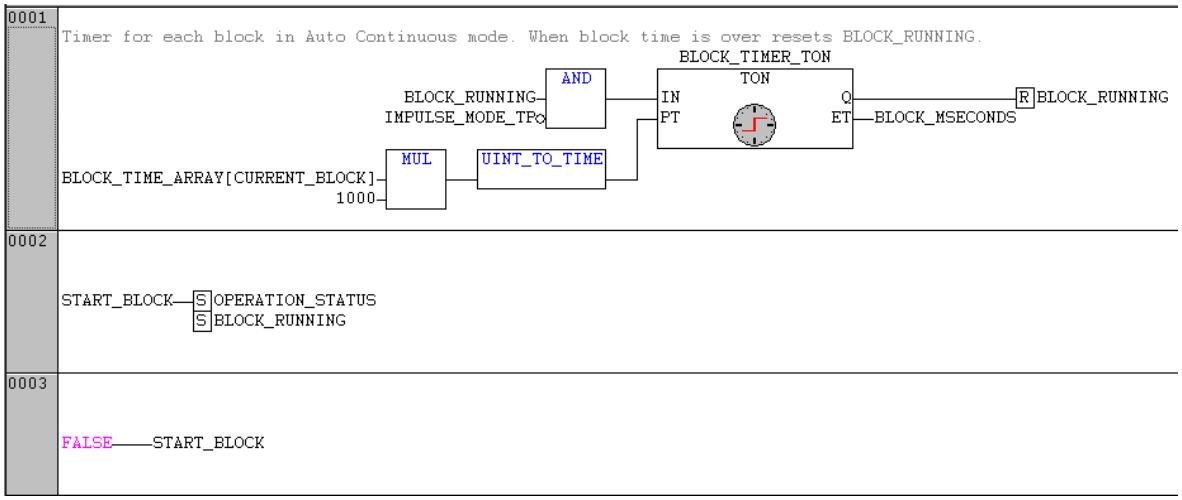
In this block, a Real Time Clock is called to be enables while the sequence is running, so the total time of the sequence can be measured.

6.4.18. GET_TOTAL_TIME (FB)

```
CURRENT_TIME := TIME_TO_DWORD(TOTAL_TIMER.CDT - TOTAL_TIMER.PDT)/1000;

TOTAL_TIME_SECONDS := DWORD_TO_UINT(CURRENT_TIME MOD 60);
TOTAL_TIME_MINUTES := DWORD_TO_UINT((CURRENT_TIME / 60) MOD 60);
TOTAL_TIME_HOURS := DWORD_TO_UINT((CURRENT_TIME / 60) / 60);
```

This piece of code gets the total time spent by the sequence subtracting the initial time from the current time in the RTC in TOTAL_TIME_COUNTER. Later, it converts the value to hours, minutes and seconds to represent it in an understandable way.

6.4.19. BLOCK_TIMER (FB)



BLOCK_TIMER function block controls the duration of each block in Auto Continuous mode. Whenever BLOCK_RUNNING is true and any of the Continuous modes are selected, it starts counting the specified time for the current block. When time is over, it resets the variable BLOCK_RUNNING to send the sequence to another state. The variable BLOCK_MSECONDS takes the current time value counted in milliseconds, used later to calculate the desired voltage at that exact moment of the sequence.

Networks 2 and 3 are used to start the timer. When the variable START_BLOCK is true, it sets OPERATION_STATUS and BLOCK_RUNNING, that will activate the voltage supply and the explained timer to count the block duration. Immediately after START_BLOCK is set false again not to interfere with the normal functioning of the timer and the sequence control.

6.4.20. PREPARE_AUTO_CONTINUOUS (FB)

```
--------

IF READ_BLOCK THEN
        START_VOLT_BLOCK := BLOCK_ARRAY[BLOCK_NUM - 1].VOLT;
        END_VOLT_BLOCK := BLOCK_ARRAY[BLOCK_NUM].VOLT;
        BLOCK_FREQUENCY := BLOCK_FREQUENCY_ARRAY[BLOCK_NUM];
        DURATION_BLOCK := BLOCK_TIME_ARRAY[BLOCK_NUM];
        TOTAL_TIME := BLOCK_ARRAY[TOTAL_BLOCKS_PLUS1 - 1].ACUM_TIME;
        FIND_MAX_VOLT;
        ESCALATE_GRAPH;
        READ_BLOCK := FALSE;
END_IF;

IF EDIT_BLOCK THEN
        BLOCK_ARRAY[BLOCK_NUM].VOLT := END_VOLT_BLOCK;
        BLOCK_FREQUENCY_ARRAY[BLOCK_NUM] := BLOCK_FREQUENCY;
        BLOCK_TIME_ARRAY[BLOCK_NUM] := DURATION_BLOCK;
        BLOCK_ARRAY[BLOCK_NUM].ACUM_TIME := BLOCK_ARRAY[BLOCK_NUM - 1].ACUM_TIME +
        BLOCK_TIME_ARRAY[BLOCK_NUM];

        I := BLOCK_NUM + 1;
        WHILE I < TOTAL_BLOCKS_PLUS1 DO
                BLOCK_ARRAY[I].ACUM_TIME := BLOCK_ARRAY[I - 1].ACUM_TIME +
                BLOCK_TIME_ARRAY[I];
                I := I + 1;
        END_WHILE;

        BLOCK_NUM := BLOCK_NUM + 1;

        IF BLOCK_NUM > TOTAL_BLOCKS_PLUS1 THEN
                TOTAL_BLOCKS_PLUS1 := TOTAL_BLOCKS_PLUS1 + 1;
        ELSIF BLOCK_NUM <= TOTAL_BLOCKS_PLUS1 THEN
                BLOCK_NUM := TOTAL_BLOCKS_PLUS1;
        END_IF;

        READ_BLOCK := TRUE;
        EDIT_BLOCK := FALSE;
END_IF;

IF REMOVE_BLOCK AND 1 <= BLOCK_NUM AND BLOCK_NUM < TOTAL_BLOCKS_PLUS1 THEN
        I := BLOCK_NUM;
        WHILE I < TOTAL_BLOCKS_PLUS1 DO
                BLOCK_ARRAY[I].VOLT := BLOCK_ARRAY[I + 1].VOLT;
                BLOCK_ARRAY_ESCALATED[I].VOLT := BLOCK_ARRAY_ESCALATED[I + 1].VOLT;
                BLOCK_FREQUENCY_ARRAY[I] := BLOCK_FREQUENCY_ARRAY[I + 1];
                BLOCK_TIME_ARRAY[I] := BLOCK_TIME_ARRAY[I + 1];
                BLOCK_ARRAY[I].ACUM_TIME := BLOCK_ARRAY[I - 1].ACUM_TIME +
                BLOCK_TIME_ARRAY[I];
                BLOCK_ARRAY_ESCALATED[I].ACUM_TIME := BLOCK_ARRAY_ESCALATED[I -
                1].ACUM_TIME + BLOCK_TIME_ARRAY[I]*ESCALE_VOLT_FACTOR;

                I := I + 1;
        END_WHILE;

        TOTAL_BLOCKS_PLUS1 := TOTAL_BLOCKS_PLUS1 - 1;
        BLOCK_NUM := TOTAL_BLOCKS_PLUS1;
        BLOCK_ARRAY[BLOCK_NUM].ACUM_TIME := 0;
        BLOCK_ARRAY_ESCALATED[BLOCK_NUM].ACUM_TIME := 0;
```

```
        READ_BLOCK := TRUE;
        REMOVE_BLOCK := FALSE;
ELSIF REMOVE_BLOCK AND (BLOCK_NUM < 1 OR BLOCK_NUM >= TOTAL_BLOCKS_PLUS1) THEN
        REMOVE_BLOCK := FALSE;
END_IF;

IF READY THEN
        IF PREPARE_OSCI AND NOT OSCI_READY THEN
                PREPARE_OSCI_CONFIG;
        ELSE
                SNR := 6;
                CURRENT_BLOCK := 1;
                CURRENT_REPETITION := 1;
                ESCALATE_GRAPH;
                READY := FALSE;
        END_IF;
END_IF;

IF TEST_COMPLETED AND NOT OSCI_DATA_STORED THEN
        STORE_OSCI;
END_IF;

--------
```

This part of the application is executed while in AC or DC automatic modes but the sequence is not running. It is basically built by different pieces of code grouped under *IF* conditions that are executed depending on the buttons pressed in the touch panel, mainly to set the parameters that will define the sequence.

The first part of the code is executed when the variable READ_BLOCK is true. This code updates, according to the current block selected, the different parameters to show in the screen where blocks are configured. It also calls the action to find the maximum voltage of the sequence and the function block to find the escalated values to show in the graphs.

Second part, executed when EDIT_BLOCK is set true (also button “Edit Block” in touch panel), is in charge of storing the selected values for the current block in the arrays that will later be read to execute the sequence, regardless it’s a new block being created or just editing an already existing one. In order to be able to edit a block that is already inside the sequence, there is a *WHILE LOOP* that updates the array with accumulated times every time a block which is not new is modified. The current block number is also set to be increased only if the edited block was the last one (so if a new block was created). Finally, it sets READ_BLOCK true so to update the values in the screen to the ones of the new block number.

The third *IF* in the function block is executed when pressing to remove a block. In this case, and if the current selected block is bigger than zero and smaller or equal than the total number of created blocks, it will erase the values stored in the arrays, and move the values in following blocks one position back. It will also decrease the total number of blocks and set READ_BLOCK to true to update the values shown in the screen.

The fourth part of the function block is executed when the sequence has already been defined and the user presses “Ready” in the touch panel. Then, if the oscilloscope has to be set automatically (PREPARE_OSCI variable true) it will call PREPARE_OSCI_CONFIG until the oscilloscope has been configured (OSCI_READY set to true). Otherwise, it will just send the touch panel to the following screen and escalate and update the graph again.

Finally, there is a little part of code that calls STORE_OSCI function block in order to store the data from the last repetition when the test is completed and this data has not been stored yet.

6.4.20.1. FIND_MAX_VOLT (Action)

```
--------

N := 1;
MAX_VOLT_SEQUENCE := 0;
WHILE N < TOTAL_BLOCKS_PLUS1 DO
        IF BLOCK_ARRAY[N].VOLT > MAX_VOLT_SEQUENCE THEN
                MAX_VOLT_SEQUENCE := BLOCK_ARRAY[N].VOLT;
        END_IF
        N := N + 1;
END_WHILE;

--------
```

This little action is in charge of checking the array with the voltages of each block stored in it and find the highest one, needed for the escalation of the graphs to be shown in the touch panel.

6.4.21. RUN_AUTO_CONTINUOUS (FB)

```
--------

IF CURRENT_REPETITION <= REPETITIONS AND NOT STOP AND OPERATION_STATUS AND NOT
BREAKDOWN_DETECTED AND NOT GROUNDING_PAUSE AND NOT GO_TO_0 AND NOT PAUSE THEN
        IF BLOCK_RUNNING THEN
                FREQUENCY_SHOULD := BLOCK_FREQUENCY_ARRAY[CURRENT_BLOCK];
                CALCULATE_VOLTAGE_SHOULD;
        ELSIF NOT BLOCK_RUNNING AND NOT START_BLOCK THEN
                PREPARE_NEXT_BLOCK;
        END_IF;

ELSIF TEST_COMPLETED OR STOP OR SANITY_DIVIDER OR MAX_SOURCE OR NOT SAFETY_STATUS
THEN
        RUNNING := FALSE;
        STOP := FALSE;
        VOLTAGE_SHOULD := 0;
        OPERATION_STATUS := FALSE;
        BREAKDOWN_DETECTED := FALSE;
        GROUND_SETUP := TRUE;

        SNR := 7;

ELSIF GO_TO_0 THEN
        IF NOMINAL_VOLTAGE THEN
                IF CURRENT_REPETITION = REPETITIONS THEN
                        TEST_COMPLETED := TRUE;
                        GO_TO_0 := FALSE;
                ELSE
                        PAUSE := TRUE;
                        GO_TO_0 := FALSE;
                        CURRENT_BLOCK := 1;
                        CURRENT_REPETITION := CURRENT_REPETITION + 1;
                        IF GROUND_BETWEEN_SEQUENCES THEN
                                GROUNDING_PAUSE := TRUE;
                        END_IF;
                END_IF;

                OSCI_DATA_STORED := FALSE;
                OSCI_ARMED := FALSE;

                BREAKDOWN_DETECTED := FALSE;
        END_IF;

ELSIF BREAKDOWN_DETECTED AND CONTINUE_AFTER_BD THEN
        REPETITION_REGISTER := CURRENT_REPETITION;
        BD_REGISTER := 1;
        BLOCK_REGISTER := CURRENT_BLOCK;
        VOLT_REGISTER := VOLTAGE_SHOULD;
        UPDATE_TABLE := TRUE;

        NUM_BD := NUM_BD + 1;
        GO_TO_0 := TRUE;
        VOLTAGE_SHOULD := 0;
        OPERATION_STATUS := FALSE;
        BREAKDOWN_DETECTED := FALSE;

ELSIF PAUSE OR GROUNDING_PAUSE THEN
        IF NOT OSCI_DATA_STORED THEN
                STORE_OSCI;
```

```
        ELSIF NOT OSCI_ARMED THEN
                ARM_OSCI;
        END_IF;

ELSIF BREAKDOWN_DETECTED AND NOT CONTINUE_AFTER_BD AND NOT TEST_COMPLETED THEN
        REPETITION_REGISTER := CURRENT_REPETITION;
        BD_REGISTER := 1;
        BLOCK_REGISTER := CURRENT_BLOCK;
        VOLT_REGISTER := VOLTAGE_SHOULD;
        UPDATE_TABLE := TRUE;

        NUM_BD := NUM_BD + 1;
        VOLTAGE_SHOULD := 0;
        OPERATION_STATUS := FALSE;


        OSCI_DATA_STORED := FALSE;
        OSCI_ARMED := FALSE;

        TEST_COMPLETED := TRUE;
        BREAKDOWN_DETECTED := FALSE;

END_IF;

GET_TOTAL_TIME;

--------
```

This function block is executed when in Auto AC or DC modes and sequence is running, in order to control all the automatic phases. It consists in a big *IF* condition followed by some *ELSIF* statements, meaning that the part of the code that will be run will always be the first to be found true. The following ones, regardless they are true or not, they won't be executed.

The first part of the code is the one that will be executed usually during most of the sequence. It basically sets the frequency and the voltage for every moment while a block is running, and prepares the following one as soon as a block has ended.

The part with the second highest priority is the code executed when the sequence is ended or stopped by some reason, either manually or some safety feature. In that case it will stop the sequence, turn off the power source, set an order to ground the setup as soon as it will be discharged and bring the touch panel to the end screen for the Continuous mode.

Third part of the sequence code is executed when GO_TO_0 is active. As the variable name suggests, this block of code will disconnect the AC Source and discharge the circuit. As soon as the voltage is back to 0, it will check if this was the last repetition, and set TEST_COMPLETED to true if this was the case. Otherwise, it will add one to the number of repetitions, restore the current block number to one, send the command to start the pause between repetitions, and set the variable to ground for some seconds if programmed to do so. Finally, it will also set the necessary variables to store the data from the oscilloscope and let it ready for the next repetition.

If there is a Breakdown during the sequence, and it has been defined that the experiment should continue after a breakdown, the program will execute the fourth part of the function block. This piece of code will first register some information on when the breakdown happened and set the necessary variable for the touch panel to store it in Logging Buffer number 3 and then show it in a table. Right after it will stop the sequence and set the command to bring the voltage back to 0, so to execute the third part of the function block in the following cycle.

Fifth part of the code is the part that will be executed when in a pause or a grounding pause between repetitions. It uses the time to store the data from the oscilloscope and get it ready for the following sequence execution.

Last part of the *IF* condition in the function block will be executed if there has been a breakdown during the sequence but the program is been set no to continue after it. In this case, it will register the data from the breakdown and set the table to be updated, turn the operation off and set TEST_COMPLETED to true to end the automatic sequence.

Finally, every time the function block is executed, it calls also GET_TOTAL_TIME, in order to get the current duration of the sequence.

6.4.22. CALCULATE_VOLTAGE_SHOULD (FB)

```
KVOLT_DIFFERENCE := UINT_TO_INT(BLOCK_ARRAY[CURRENT_BLOCK].VOLT) -
UINT_TO_INT(BLOCK_ARRAY[CURRENT_BLOCK - 1].VOLT);

SLOPE := INT_TO_REAL(KVOLT_DIFFERENCE) /
UINT_TO_REAL(BLOCK_TIME_ARRAY[CURRENT_BLOCK]);

PREVIOUS_VOLT := UINT_TO_REAL(BLOCK_ARRAY[CURRENT_BLOCK - 1].VOLT);
CURRENT_ADD_VOLT := TIME_TO_REAL(BLOCK_MSECONDS) * SLOPE / 1000;
VOLTAGE_SHOULD_REAL := PREVIOUS_VOLT + CURRENT_ADD_VOLT;
VOLTAGE_SHOULD := REAL_TO_INT(VOLTAGE_SHOULD_REAL);
```

This piece of code is executed to calculate the nominal voltage in every moment of an automated continuous sequence. It first calculates the voltage difference between the current block and the previous one, and divides it by the duration of the current block in order to have the slope of this block. Then, multiplying this slope by the exact current time of the block stored in BLOCK_MSECONDS, it gets the voltage needed to be add to the one of the previous block to reach the value where the setup should be. This real value is converted to an integer and stored in VOLTAGE_SHOULD in order to read it from CONTROLLER function block.

6.4.23. PREPARE_NEXT_BLOCK (FB)

```
--------

IF CURRENT_BLOCK >= (TOTAL_BLOCKS_PLUS1 - 1) THEN
        REPETITION_REGISTER := CURRENT_REPETITION;
        BD_REGISTER := 0;
        BLOCK_REGISTER := 0;
        VOLT_REGISTER := 0;
        UPDATE_TABLE := TRUE;

        OPERATION_STATUS := FALSE;
        VOLTAGE_SHOULD := 0;
        GO_TO_0 := TRUE;

ELSE
        CURRENT_BLOCK := CURRENT_BLOCK + 1;
        START_BLOCK := TRUE;
END_IF;

--------
```

This function block prepares the following block when one is finished in the Automatic Continuous modes. If the block was the last one of the sequence, it registers that there has not been a breakdown and sets the command to update the corresponding logging buffer and table, before stopping the operation mode and activate the command to send the voltage to 0 (that will also prepare everything for the following repetition if needed).

6.4.24. PREPARE_IMPULSE (FB)

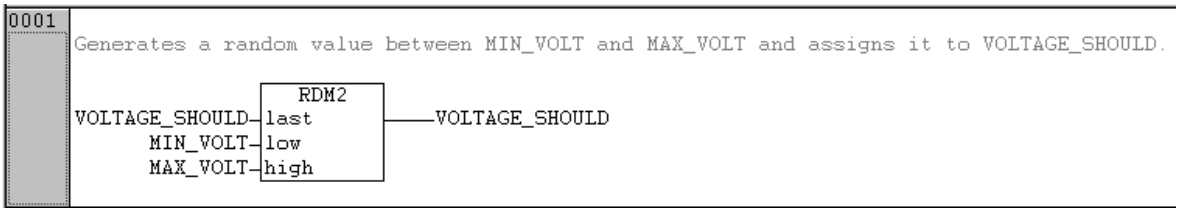
```
--------

IF NOT VOLTAGE_SELECTED THEN
        IF ORDERED_IMPULSE THEN
                VOLTAGE_SHOULD := START_VOLT;
        ELSIF RANDOM_IMPULSE THEN
                GENERATE_RANDOM_VOLTAGE;
        END_IF;

        MAX_VOLT_SEQUENCE := VOLTAGE_SHOULD;
        VOLTAGE_SELECTED := TRUE;
END_IF;

IF PREPARE_OSCI AND NOT OSCI_READY THEN
        PREPARE_OSCI_CONFIG;

ELSIF NOT PREPARE_OSCI AND NOT OSCI_READY THEN
        OSCI_READY := TRUE;

ELSIF SPARKGAP_READY AND OSCI_READY THEN
        IF NUMBER_OF_STAGES = 1 THEN
                SNR := 11;
        ELSIF NUMBER_OF_STAGES = 2 THEN
                SNR := 12;
        ELSIF NUMBER_OF_STAGES = 3 THEN
                SNR := 13;
        END_IF;
        READY := FALSE;
END_IF;

--------
```

PREPARE_IMPULSE (FB) is executed when, after setting all the parameters, “Ready” is pressed in the Impulse mode preparation screen of the touch panel.

First of all it prepares the starting voltage either taking the assigned one (in ordered mode) or generating a random value (for the random mode), if necessary. In that moment it sets the maximum voltage of the sequence to the starting one, as it’s the first and unique shot set.

Then, if the oscilloscope is set to be configured automatically, it executes PREPARE_OSCI_CONFIG until the oscilloscope is ready. When the voltage has been selected, the oscilloscope is ready and the SparkGap is also ready (it should have started to move as soon as the voltage was set), it changes the screen in the touch panel to the one to run the sequence. Depending on the number of stages, it will go to one or another of the three almost identical screens to run the Impulse sequence. They differ only in the maximum value the live graph can represent, in order to adapt a little bit the scale in the graph to the expected voltage values as it is not possible to escalate the data collected live.

6.4.25. GENERATE_RANDOM_VOLTAGE (FB)



This function block generates a random value between the defined minimum and maximum voltages, and assigns it to VOLTAGE_SHOULD. It uses the block RDM2 from the OSCAT library, that had to be downloaded from the internet as it is not included among the basic ones.

6.4.26. RUN_IMPULSE (FB)

```
--------

IF NUM_SHOTS < MAX_SHOTS AND NUM_BD < MAX_BD AND NOT TRIGGER_NOW AND NOT
VOLTAGE_LIMIT AND NOT SHOT_DONE AND NOT PAUSE AND NOT GROUNDING_PAUSE AND NOT
STOP AND NOT BREAKDOWN_DETECTED AND NOT GO_TO_0 AND NOT TEST_COMPLETED AND
SAFETY_STATUS AND NOT SANITY_DIVIDER AND NOT MAX_SOURCE  THEN
        IF SPARKGAP_READY AND NOT NOMINAL_VOLTAGE THEN
                OPERATION_STATUS := TRUE;
        ELSIF SPARKGAP_READY AND NOMINAL_VOLTAGE THEN
                TRIGGER_NOW := TRUE;
        END_IF;

ELSIF SHOT_DONE THEN
        NUM_SHOTS := NUM_SHOTS + 1;
        SHOTS_ARRAY[NUM_SHOTS].ACUM_TIME := NUM_SHOTS;
        SHOTS_ARRAY[NUM_SHOTS].VOLT := VOLTAGE_SHOULD;
        FIND_MAX_VOLT;

        TOTAL_BLOCKS_PLUS1 := NUM_SHOTS + 1;
        ESCALATE_GRAPH;

        GO_TO_0 := TRUE;
        VOLTAGE_SHOULD := 0;
        SHOT_DONE := FALSE;

ELSIF TEST_COMPLETED OR STOP OR NOT SAFETY_STATUS OR SANITY_DIVIDER OR MAX_SOURCE
THEN
        VOLTAGE_SHOULD := 0;
        RUNNING := FALSE;
        STOP := FALSE;
        OPERATION_STATUS := FALSE;
        VOLTAGE_LIMIT := FALSE;
        BREAKDOWN_DETECTED := FALSE;
        GROUND_SETUP := TRUE;

        IF NUMBER_OF_STAGES = 1 THEN
                SNR := 14;
        ELSIF NUMBER_OF_STAGES = 2 THEN
                SNR := 15;
        ELSIF NUMBER_OF_STAGES = 3 THEN
                SNR := 16;
        END_IF;

ELSIF GO_TO_0 THEN
        IF NOMINAL_VOLTAGE THEN
                PAUSE := TRUE;
                GO_TO_0 := FALSE;

                IF GROUND_BETWEEN_SEQUENCES THEN
                        GROUNDING_PAUSE := TRUE;
                END_IF;

                PREPARE_NEXT_SHOT;
                SPARKGAP_READY := FALSE;
                BREAKDOWN_DETECTED := FALSE;
        END_IF;

ELSIF (PAUSE OR GROUNDING_PAUSE) AND NOT OSCI_READY THEN
        IF NOT OSCI_DATA_STORED THEN
```

```
            STORE_OSCI;
        ELSIF PREPARE_OSCI AND NOT OSCI_READY THEN
            PREPARE_OSCI_CONFIG;
        ELSIF NOT PREPARE_OSCI AND NOT OSCI_ARMED THEN
            ARM_OSCI;
        END_IF;

ELSIF BREAKDOWN_DETECTED THEN
        STOP := TRUE;
END_IF;

GET_TOTAL_TIME;

--------
```

This function block is in charge of running the automatic sequence for the impulse mode. It works very similar to RUN_AUTO_CONTINUOUS, with a set of *IF* and *ELSIF* conditions that separate the different possible stages during an automatic sequence, giving priority to the higher ones.

The first condition block it's the basic code executed to charge the circuit and prepare a shot. It waits for the Spark Gap to be ready in order to switch on OPERATION_STATUS, and then it waits for the nominal voltage to be reached in order to set the command to trigger the Spark Gap.

The second part is the one that will be executed after a shot has been done. It will first store the values of the shot in the corresponding arrays to then find the maximum voltage of the sequence and escalate and update the graph representing it. Then it will activate GO_TO_0 to bring the voltage back to zero and prepare the following shot.

The part of the code with third highest priority is the one in charge of stopping the sequence when it's ended or has been stopped manually or due to any problem. In this case, it will stop the operation, set the voltage to zero, send the command to ground the setup and change the screen to the one of the finished Impulse sequence in the touch panel.

The fourth part of the function block will be executed when GO_TO_0 is true. It will wait until voltage reaches zero and then it will start the pause between shots and set the command to ground the setup if configured to do so between shots. It will also call the function block to prepare next block and set the Spark Gap as not ready, so that it adjusts again to the new voltage that will be set.

Fifth block in the code is the one executed while in pause or grounded pause. Like in the continuous mode, it will take this time to store the data from the oscilloscope in the CF card and to recalibrate the oscilloscope according to the new voltage values if configured to do so. It will finally arm the oscilloscope to let it ready to capture the following shot.

The final part of the *IF* conditions in the function block checks if a Breakdown occurs before the actual triggering. In such a case, the sequence will stop.

Finally, in each cycle like in the Continuous mode, GET_TOTAL_TIME will be called to show the total duration of the sequence till the moment.

6.4.26.1. FIND_MAX_VOLT (Action)

```
N := 1;
WHILE N <= NUM_SHOTS DO
        IF SHOTS_ARRAY[N].VOLT > MAX_VOLT_SEQUENCE THEN
                MAX_VOLT_SEQUENCE := SHOTS_ARRAY[N].VOLT;
        END_IF
        N := N + 1;
END_WHILE;
```

As in the Continuous mode, this short action will look for the maximum voltage among the already shot values during the sequence, in order to adapt the axis in the graphs to it.

6.4.27. PREPARE_NEXT_SHOT (FB)

```
--------

IF ORDERED_IMPULSE THEN
        IF BREAKDOWN_DETECTED THEN
                IF SIGN_AFTER_BD THEN
                        VOLTAGE_SHOULD := VOLTAGE_SHOULD + AFTER_BD;
                ELSIF NOT SIGN_AFTER_BD THEN
                        VOLTAGE_SHOULD := VOLTAGE_SHOULD - AFTER_BD;
                END_IF;

                NUM_BD := NUM_BD + 1;
                BREAKDOWN_DETECTED := FALSE;

        ELSIF NOT BREAKDOWN_DETECTED THEN
                IF SIGN_AFTER_NO_BD THEN
                        VOLTAGE_SHOULD := VOLTAGE_SHOULD + AFTER_NO_BD;
                ELSIF NOT SIGN_AFTER_NO_BD THEN
                        VOLTAGE_SHOULD := VOLTAGE_SHOULD - AFTER_NO_BD;
                END_IF;
        END_IF;

        IF VOLTAGE_SHOULD < 10 OR VOLTAGE_SHOULD > (140 * NUMBER_OF_STAGES) THEN
                VOLTAGE_LIMIT := TRUE;
        END_IF;

ELSIF RANDOM_IMPULSE THEN
        GENERATE_RANDOM_VOLTAGE;
        IF BREAKDOWN_DETECTED THEN
                NUM_BD := NUM_BD + 1;
                BREAKDOWN_DETECTED := FALSE;
        END_IF;
END_IF;

IF NUM_SHOTS >= MAX_SHOTS OR NUM_BD >= MAX_BD OR VOLTAGE_LIMIT THEN
        TEST_COMPLETED := TRUE;
END_IF;

VARIABLES_PREPARED := FALSE;
OSCI_CHANNELS_CONFIGURED := FALSE;
OSCI_READY := FALSE;
OSCI_DATA_STORED := FALSE;
OSCI_ARMED := FALSE;

--------
```

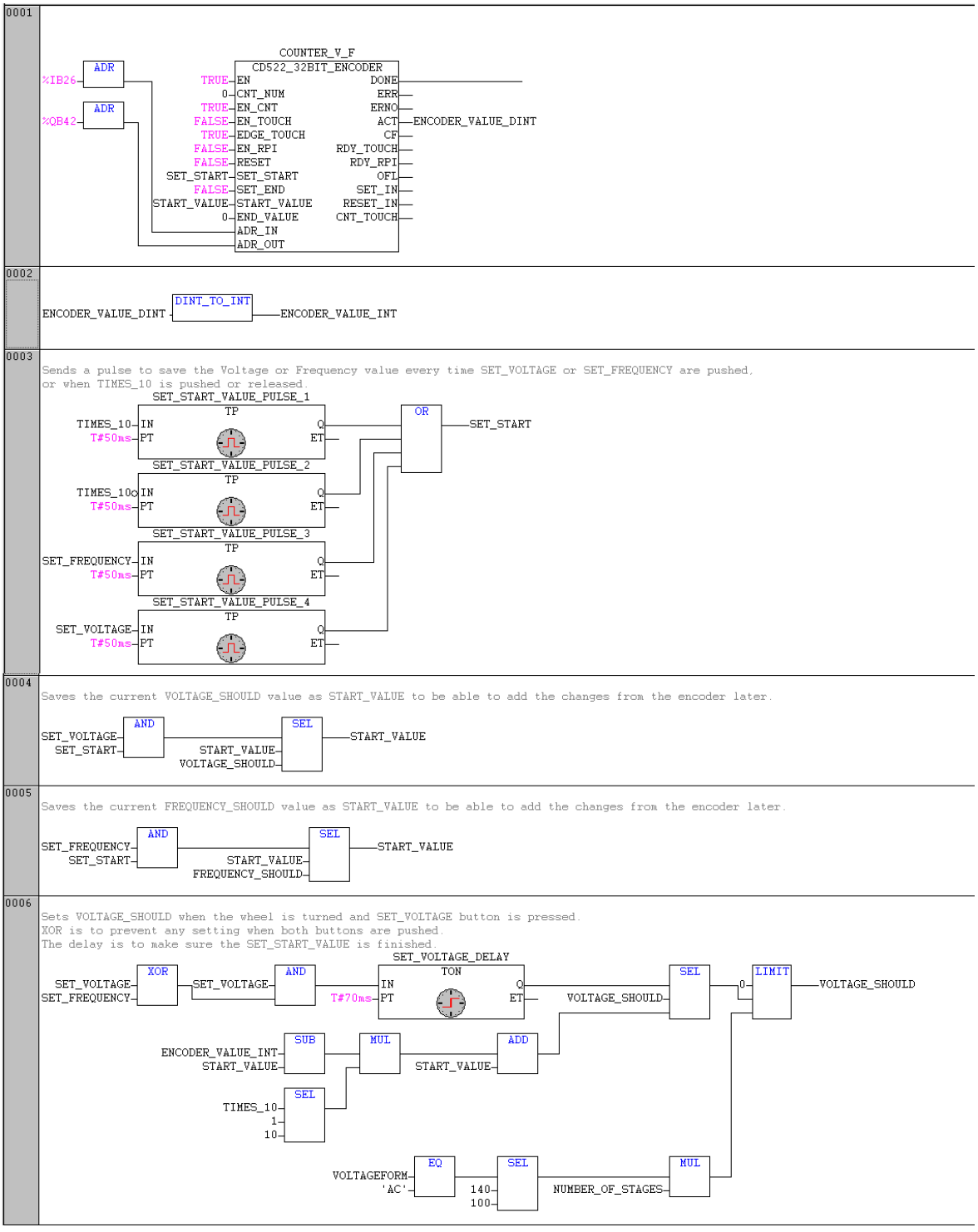
This function block has the task to prepare the following shot while in an automatic Impulse sequence.

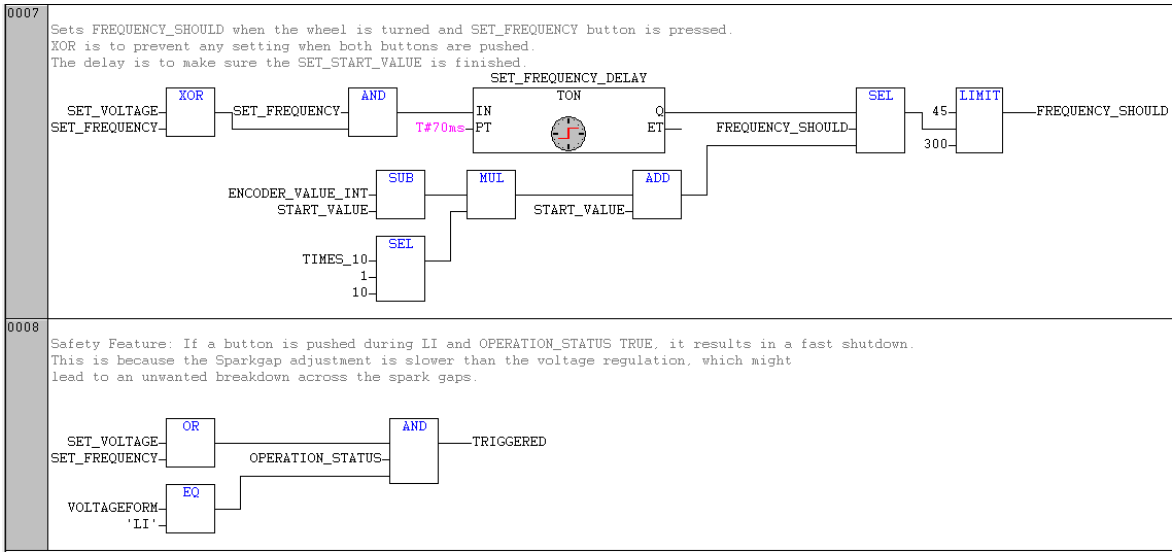
If the sequence is in Ordered mode, it checks if there has been a breakdown and then it sets the next voltage to shoot according to what has been defined to be done in every case. It also checks if this new voltage is higher than the highest possible (140 kV multiplied by the number of stages), or lower than 10 kV. In these cases it will set VOLTAGE_LIMIT to true to end the sequence. If in Random mode, it calls the function block to generate a new random voltage and checks if there has been a breakdown to count it or not.

After this, it also checks if any of the stated limits for number of shots or number of breakdowns or the limits of voltage have been reached, in order to then set TEST_COMPLETED to true and finish the sequence.

Finally, it also resets the variables that indicate that the oscilloscope is ready in order to, if needed, configure it again according to the new values.

6.4.28. SET_V_F (FB)





SET_V_F is a function block executed in the manual mode mainly to set the voltage and frequency through the buttons in the Console.

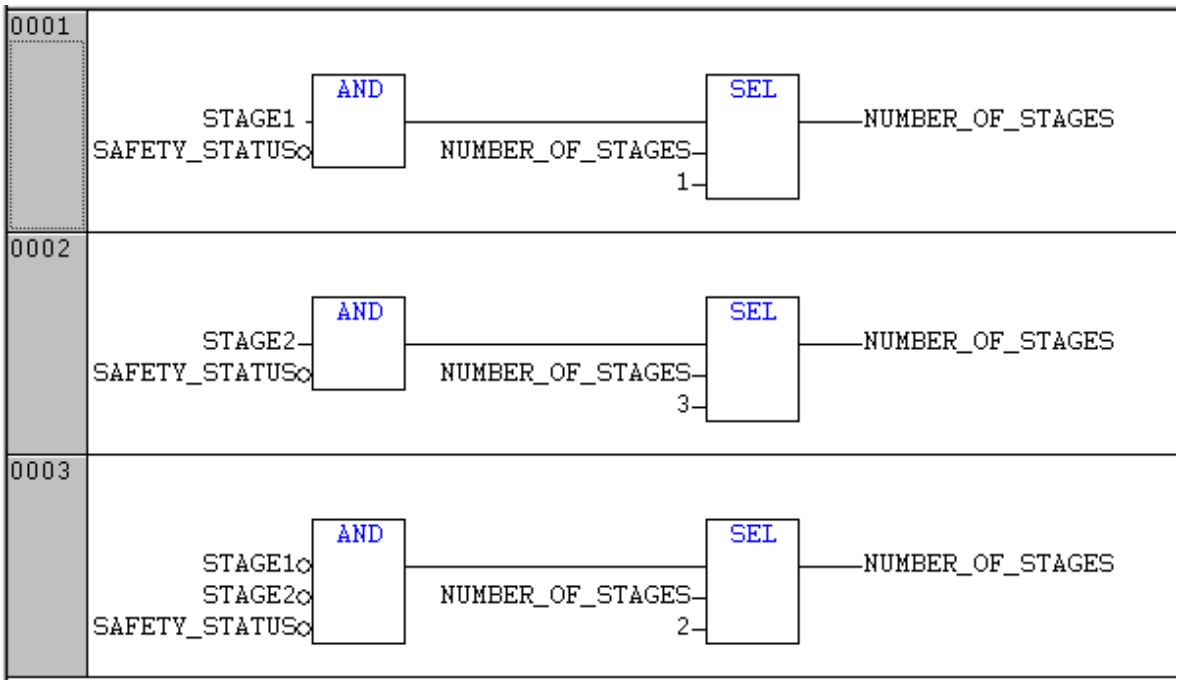
First network is the Encoder module that reads the turning of the wheel to set the different values, and it's converted to a normal integer in the second network. The third network is used to set the encoder top start counting when any of the set voltage or frequency buttons are pushed, or whenever there is a change in the state of the button that allows to change the value 10 times faster (this is to detect both when pressed and released and set the encoder to start counting again).

In networks four and five, whenever SET_VOLTAGE or SET_FREQUENCY are pressed, it stores the current voltage or frequency value in order to add later the difference set through the wheel and captured with the encoder.

Networks six and seven are the ones in charge of setting VOLTAGE_SHOULD and FREQUENCY_SHOULD when the wheel is turned. They take into account if TIMES_10 button is pressed or not, and also that the voltage has to be between zero and a value dependent of the voltage form and the number of stages.

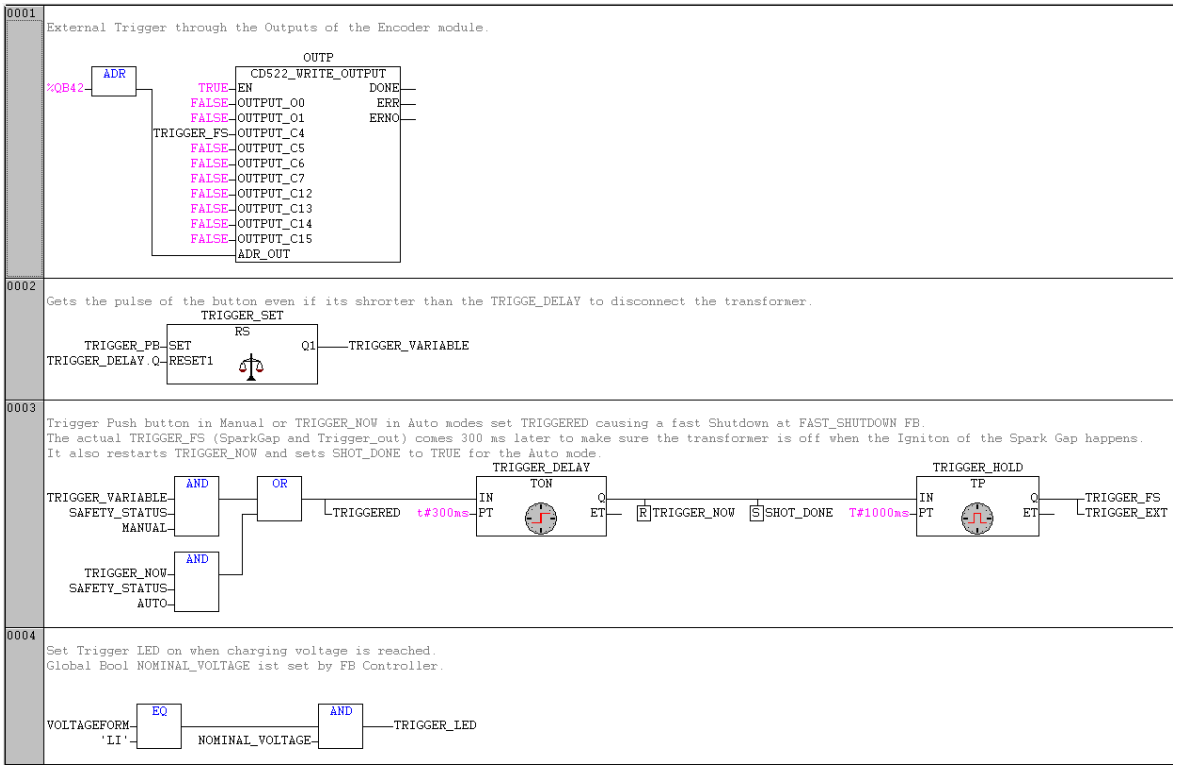
Finally, the eight network is a safety feature to prevent any adjustment in Impulse mode while voltage is being applied, as the Spark Gap is already adjusted for a certain voltage and due to its slow movement to get ready again it could lead to an undesired breakdown across it.

6.4.29. NUMBER_OF_STAGES_SELECT (FB)



This function block is executed in the Manual mode in order to read the position of the number of stages selector in the Console. It basically reads two variables that are set directly by the position of the selector in the Console, and depending on these values it sets NUMBER_OF_STAGES to 1, 2 or 3.

6.4.30. TRIGGER (FB)



TRIGGER function block is the POU in charge of triggering the Spark Gap in the Impulse modes.

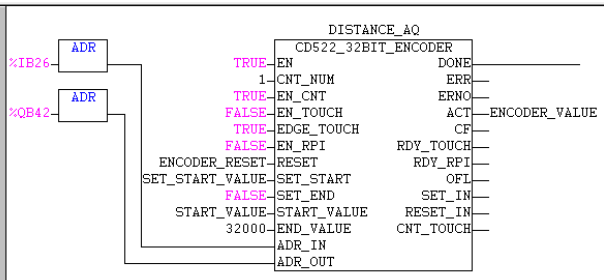
The first network is a representation of the Encoder Module, to set variables to its output. At the moment the output C4 is set to the value of TRIGGER_FS variable, the one used to physically trigger the Spark Gap.

Second network is used to detect the pulsation of the trigger button even if it's done very fast. It sets TRIGGER_VARIABLE, that will be used in network three to activate the trigger. If this variable in Manual mode or TRIGGER_NOW in Auto mode are active and there is SAFETY_STATUS, it will set true the variable TRIGGERED and start a 300 milliseconds delay timer in order to shut down the power source before triggering the Spark Gap. When the timer is finished it will reset the values used to start the triggering sequence, set SHOT_DONE for the Auto mode and send a pulse to the Spark Gap during one second to make sure it triggers.

The last network is switching ON the yellow LED in the Trigger button in the Console as soon as the Nominal Voltage is reached while being in Impulse mode.

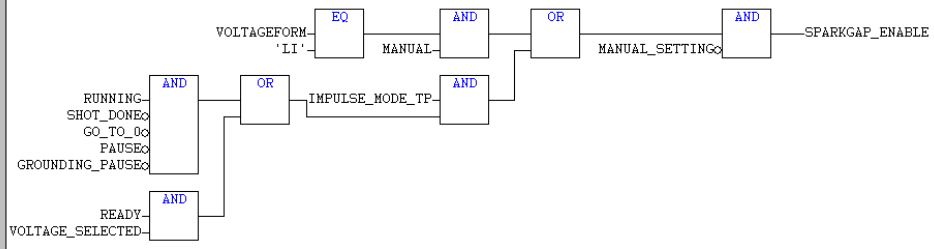
6.4.31. DISTANCE_CTRL (FB)

0001



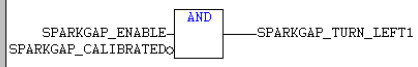
0002

If Impulse mode and not setting the SparkGap manually, enable SparkGap.



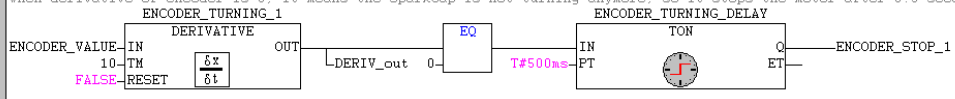
0003

Calibrate SparkGap: Turn to the left until it's in the 0 position (balls touching).



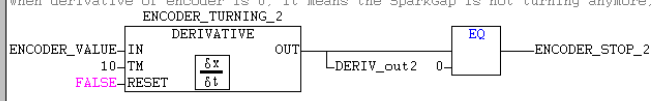
0004

When derivative of encoder is 0, it means the SparkGap is not turning anymore, so it stops the motor after 0.5 seconds.



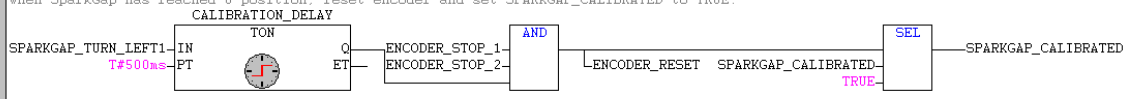
0005

When derivative of encoder is 0, it means the SparkGap is not turning anymore, so it stops the motor.



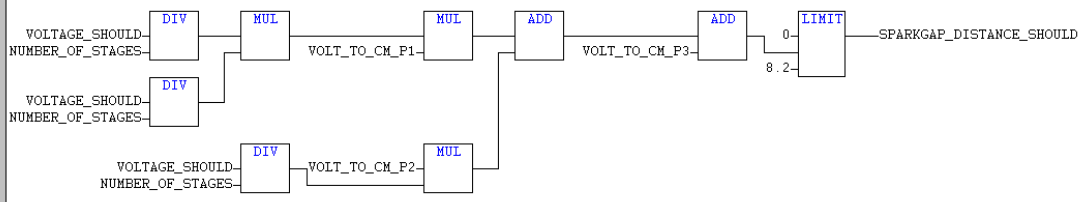
0006

When SparkGap has reached 0 position, reset encoder and set SPARKGAP_CALIBRATED to TRUE.



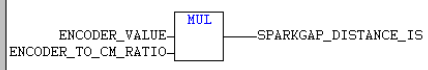
0007

Here goes the calibration curve of the sparkgaps.
Has to be checked.



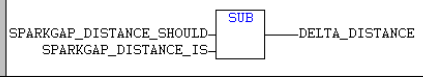
0008

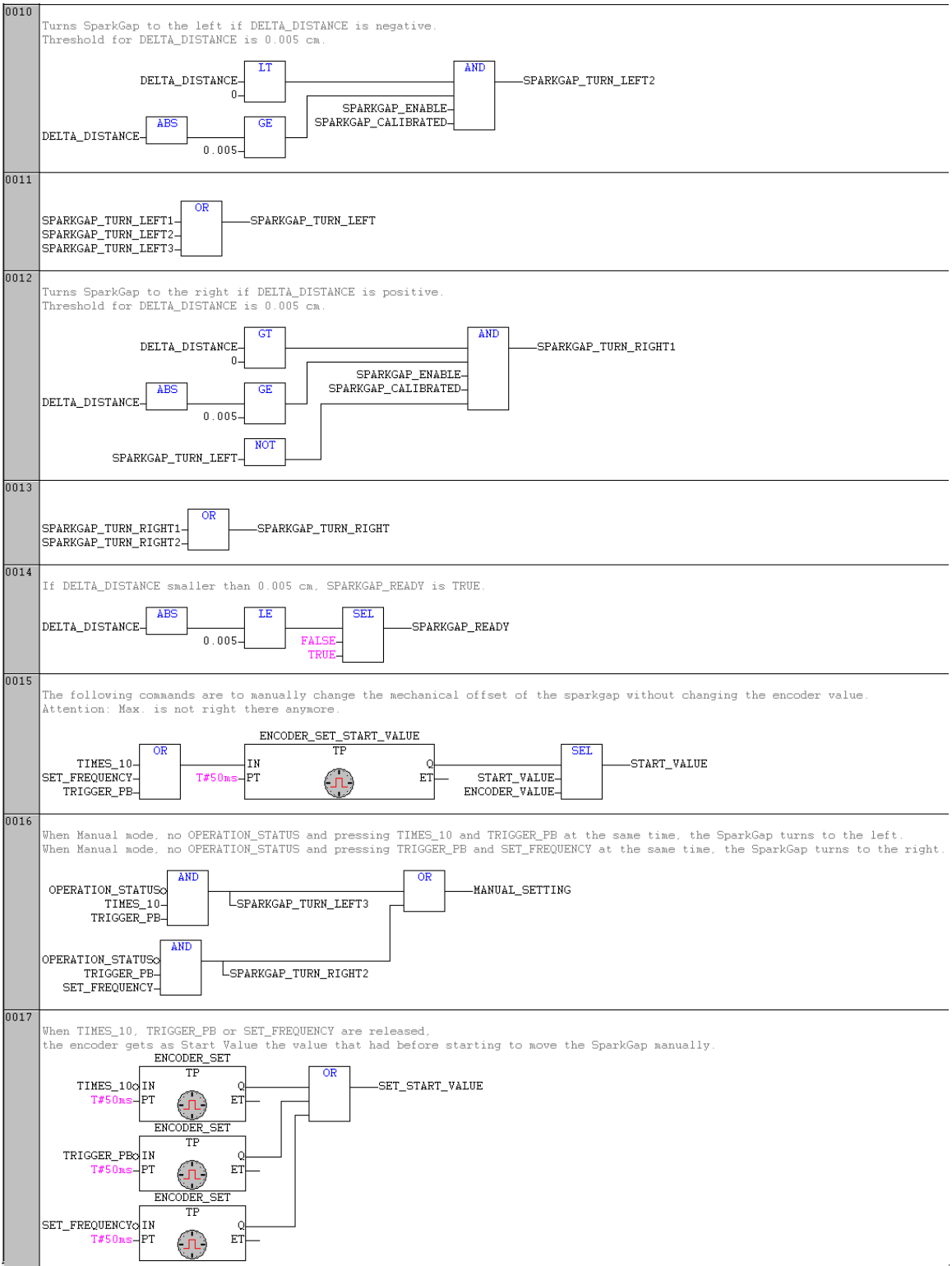
Check the current position of the SparkGap.



0009

Calculate the difference between distance that should be and that actually is in the SparkGap.





DISTANCE_CTRL is the POU that moves the Spark Gap motor and sets it to the right position according to the voltage set.

The first network has the Encoder module with the input variables to reset it or change the starting value, and the encoder value as output.

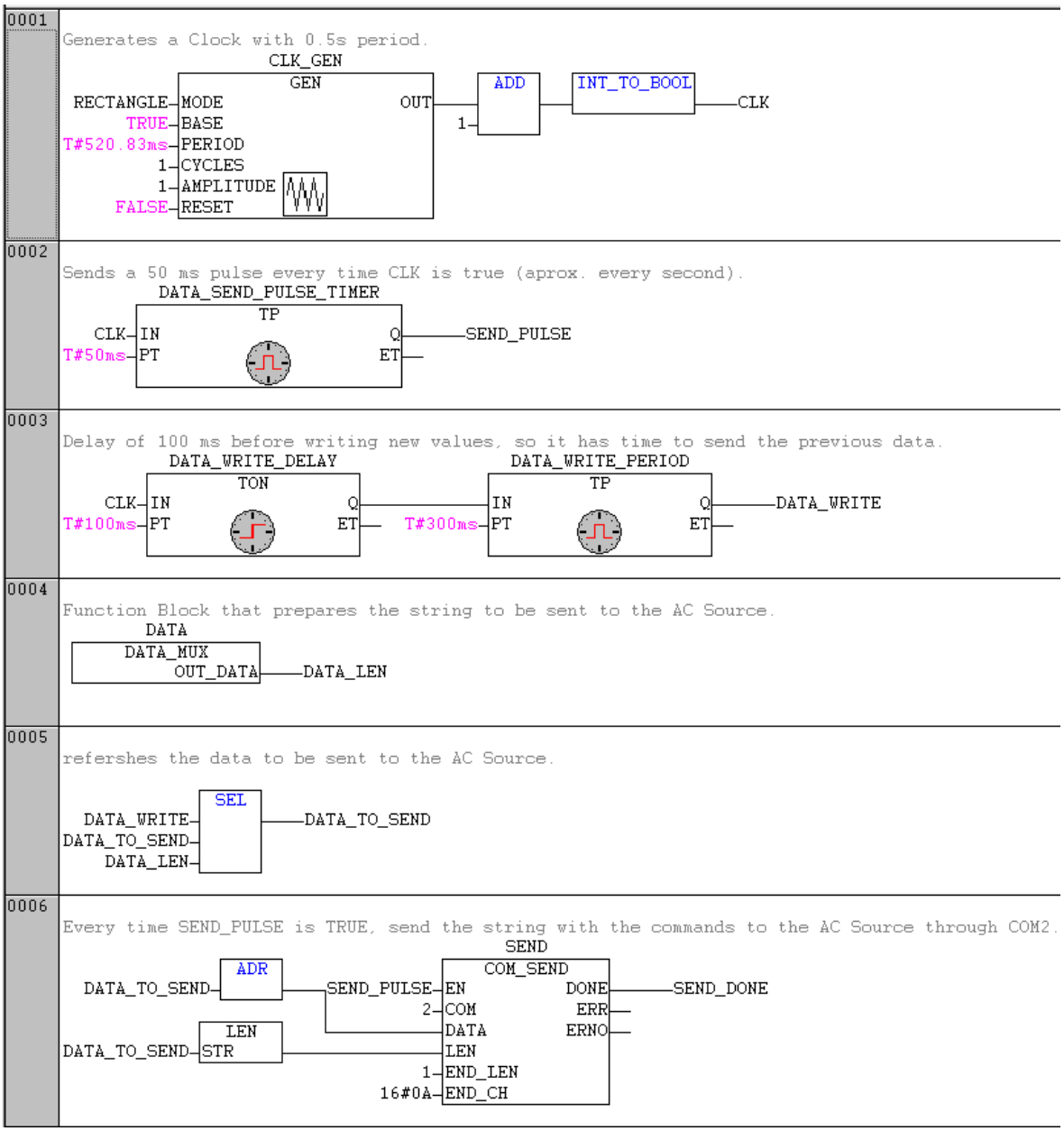
In the second network it enables the Spark Gap, not allowing the automatic adjustment of it unless in Manual Impulse mode or in some phases of the Auto impulse mode, and always in case that it is not being moved manually.

In the third to sixth networks it turns the Spark Gap to the left until the encoder detects no more movement, meaning the balls are touching one to each other. In that moment it stops the motor, resets the encoder and sets SPARKGAP_CALIBRATED to true, meaning the Spark Gap is in the zero position and can start to be regulated.

In network number seven the calibration curve for the Spark Gap is implemented, with the parameters to set the distance of the Spark Gap in relation to the nominal voltage that wants to be shoot. Networks eight and nine measure the current distance in the Spark Gap with the encoder, and compare it with the desired distance calculated in network seven. If this distance is negative and smaller than 0.005 cm, network number ten will command the SparkGap to turn left. Network twelve will make the same but if the distance is positive, turning the motor to the right then. Finally, when the difference between desired distance and measured distance is smaller than 0.005 cm, network fourteen will set the variable SPARKGAP_READY to true.

In the three last networks, a system is implemented to turn the Spark Gap manually, useful to do some adjustments if, for example, the setup is not triggering with the distance calculated. To move the Spark Gap to the left, “10x” and “Trigger” buttons need to be pressed at the same time; while to move it to the right, the combination to be pressed is “Trigger” and “Frequency Set” buttons. Always assuming that there is no Operation Status, otherwise it will not work. In order to avoid the Spark Gap wanting to come back to the original position after being turned manually, when any of the three buttons used in this manual mode are pressed, the encoder value is saved, to be set again as Start Value of the encoder as soon as any of the three buttons is released. Using this little trick, the changes in the encoder while moving the Spark Gap manually are not reflected in the actual encoder value.

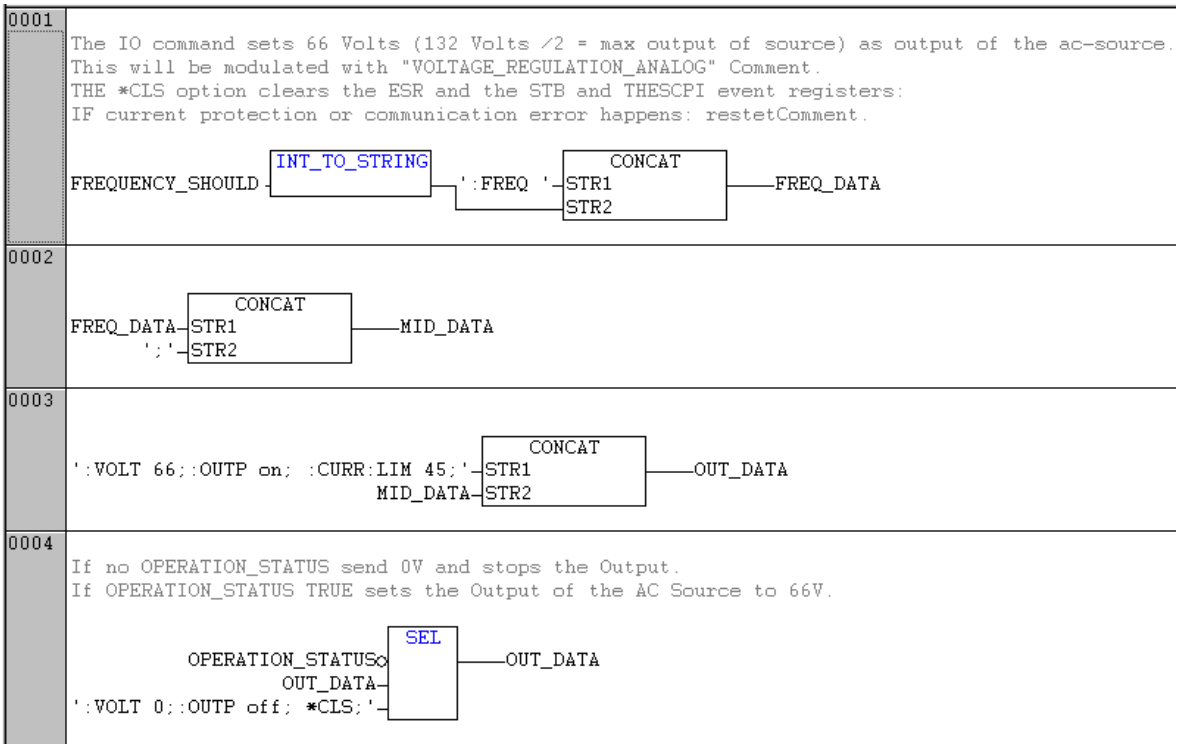
6.4.32. RS232 (FB)



RS232 is the function block regulating the communication between COM2 in the PLC and the AC Source.

It does so with a clock that changes every half a second. When CLK is set true in network number one, the second network sends a 50 ms pulse that will activate the COM2 port and send the data in network number 6. The string to be sent is prepared in the fourth network, where DATA_MUX is called, and refreshed 100 ms after CLK is set true (in order to give time to the COM port to send the previous data) in networks number three and five.

6.4.33. DATA_MUX (FB)



This POU prepares the string that will be sent to the AC Source through the COM2 port.

If there is Operation Status, it first takes prepares a string with the frequency that needs to be set; then it adds a command to set the voltage output to 66V (half of 132V, the maximum output of the AC Source); third a line to set the Output as ON; and it finally adds a limitation to 45 for the current. The voltage output is set to half of the maximum value in order to later regulate it through VOLTAGE_REGULATION_ANALOG in FAST_SHUTDOWN.

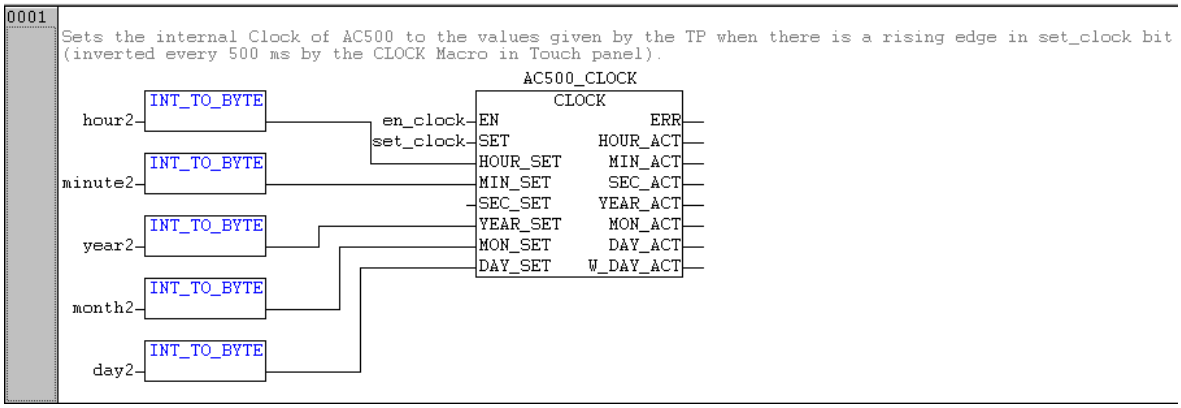
If there is no Operation Status, it prepares a command to set the voltage to zero, the output in OFF, and clear the event registers (through the *CLS option).

6.4.34. TPClock_to_AC500 (FB)

```
minute2 := BCD_TO_INT(WORD_TO_BYTE(wTime1));
hour2 := BCD_TO_INT(WORD_TO_BYTE(ROR(wTime1,8)));
day2 := BCD_TO_INT(WORD_TO_BYTE(wTime2));
month2 := BCD_TO_INT(WORD_TO_BYTE(ROR(wTime2,8)));
year2 := BCD_TO_INT(WORD_TO_BYTE(wTime3));
day_of_week2 := BCD_TO_INT(WORD_TO_BYTE(ROR(wTime3,8)));

set_clock := bSetClock;

Set_Clock_AC500;
```

This short piece of code is used to copy the time from the Touch Panel to the PLC. It first reads the time from the variables forming the Time Block in the Touch Panel, and then calls the action *Set_Clock_AC500* in order to store them in the real time clock of the PLC.

6.4.34.1. Set_Clock_AC500 (Action)



This small action sets the time values gotten from the touch panel to the internal clock in the PLC.

To do so, it relies on *bSetClock*, a bit controlled by the CLOCK Macro in the Touch Panel and used to set the AC500_CLOCK through the variable *set_clock*. The CLOCK Macro in the Touch Panel changes the value of the bit every half a second, allowing to update the clock in the PLC every second when there is a rising edge.

6.4.35. PREPARE_OSCI_CONFIG (FB)

```
--------

IF NOT VARIABLES_PREPARED THEN
        IF MODE_AC_TP OR MODE_DC_TP THEN
                AUX_VDIV_REAL := UINT_TO_REAL(MAX_VOLT_SEQUENCE);
        ELSIF IMPULSE_MODE_TP THEN
                AUX_VDIV_REAL := UINT_TO_REAL(VOLTAGE_SHOULD);
        END_IF;

        VOLT_DIVISION_REAL := AUX_VDIV_REAL * 1100 / 400;
        VOLT_DIVISION_CP := REAL_TO_UINT(VOLT_DIVISION_REAL);

        VOLT_DIVISION := VOLT_DIVISION_CP * 5;
        VOLT_DIVISION_AC := VOLT_DIVISION * 2;
        VDIV_SUFFIX := 'MV';

        IF VOLT_DIVISION > 10000 THEN
                VOLT_DIVISION := 10000;
        END_IF

        OFFSET := REAL_TO_UINT(UINT_TO_REAL(VOLT_DIVISION) * 4 * 0.9);
        OFFSET_AC := 0;

        VARIABLES_PREPARED := TRUE;
END_IF

IF NOT TIME_DIV_CONFIGURED THEN
        SET_TIME_DIV;

ELSIF NOT OSCI_CHANNELS_CONFIGURED THEN
        IF OSCI_CHANNEL <= 4 THEN
                SEND_OSCI_CHANNELS_CONFIG;
        ELSE
                OSCI_CHANNELS_CONFIGURED := TRUE;
        END_IF;

ELSIF NOT OSCI_READY THEN
        SEND_OSCI_CONFIG;
END_IF;

--------
```

This is the function block that prepares the commands to send through the COM1 port in the PLC to the Oscilloscope to set it automatically.

The first *IF* condition takes care of preparing the variables for the volts per division, offset, etc. To get the volts per division, it takes the maximum voltage of the sequence for the continuous mode or the voltage of the following shot for the Impulse mode and, after converting it to mV, it multiplies it by 1.1 in order to have some allowance in the screen (this will be the highest voltage wanted to be shown in the screen). Then, it divides it by 8 divisions in the screen for the DC and Impulse voltage measurements; and by 4 for the Current and AC Voltage measurements (they don't have polarity, so the offset will be set in the middle of the screen, allowing half of the divisions to represent the same voltage.

After calculating the volts per division, it will set the offset to 0 for the AC or Current Probe measures, and to VOLT_DIVISION multiplied by 4 (number of divisions in half a screen) and by 0.9 (to give some allowance). Later on, when the command will be set, the polarity will set if this offset is in the top or bottom of the screen.

Once these variables are ready it starts to send commands to the oscilloscope. The first thing to be configured are the times per division, so that it calls SET_TIME_DIV function block. When the times per division are set, it tries to set the specific parameters for each of the four channels in the oscilloscope, calling SEND_OSCI_CHANNELS_CONFIG many times. Finally, it calls SEND_OSCI_CONFIG to set some general parameters of the oscilloscope.

6.4.36. SET_TIME_DIV (FB)

```
--------

IF MODE_AC_TP OR MODE_DC_TP THEN
        TIME_DIVISION := 200;
        TDIV_SUFFIX := 'US';
ELSIF LI_MODE THEN
        TIME_DIVISION := 200;
        TDIV_SUFFIX := 'US';
ELSIF SI_MODE THEN
        TIME_DIVISION := 10;
        TDIV_SUFFIX := 'MS';
ELSIF PT_MODE THEN
        TIME_DIVISION := 20;
        TDIV_SUFFIX := 'US';
ELSIF DEF_MODE THEN
        CALC_TIME_DIV;
END_IF;

SEND_TIME_DIV;

--------
```

As its name says, this function block sets the time per division in the Oscilloscope. It first assigns specific values to the variables TIME_DIVISION and TDIV_SUFFIX, to finally send them to the oscilloscope calling the function block SEND_TIME_DIV.

6.4.36.1. CALC_TIME_DIV (Action)

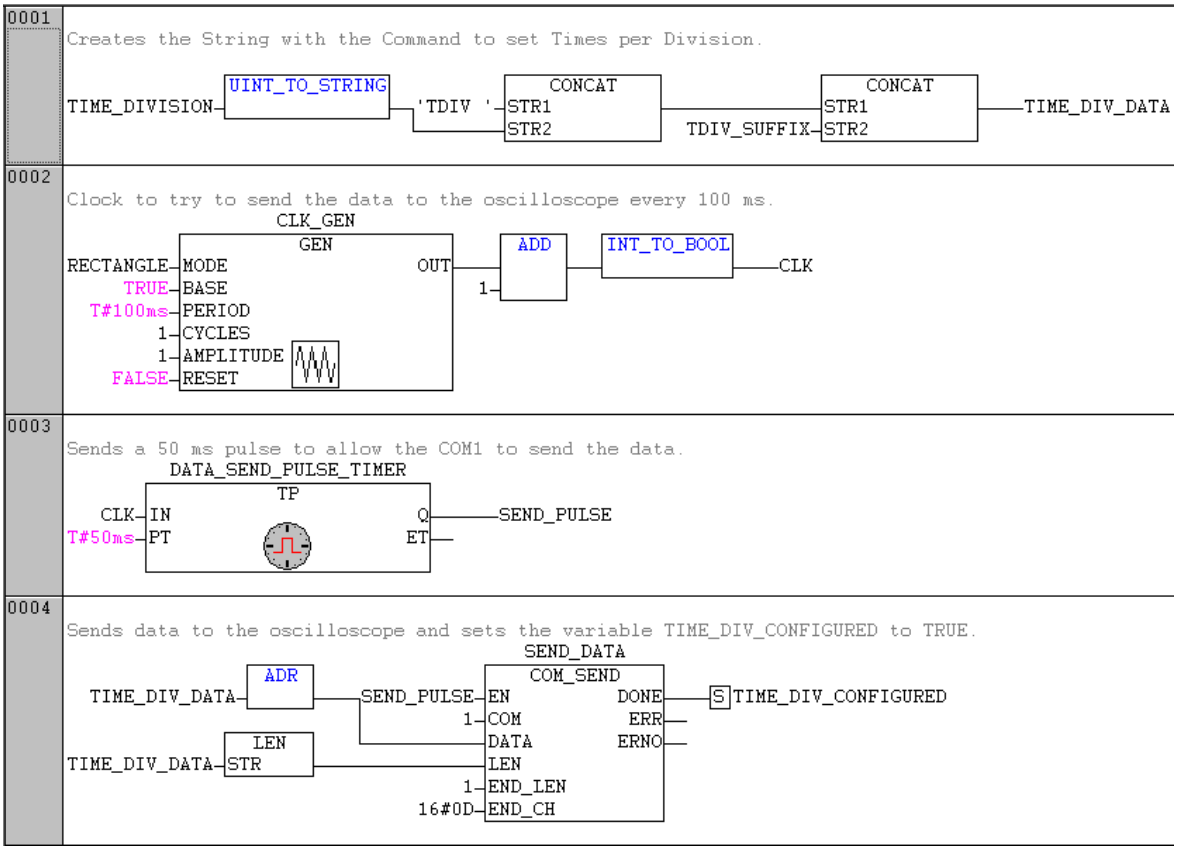
```
--------

AUX_TDIV_REAL := UINT_TO_REAL(3*T2 + 2*T1) * 1000 / 10;

N := 0;
WHILE AUX_TDIV_REAL >= 10 DO
        AUX_TDIV_REAL := AUX_TDIV_REAL / 10;
        N := N +1;
END_WHILE;

IF AUX_TDIV_REAL <= 1 THEN
        TIME_DIVISION := 1;
ELSIF AUX_TDIV_REAL > 1 AND AUX_TDIV_REAL <= 2 THEN
        TIME_DIVISION := 2;
ELSIF AUX_TDIV_REAL > 2 AND AUX_TDIV_REAL <= 5 THEN
        TIME_DIVISION := 5;
ELSIF AUX_TDIV_REAL > 5 THEN
        TIME_DIVISION := 10;
END_IF;

IF N <= 2 THEN
        TDIV_SUFFIX := 'NS';
        IF N > 0 THEN
                TIME_DIVISION := TIME_DIVISION * 10 * N;
        END_IF;
ELSIF N >= 3 AND N <= 5 THEN
        TDIV_SUFFIX := 'US';
        IF N > 3 THEN
                TIME_DIVISION := TIME_DIVISION * 10 * (N - 3);
        END_IF;
ELSIF N >= 6 AND N <= 8 THEN
        TDIV_SUFFIX := 'MS';
        IF N > 6 THEN
                TIME_DIVISION := TIME_DIVISION * 10 * (N - 6);
        END_IF;
ELSIF N >= 9 AND N <= 11 THEN
        TDIV_SUFFIX := 'S';
        IF N > 9 THEN
                TIME_DIVISION := TIME_DIVISION * 10 * (N - 9);
        END_IF;
ELSIF N >= 12 THEN
        TDIV_SUFFIX := 'KS';
        TIME_DIVISION := 1;
END_IF;

--------
```

CALC_TIME_DIV is an action used to calculate times per division from the T1 and T2 values entered manual in Automatic Impulse mode.

T1 and T2 define the rising and decay time of the impulse wave, and are introduced in microseconds. So that, in order to get the total time wanted to be seen in the screen, T2 is multiplied by 3 and added to T1 multiplied by 2. Then, the resulting value is converted to nanoseconds and divided by 10 divisions in the screen, to get the desired times per division.

As the oscilloscope can only accept times per division values from 1 ns to 1ks, following always the series 1, 2, 5, the calculated value will have to be converted to any of these values. To do so, the real number in ns is first divided by 10 until it's smaller than 10, storing the number of times it has been divided. Once a 1 digit number has been found, it's classified to its immediate superior value among 1, 2, 5 or 10. Finally, using the number of times it was divided by 10, it can be classified again to any of the scale units available and the final TIME_DIVISION integer value can be calculated.

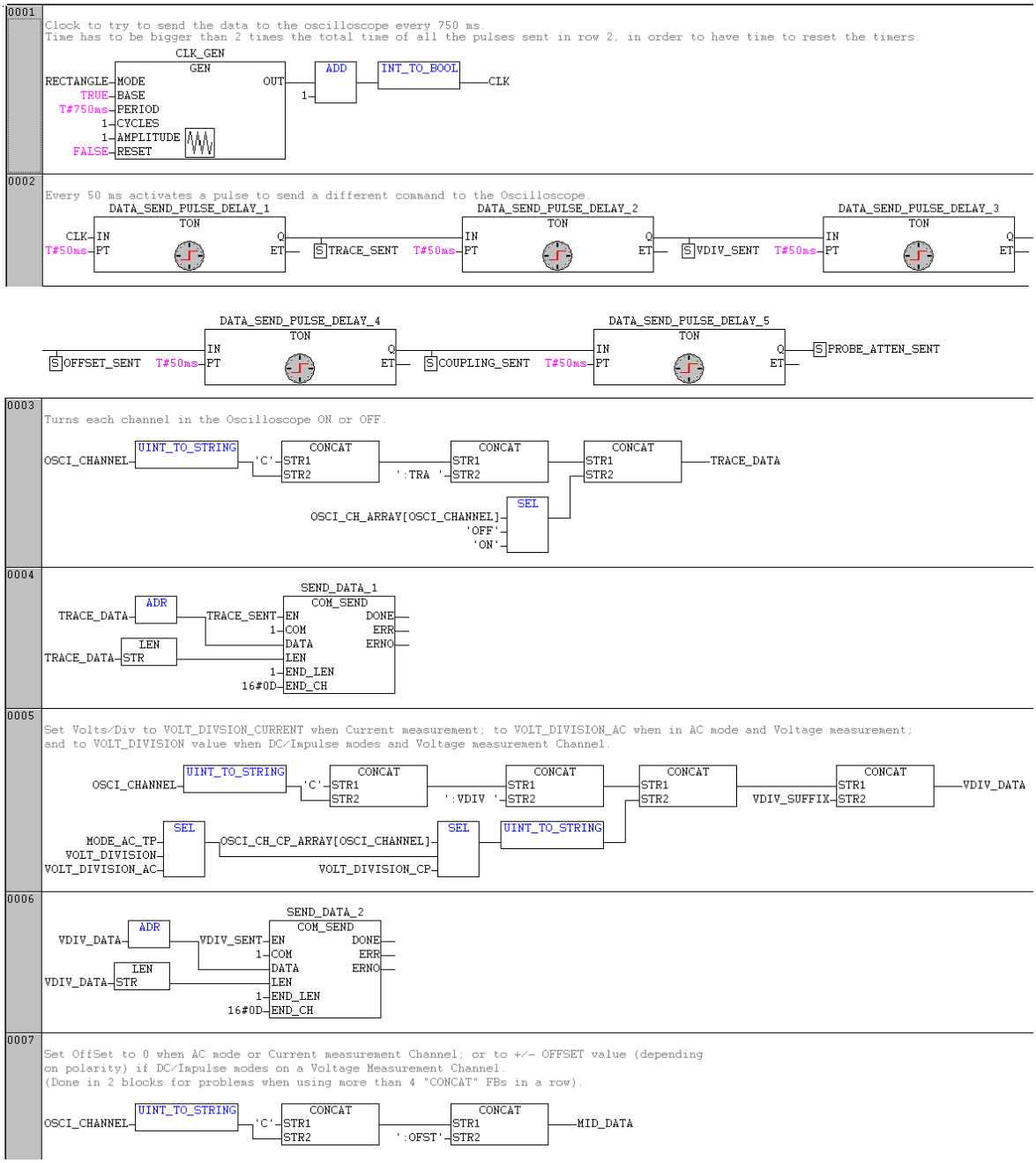
6.4.36.2. SEND_TIME_DIV (Action)



This POU written in FBD is the one sending the volts per division command to the oscilloscope.

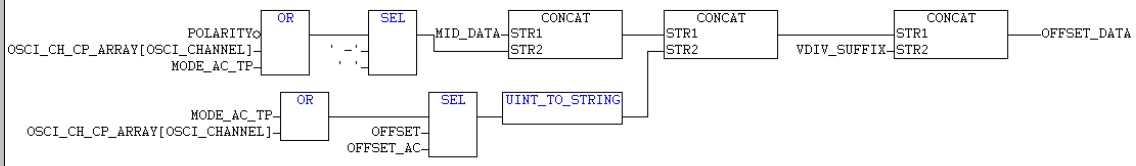
To do so, it first creates a string with the data to be sent. Then, it sets a 100 ms clock that will trigger a 50 ms pulse to activate the COM1 port and send the data. As soon as the transmission is done, TIME_DIV_CONFIGURED will be set to true, to allow to go further in the sequence.

6.4.37. SEND_OSCI_CHANNELS_CONFIG (FB)

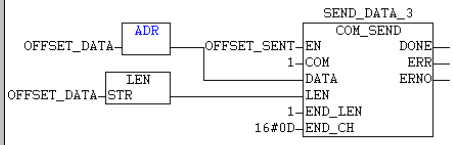


0008

Set OffSet to 0 when AC mode or Current measurement Channel; or to +/- OFFSET value (depending on polarity) if DC/Impulse modes on a Voltage Measurement Channel. (Done in 2 blocks for problems when using more than 4 "CONCAT" FBs in a row).

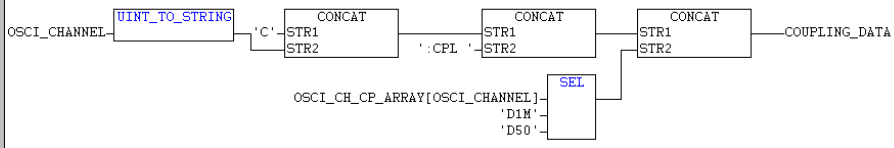


0009

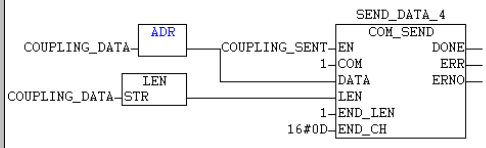


0010

Set Coupling to DC - 1 MOhm when Voltage measurement Channel; or to DC - 50 Ohm when Current measurement Channel.

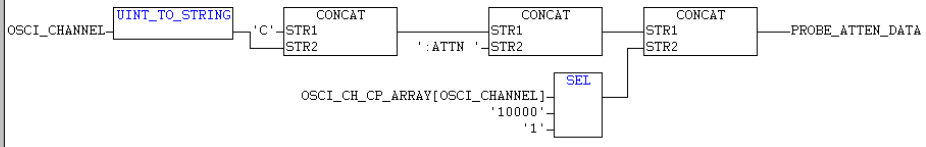


0011

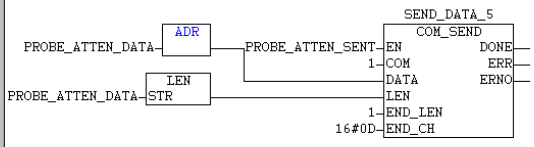


0012

Set Probe Attenuation to 10k when Voltage measurement Channel; or to 1 when Current measurement Channel.

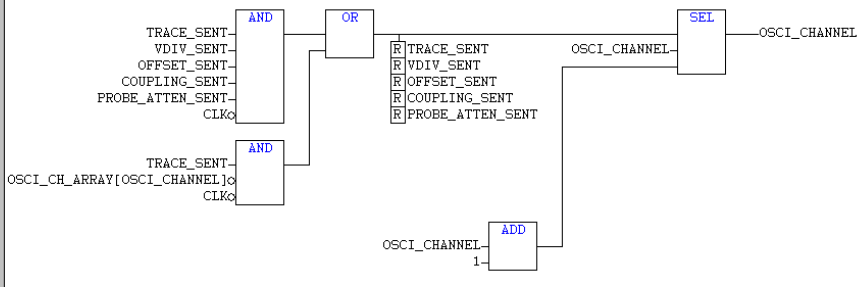


0013



0014

If everything is sent, resets the variables and goes to the following Oscilloscope Channel Number.



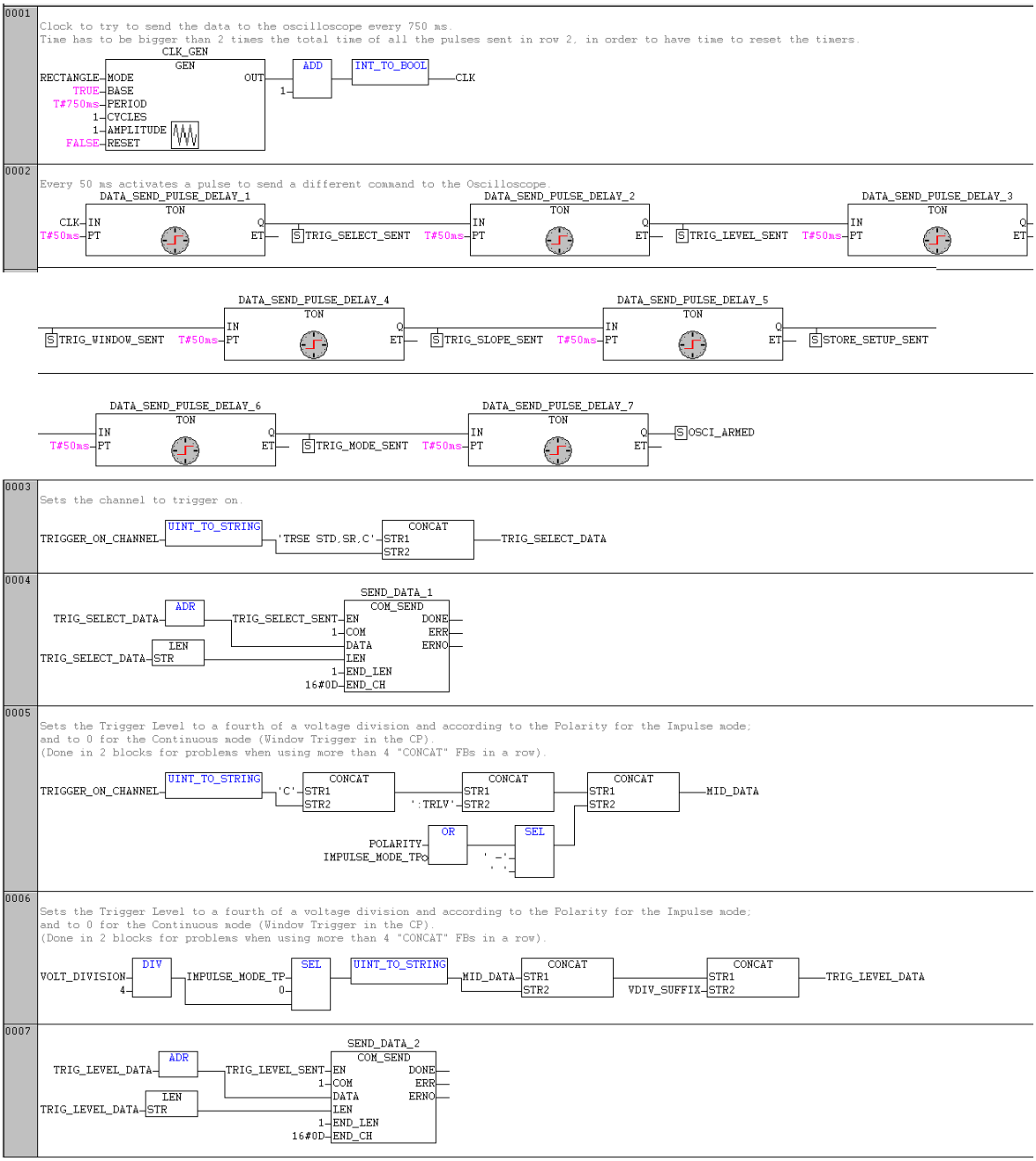
SEND_OSCI_CHANNELS_CONFIG is the function block in charge of sending the commands to configure each channel in the oscilloscope to the desired values. It is called several times until all the four channels have been configured.

It works the same way as the other blocks to send commands to the oscilloscope. It first sets a clock, which will try to send pulses to activate the COM1 port during half of its period. In network number 2, a succession of “Timer ON Delay” blocks will try to send five 50 ms pulses one after the other, to send the different commands to the oscilloscope in an ordered way and allowing time to the oscilloscope to process them. It’s important to note that the period time of the clock has to be bigger than two times the total time of the five “TON” blocks together, in order to allow time for all the pulses to be sent and the timers to be reset. In this case: five times 50ms equals 250 ms, and two times 250 ms equals 500 ms, smaller than 750 ms, so it’s ok.

The following networks will create the strings for every command to be sent, and send the through the COM1 port as soon as the pulse for their command is received. Every time the pulse is sent, a variable showing this command has already been sent is activated, to increase the channel number in the last network as soon as all the commands are sent, and so try to configure the following one.

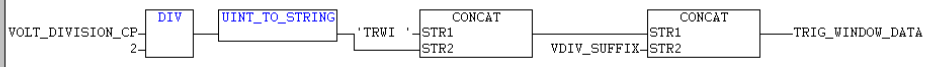
The first command to be sent is the one to switch a channel ON or OFF, depending on the selection in the touch panel. After it, the volts per division are set to the previously calculated value, depending if it’s a current measure, voltage measure in AC, or voltage measure in DC or Impulse. Networks seven to nine send the Offset command, taking into account the polarity if DC or Impulse voltage measurement selected, or setting it to zero if AC voltage measure or Current measurement. The lasts commands to be sent are the coupling and probe attenuation of the channel, set to DC – 1 MOhm and 10k in Voltage measurements, and to DC – 50 Ohm and 1 in Current measurements.

6.4.38. SEND_OSCI_CONFIG (FB)

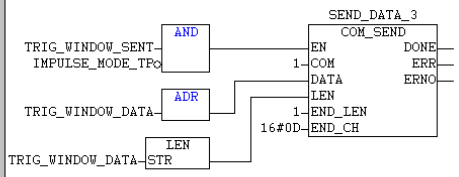


0008

Sets the Triggger Window to half of a voltage division above and below for the Continuous mode.

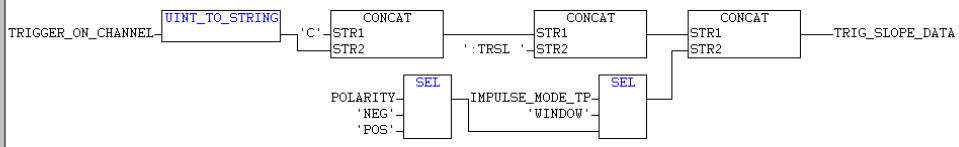


0009

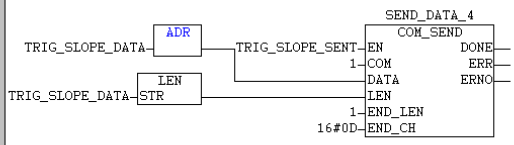


0010

Sets the Trigger Slope to "Positive" or "Negative" depending on the Polarity when Impulse mode; or to "Window" when in Continuous mode

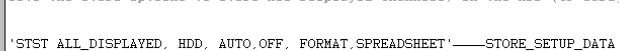


0011

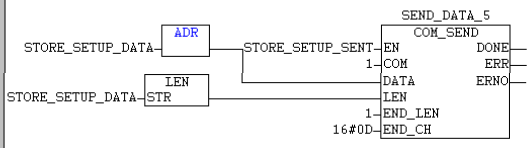


0012

Sets the Store options to store All Displayed Channels, in the HDD (CF card), auto mode OFF, and in SpreadSheet format.



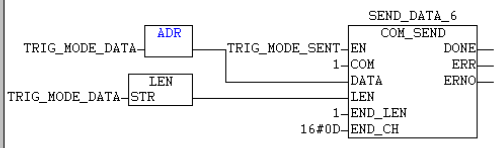
0013



0014

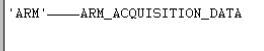
Sets the Trigger Mode to "Single".

0015

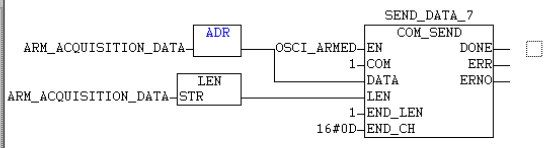


0016

Prepares the command to enable the signal acquisition by changing the acquisition state (trigger mode) from "Stopped" to "Single".

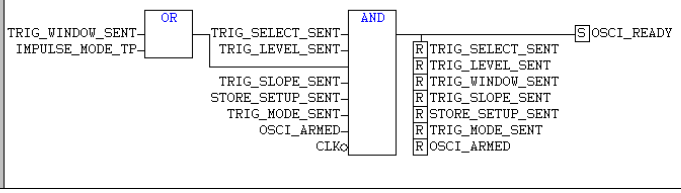


0017



0018

If everything is sent, resets the variables, resets PREPARE_OSCI and sets OSCI_READY TRUE..



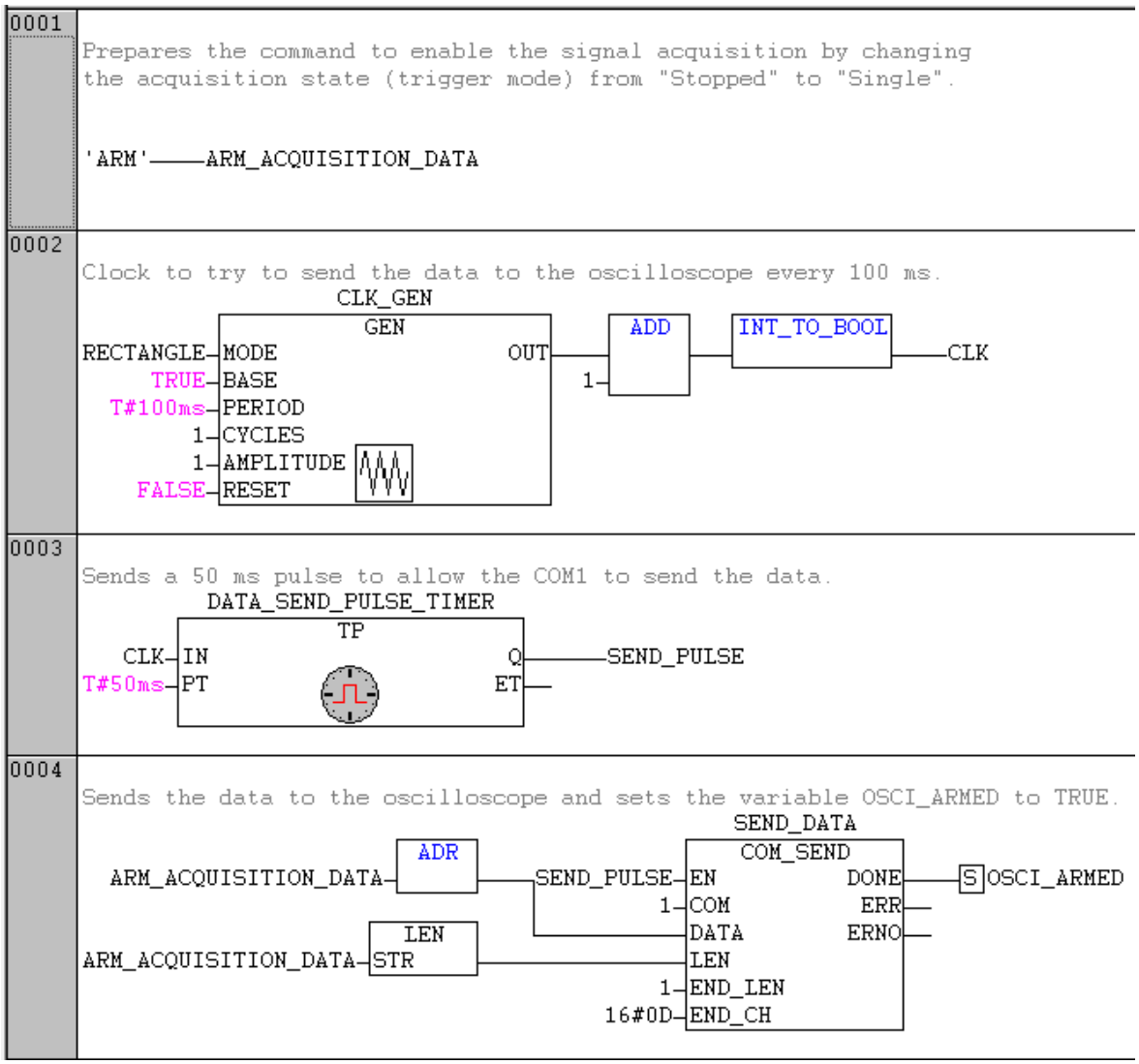
cope, same and a nd in ves a

The first command to be sent in this case is the channel where to trigger; followed by the trigger level, which will be set to zero for the Continuous mode (trigger window is used in the Current Probe), and to a fourth of a voltage division and according to the polarity for the Impulse mode. Networks eight and nine set the trigger window to half of a voltage division of a current measurement, but this command is only sent when in Continuous mode. After it, the following command sets the trigger slope to “positive” or “negative” according to the polarity in Continuous mode, or to “window” in Impulse mode.

Fifth command sets the storing options in order to store all displayed channels in the HDD (Compact Flash Card), with automatic mode OFF, and in Spreadsheet format. Finally, networks fourteen and fifteen set the trigger mode to “single”; while the last command makes sure the oscilloscope is enabled to acquire signals by “arming” it (setting the trigger mode to “single” in case it is in “stopped”).

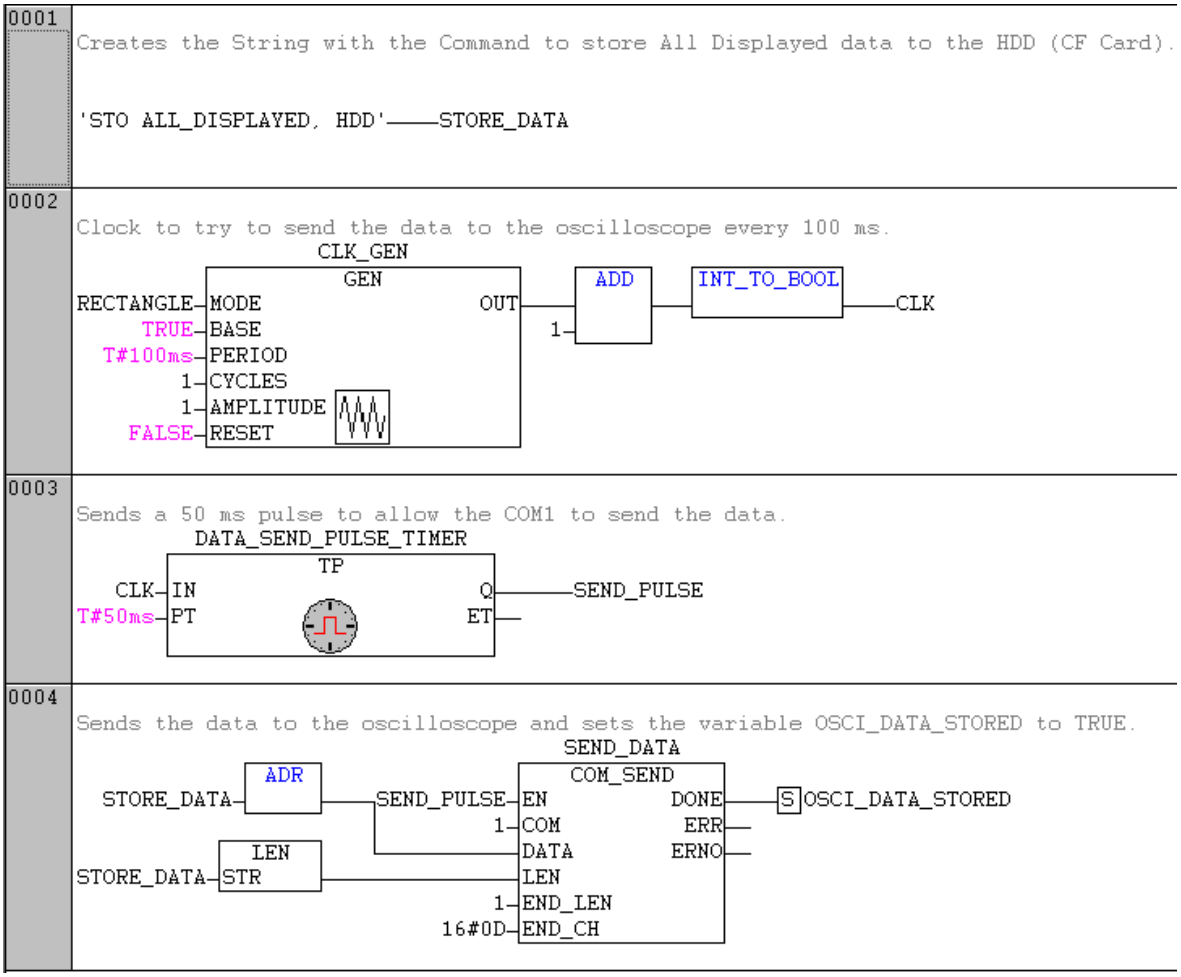
When all the commands are sent, network eighteen resets the variables indicating it and sets OSCI_READY true to communicate the Oscilloscope has been configured.

6.4.39. ARM_OSCI (FB)



ARM_OSCI sends a command to the oscilloscope to change its trigger mode to “Single” in case it was in “Stopped” (typically after it triggered), and so get it ready to acquire data again. It works the same way as the other blocks to send commands to the oscilloscope, with the clock and the timer to send pulses and enable the COM port number one.

6.4.40. STORE_OSCI (FB)



This function block, using a clock and a timer as all the other blocks that send commands to the oscilloscope, sends a command through the COM port 1 in the PLC to make the oscilloscope store all the displayed data in the HDD (CF Card in ML 9901 Lab). As soon as it's done, it sets OSCI_DATA_STORED to true.

7. Miscellaneous

7.1. Spark Gap Control

The disruptive discharge voltages for various spaces between spheres are standardized and given in tables for the standard atmospheric conditions of temperature and pressure (20 ºC and 101,300 Pa), and with an absolute humidity between 5 gm^{-3} and 12 gm^{-3} (averaging 8.5 gm^{-3}). As in the Lab a self-trigger is not desired, but a controlled one through the ignition of the Spark Gap is wanted, slightly bigger distances have to be chosen for the same voltage.

However, changes in temperature, pressure and humidity can shift these values, causing a self trigger or the impossibility to trigger even with the spark. To check how big these effects were in ML9901 Lab, data was collected from a weather station in the lab, and the worst case scenarios for winter and summer were calculated from statistical data. Then the correction factors for the disruptive voltage at a certain distance were calculated according to the following formulas:

The relative air density δ is defined by

$$\delta = \frac{b}{b_0} \times \frac{273+t_0}{273+t}$$

where

the atmospheric pressures b and b_0 are expressed in the same units (kPa);

t and t_0 are the temperatures in degrees Celsius.

The average value of absolute humidity h under which the values in tables 2 and 3 were obtained is 8,5 gm^{-3}. The values in tables 2 and 3 shall be corrected for humidity by multiplying the values in those tables by the humidity correction factor k given by the following equation:

$$k = 1 + (0{,}002 \times (h/\delta - 8{,}5)) \qquad (2)$$

with the ambient absolute humidity h in gm^{-3}.

Fig. 47 Formulas to calculate the correction factors due to atmospheric conditions for the IEC curve of disruptive discharge voltage for various spacings between spheres

After analyzing the results, the effect of humidity was found very irrelevant, but temperature and pressure made a bit more of difference. An average correction factor of **0.94** was found for the usual conditions in the lab, shifting between **0.92** and **0.95** as extreme values. This is equivalent to multiplying the distance between spheres by **1.06** in order to have a discharge at the voltage stated in standardized conditions.

The worst extreme case scenarios from statistical data for summer and winter were found to give correction factors of **0.88** and **0.99**, equivalents to multiplying the distance between spheres by **1.01** and **1.14**.

All these values show that, for the weather conditions in the Lab, a self trigger should be expected to happen some times before desired if the values for the standardized conditions are used. It also shows a variation of around **+/- 5%** around the average correction factor of **0.94** due to changes in weather conditions between summer and winter.

The graph in **Fig. 48** shows the IEC curve of disruptive discharge voltage for various spacings between spheres, together with the values measured experimentally in the Lab, from 10 to 140 kV. The experimental values were measured twice, with the circuit in positive and in negative polarity, and with two deviders every time (the DC Divider and another divider from the company *NorthStar*). That's why there are five curves in the graph.

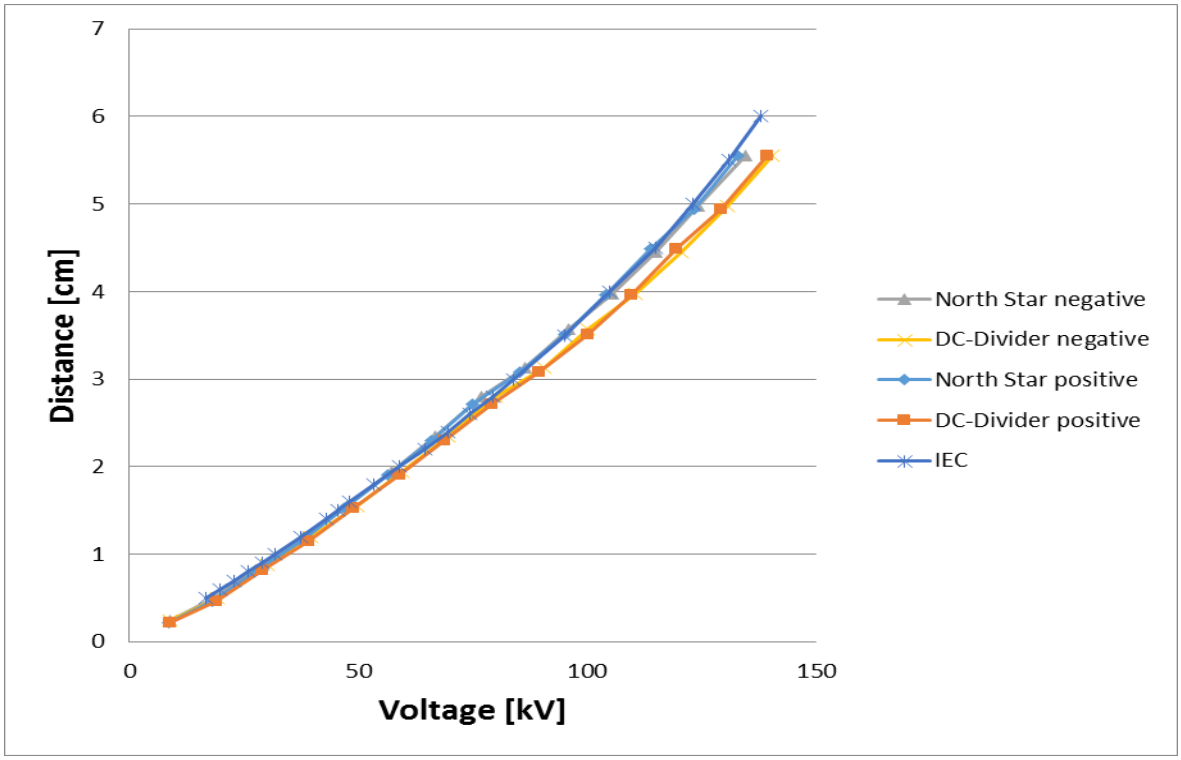


Fig. 48 Graph showing the IEC curve of disruptive discharge voltage for various spacings between spheres, and the curves actually measured in the Lab with two polarities and two different dividers

7.2. Variable speed fans

In order to reduce the heat inside of the Control Rack where the Safety Control Unit, PLC and AC Source are, a panel composed by 3 fans has been installed.

The fans used are the model *MultiFan 4214* by *PAPST*. Their nominal values are 4.3 W and 24 V DC, but they can work between 12V and 28 V, changing the rotation speed. Each fan drains a current up to 670 mA at 28 V.

In order to be able to switch them on and off independently from the rest of the rack and to regulate its speed, a basic voltage regulation circuit has been attached to them. This circuit includes a switch to start or stop them, a red LED showing when the switch is in the on position, and a potentiometer connected with an *LM317* to regulate the voltage applied to them and control their spinning speed. All this is powered by an AC/DC adapter unit that, connected to one of the 230 V plugs inside the tower, delivers 29.5 V of direct current to the fans control circuit.

Fig. 49 Image of the fans installed and working in the Control Rack

Fig. 50 Back part of the fans, with detail of the control circuit

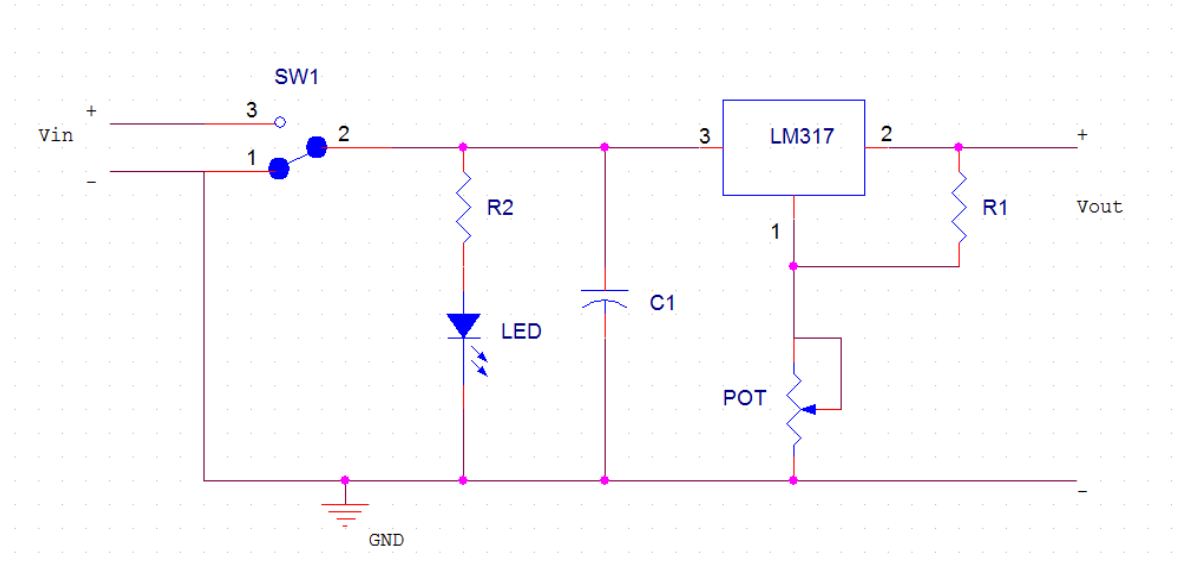


Fig. 51 Schematics of the control circuit of the fans for the Control Rack

7.3. Measurement PCBs

In order to pre-process the data from the dividers, the encoder and the current probe and bring it to the PLC, some little electronic circuits are attached to it. Following, their schematics and main functionality are described.

7.3.1. AC Divider PCB

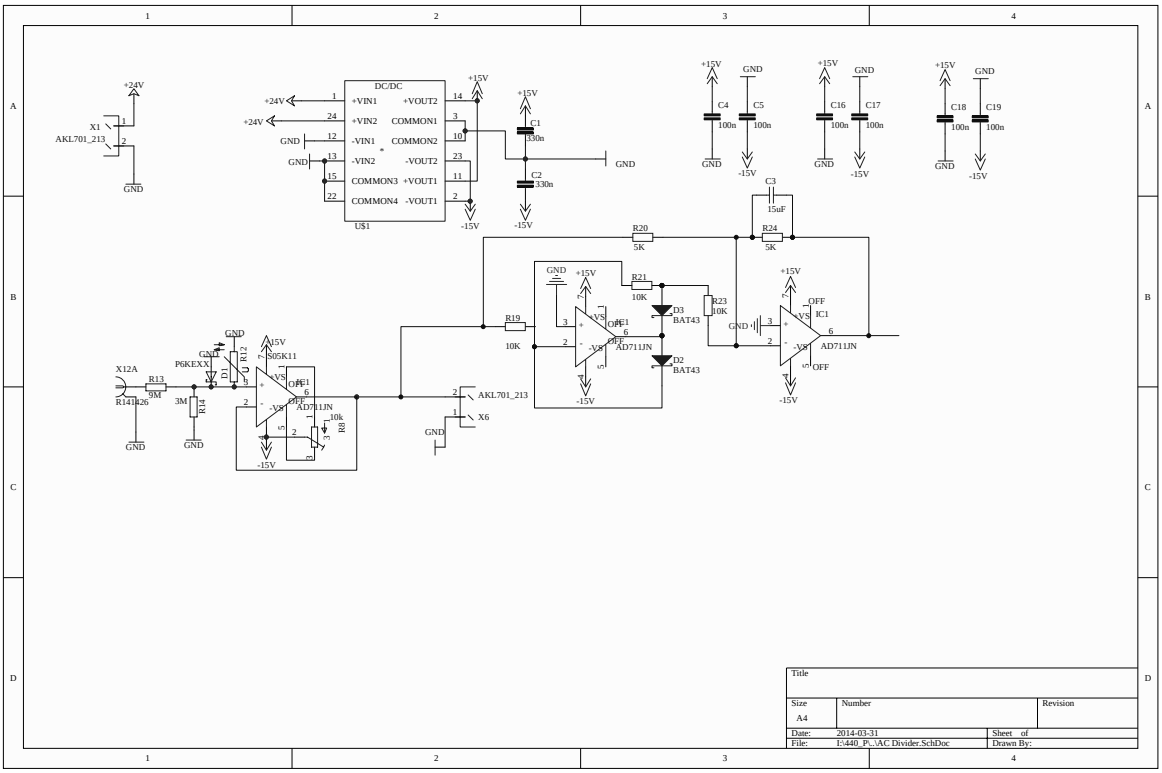


Fig. 52 Schematics for the PCB to process the signals from the AC Divider

It basically consists in a succession of Operation Amplifiers that, apart from decoupling the HV Setup from the PLC, rectify the AC voltage coming from the divider and give out its *rms* value, so the PLC can handle it.

There is also another output connected only through one amplifier, to jump the original signal directly to an output where the oscilloscope can be connected.

It has also some diodes to protect the PLC and the amplifiers from possible voltage peaks.

7.3.2. DC Divider PCB

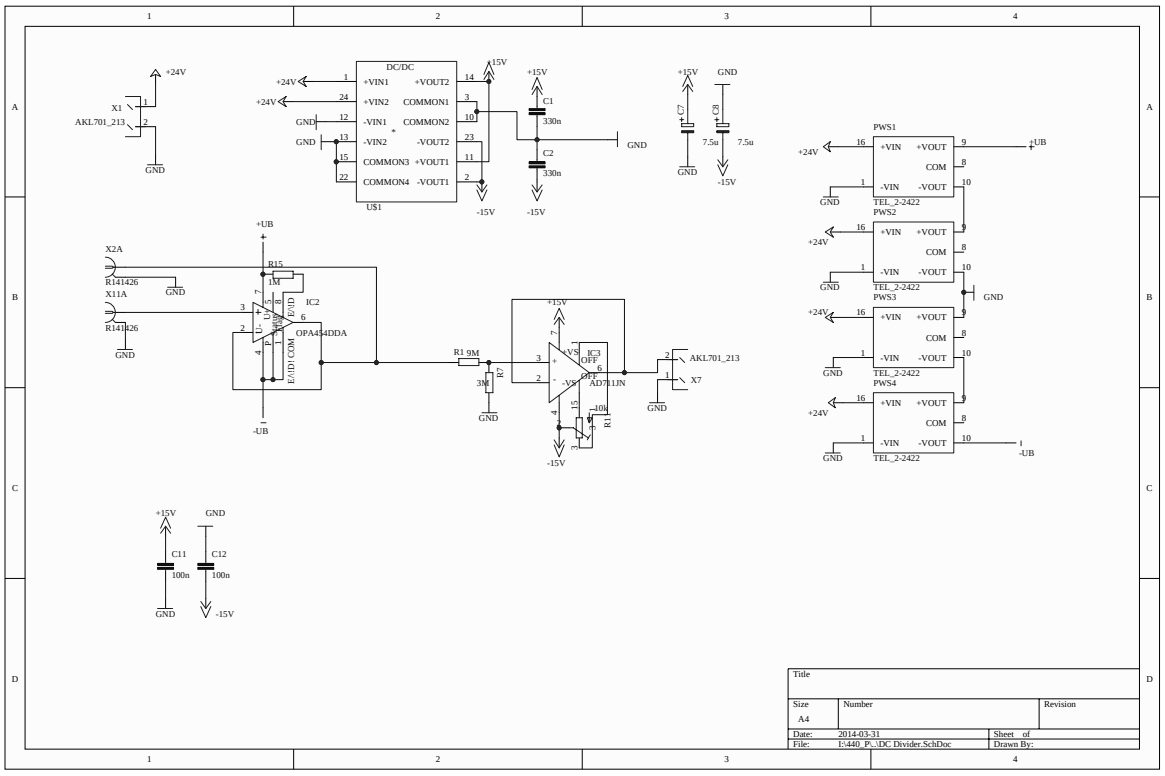


Fig. 53 Schematics for the PCB to process the signals from the DC Divider

It consists also in a succession of Operational Amplifiers to decouple the HV Setup from the PLC and filter the signal to a value the PLC can handle.

In order to reduce the 24V from the PLC voltage supply to the 12V or 15V the Operational Amplifiers need, a bunch of DC/DC converters are also used.

7.3.3. Current Probe PCB

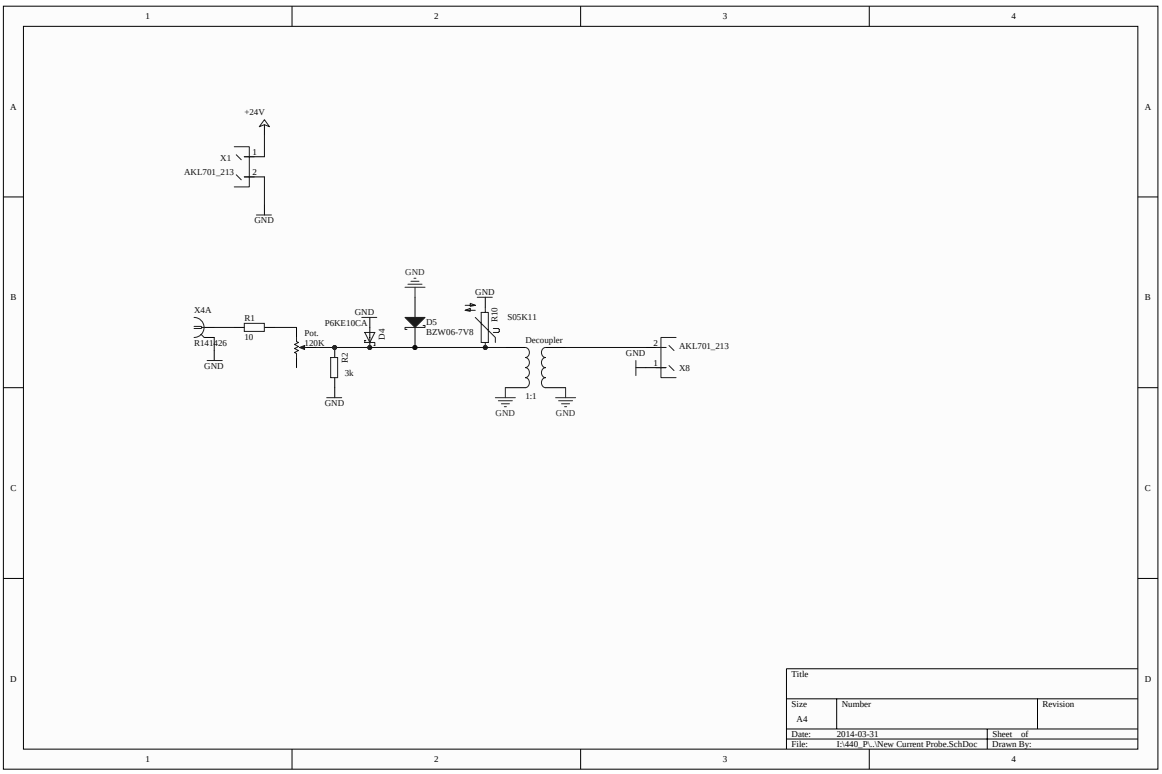


Fig. 54 Schematics for the PCB to process the signals from the Current Probe

It is basically compounded by a succession of protective diodes and varistors that limit the voltage to a value lower than 10V, the maximum the PLC can handle in its inputs. It also features a 1:1 transformer to decouple the HV setup from the PLC.

In the beginning of the circuit there is also a voltage divider made with a resistor and a potentiometer, in order to regulate the voltage the circuit delivers and so cover all the range of possible breakdowns in the High Voltage Setup.

7.3.4. Encoder PCB

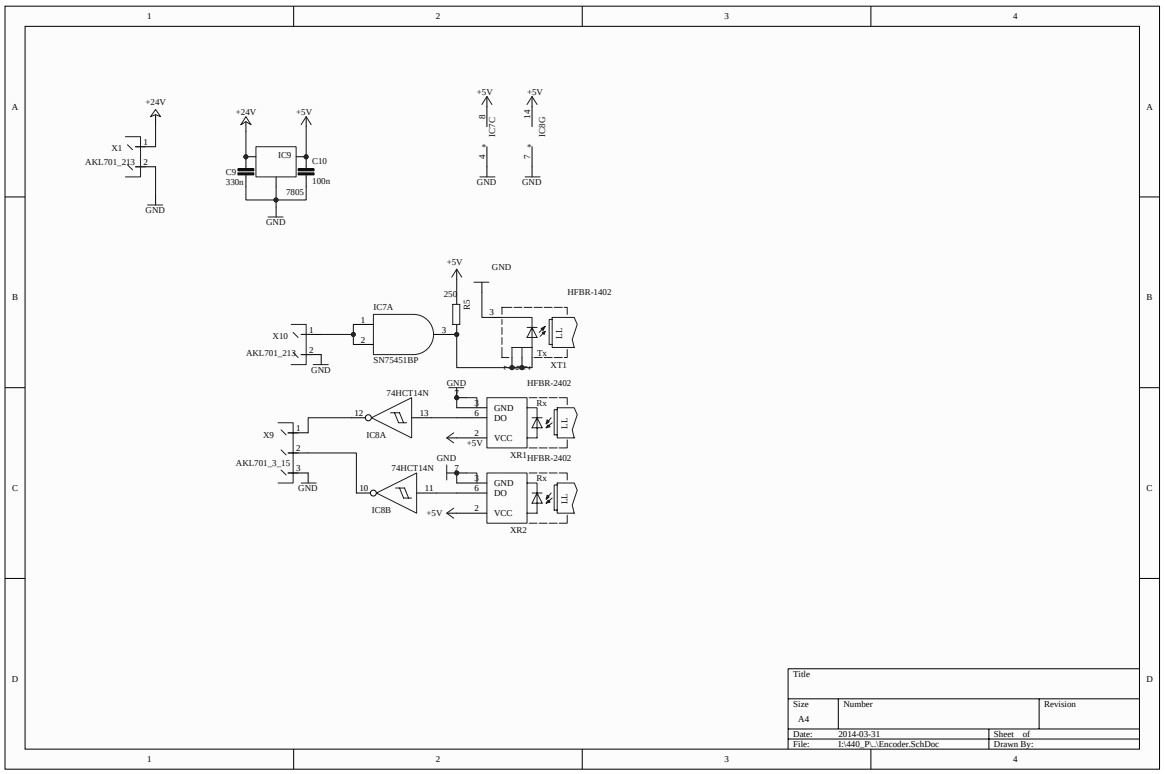


Fig. 55 Schematics for the PCB to process the signals from the Encoder

One of the features of this PCB are two Fiber Optic receivers that get the light signals from the encoder next to the Spark Gap motor and convert them to digital signals for the inputs of the PLC. Before getting into the Controller, these signals are also filtered through a *Schmitt Trigger* comparator in order to reduce their noise.

Moreover, in this PCB there is also the Fiber Optic Transmitter that takes the digital signal from the PLC to Trigger the Spark Gap and converts it to a pulse of light that will be sent through a fiber optic cable and actually trigger the device.

8. Environmental Impact

As this project was mainly focused in the control system of a Lab, its direct environmental impact is quite insignificant, left only to the electricity consumption of the devices used, and the disposed garbage from the renovations in the Lab. So that, the environmental impact will be studied for the whole Lab and research group where the student was, in these two fields:

- Vectors of environmental impact from the construction and use of ML9901 Lab.
- Comparison between the use of SF_6 and other gases in Gas Circuit Breakers.

8.1. Environmental Impact of ML9901 Lab

Main environmental impact of ML9901 Lab comes from its building process, the energy spent during its operation, and the waste generated in it. Analyzing the different environmental vectors it can affect we find:

- **Water**: There are no hazardous liquids that could pollute the surrounding waters being used in the Lab, so there is no impact on the water.
- **Air, flora and fauna**: Some experiments can produce very loud explosions and noise, disturbing so the fauna in the surroundings. However, ML9901 and most of the other big labs are built underground, so the effects of noise are mitigated. Apart from that, while carrying out experiments with electricity there is always the possibility to ignite a fire or burning part of the setup, and so emitting smoke and gases to the air, polluting it and maybe affecting the living plants and animals around.
- **Landscape**: Even if the Lab is underground, it's in a big building with many facilities, including a heating system with a tall chimney, and needs roads and transport utilities for the people to go to work there. All these has a visual impact in an otherwise quite rural area.
- **Humans**: The main harm from humans come from the possibility of suffering an accident when using high voltage or carrying experiments in the Lab.

- **Land**: There is not a big impact on land for the use of the Lab. Nevertheless, land was affected when the Research Center was built, as moreover it has many underground Labs.

8.2. SF_6 versus other gases in Gas Circuit Breakers

SF_6 has always been used in Gas Circuit Breakers due to its high dielectric strength and arc-quenching capability, which make it the perfect gas to blow the arc and cut the current flow in high voltage breakers. However, SF_6 has been recognized as one of the most potent greenhouse gases, with a Global Warming Potential (GWP) of more than 22,000 times the one of the CO_2 in a 100 years horizon. Moreover, it has a very long lifetime, of more than 3,000 years. So that, Kyoto Protocols are encouraging to reduce its usage all over the world.

There are no many gases that can substitute SF_6 in the Circuit Breakers, as apart from the exposed capabilities, they need to be no flammables and should stay gaseous at low temperatures. This narrows the options to N_2 and its derivates (like air) and Carbon Dioxide (CO_2). This last ones seems to be the most promising one, having not so good capabilities as SF_6 but really better ones than air, and even when being a greenhouse gas too, it is far less strong than SF_6. So that, in the Gas Circuit Breakers group of the ABB Corporate Research Center, investigation is being done in new CO_2 Circuit Breakers, together with improving the efficiency and emissions of the current SF_6 ones.

Recently, there have been some projects to develop a Circuit Breaker with a Global Warming Potential of zero, using vacuum as the interrupting technology and dry air for insulation between the terminals. However, it is still a very new development and at the moment only for mid voltage of around 70 kV.

9. Budget

The main costs during the project came from buying a new CPU for the PLC and the Touch Panel. Moreover, other materials and pieces had to be bought too for the upgrade and maintenance of the Lab, amounting around 5000 CHF more. Regarding the software, it did not carry any cost, as *ABB* already had licenses for the programs needed during the project, like *CoDeSys*, *MatLab* or *Microsoft Office*. Adding this to the salary for intern and master student during six months, it gives a budget of **27,315 CHF**, which at today's exchange rate equals to **22,430 €**, as detailed in **Table 15**.

Table 15 Costs of the project

Concept	Unitary Cost	Quantity	Cost (CHF)
Intern Salary	3400 CHF/month	4 months	13600
Master Thesis Salary	1600 CHF/month	2 months	3200
PLC CPU ABB PM591-ETH	4315 CHF/unit	1 unit	4315
Touch Panel ABB CP435T-ETH	1200 CHF/unit	1 unit	1200
Other materials for the Lab	5000 CHF	-	5000
		Total costs	**27315 CHF (22430 €)**

Conclusions

As this is not a purely research project, there are no special conclusions taken out of it. There is just the work done and left in the Lab to accomplish the objective of endowing it with the capacity to run some automated test sequences. So, the main conclusion of the project is the fact that *ABB*'s Corporate Research Center Laboratory ML9901 can now be programmed from a touch panel in order to run some automated test sequences.

The other important goal of the project was to understand the functionality of the Lab and the relations between all of its components, and write it down to make it clear for future users. This is what has been done in the first part of this thesis, where there is an overview of all the components in the Lab and what they are used for.

After having acquired all this knowledge about the Lab, it was possible to design a new communication system that would implement the ability to run the PLC from the Touch Panel and to remotely control the Oscilloscope to store data from the experiments automatically.

The occasion was also used to change almost all the physical connections in the control system, building new ones structured in a more sensed way and with ports and cables in order to have the possibility to physically disconnect the different modules. All this new communication system was accurately documented too, in order to be understandable in the future.

Once the new communication system was built, it was time to program the PLC and the Touch Panel in order to keep the manual functionalities and implement the possibility to run automated test sequences. This process proved to be quite tedious, as to be correctly debugged it required a lot of testing in the real setup.

Finally, a neat design for the menus in the Touch Screen was developed too, so the user could program the automated test sequences without many difficulties.

At the end, the automated Continuous Sequence was tried quite many times, working flawlessly under different circumstances. However, it has to be remarked that due to a lack of time and some mechanical problems in the Spark Gap, the automated Impulse Sequence was not tried so many times as the previous one, leaving the possibility of some bugs appearing in the future under unexpected circumstances. Nevertheless, the accurate documentation of the code running the PLC should make it possible to identify eventual errors and correct them, as well as adding more functionalities or changing the behavior of the already implemented ones.

All in all, *ABB*'s Corporate Research Center ML9901 High Voltage Lab can now be controlled either manually through the old Console or run some test sequences automatically if programmed through the Touch Panel. Moreover, there is a document written in English giving an overview of what the Lab does and how it works; presenting all its components and connection systems; and explaining in detail how to operate it and how the control system is programmed, so that it can be further updated in the future.

Acknowledgements

I would like to thank many people for their support during the realization of this project. First of all I am very thankful to ABB for the opportunity to work in its Corporate Research Center during six months and carry on this thesis.

Especially I would like to thank my supervisor, Dr. Philipp Simka, for his constant support, patience and willingness to teach me about high voltage, a world in which I had very few knowledge when I started in ABB. I am also very thankful to the supervisor of the thesis in Barcelona, Dr. Ricard Bosch Tous, who introduced me to the basics of high voltage before going to Switzerland, and then guided me for the elaboration of this document from the distance.

I also want to thank all the people in ABB who helped in some moments of the project, either with their knowledge, with their physical effort when renovating the Lab, or providing components we needed for the new control system. Among them, I want to give a very kind acknowledgement to the people in the ABB PLC support team in Germany, for their fast and useful answers every time I had a question about the PLC or the Touch Panel; and especially to Torsten Ruland, who even called me to help on solving a configuration problem with the new CPU.

I am very grateful to everybody in the Gas Circuit Breakers Group, for make me feel comfortable among them and for the great working environment, together with all the informal conversations and fun in coffee brakes, lunches and the group trip to Ticino. And specifically to the group of interns there: Loukas Theodoulou, Smitha Karambar, Kate Davies, Georgia Tsagaropoulou and Oliver Mathews, not only for the help while working there, but also for their good friendship and the amazing trips and things we did together in the weekends and spare time.

I want to take some time to acknowledge all the other people from all over the world that I met in Switzerland, and helped on making my stay in Zurich one of the best experiences in my life. A warm regard especially to the group of Catalans, with whom I shared fantastic moments and made me feel like I had a family away from home.

Finally, I thank my parents Jordi and Teresa and my brother Oriol, for their constant support and patience without what this project wouldn’t have been possible, and for all the visits to bring me food and stuff while I was abroad.

Bibliography

Bibliographic references

[1] 3S – SMART SOFTWARE SOLUTIONS GMBH; *User Manual for PLC Programming with CoDeSys 2.3*. Kempten (Germany), 2010.

[2] ABB; *CP400Soft Technical Manual.* Heidelberg (Germany), 2008.

[3] ABB; *Control Panel CP400 series. Connection with AC500 / KT9x. Connection with ACS-drives. First steps*. Heidelberg (Germany): 2007.

[4] ABB; *AC500 Ethernet Configuration: Basic Configuration*. Heidelberg (Germany): 2014.

Complimentary references

CORTES CHERTA, Manuel; BOSCH TOUS, Ricard; CASALS TORRENTS, Pau; *Aparamenta Elèctrica. Recull d'apunts.* Serveis Gràfics Copisteria Imatge, SL, Barcelona: 2013.

ABB; *ABB Switchgear Manual, 11th edition*. Cornelsen Verlag Scriptor GmbH & Co, KG, Berlin (Germany): 2008.

ABB; *CP435 Installation and Operation Manual*. Heidelberg (Germany): 2008.

PACIFIC SOURCE, INC.; *ASX Series. AC Power Source. Operation Manual*. Irvine, CA (USA): 2011.

LECROY; *Remote Control Manual: LeCroy Digital Oscilloscopes*. Chestnut Ridge, NY (USA), 1996.

LECROY; *Waverunner™: Operator's Manual*. Chestnut Ridge, NY (USA), 2002.

Annexes

Annex A: Electric Elements in ML9901 Lab

The following table shows all the elements in the lab with their nominal values, the measured ones, deviation between measured and nominal values, and their serial numbers. The color of the label column shows the color of the physical element in the lab.

Table 16 Inventory of Electric Elements from the HV Setup in ML9901 Lab

Label / Color	Nominal Value	Measured Value	Deviation (%)	Serial No.
Resistors				
RL 1	10 MΩ	9.183 MΩ	8.17	R - 03369
~~**RL 2**~~	10 MΩ			Not there anymore
RL 3	10 MΩ	8.583 MΩ	14.17	R - 11876
RD 1 LI	350 Ω	336.70 Ω	3.80	R - 03370
RD 2 LI	350 Ω	339.60 Ω	2.97	R - 02329
RD 3 LI	350 Ω	351.80 Ω	-0.51	R - 11878
RE 1 LI	2.4 kΩ	2'419 Ω	-0.79	R - 03373
RE 2 LI	2.4 kΩ	2'335 Ω	2.71	R - 02333
RE 3 LI	2.4 kΩ	2'425 Ω	-1.04	R - 11877
RD 1 SI	43 kΩ	42.330 kΩ	1.56	R - 11837
RD 2 SI	43 kΩ	43.220 kΩ	-0.51	R - 11836
RD 3 SI	43 kΩ	40.300 kΩ	6.28	R - 13988
RE 1 SI	98 kΩ	95.920 kΩ	2.12	R - 11838
RE 2 SI	98 kΩ	97.690 kΩ	0.32	R - 11839
RE 3 SI	98 kΩ	96.670 kΩ	1.36	R - 13989

RE 1 PT	133 Ω	138.88 Ω	-4.42	R - 13985
RE 2 PT	133 Ω	130.91 Ω	1.57	R - 13986
RE 3 PT	133 Ω	132.78 Ω	0.17	R - 13987
RD 1 PT	598 Ω	611.00 Ω	-2.17	R - 13982
RD 2 PT	598 Ω	609.20 Ω	-1.87	R - 13983
RD 3 PT	598 Ω	607.10 Ω	-1.52	R - 13984
RM	280 MΩ	280.552722 MΩ	-0.20	R - 02325
RM 2	280 MΩ			R - 131019
RM 3	280 MΩ			R - 131020
RM 4	280 MΩ			R - 131021
RM LV	28 kΩ	28.060 kΩ	-0.21	R - 02325

Capacitors				
CB 1	1200 pF	1207.90 pF	-0.66	C - 03380
CB 2	1200 pF	1181.00 pF	1.58	V - 02178
CB 2 LV	507100 pF	472500.00 pF	6.82	V - 02179
CB 3	1'200 pF	1161.70 pF	3.19	V - 11435
CB 3 LV	465633 pF	461300.00 pF	0.93	V - 11437
CS 1	25000 pF	26940.00 pF	-7.76	C - 02357
CS 2	25000 pF	26320.00 pF	-5.28	C - 01350
CS 3	25000 pF	24880.00 pF	0.48	C - 11777
CS 4	25000 pF	25540.00 pF	-2.16	C - 13902

CS 5	25000 pF	25380.00 pF	-1.52	C - 13904
CM	100 pF	107.17 pF	-7.17	C - 12811
CM	100 pF		100.00	C - 12783
CM red??	100 pF	109.40 pF	-9.40	---
CM 2	100 pF	113.50 pF	-13.50	C - 13900
CM 3	100 pF	111.40 pF	-11.40	C - 13901
CM LV	884700 pF	924400.00 pF	-4.49	Home made
CM LV	961500 pF	972100.00 pF	-1.10	Home made

Diodes				
D 1				G - 03171
D 2				G - 12433
D 3				G - 12434
D 4				G - 13481
D 5				G - 13482
D 6				G - 13483
D 7				G - 13484
D 8				G - 13485

Spark Gaps				
FS 1				S - 02057
FS 2				S - 03066
FS 3				S - 11157

Annex B: Schematics of the Safety Circuit

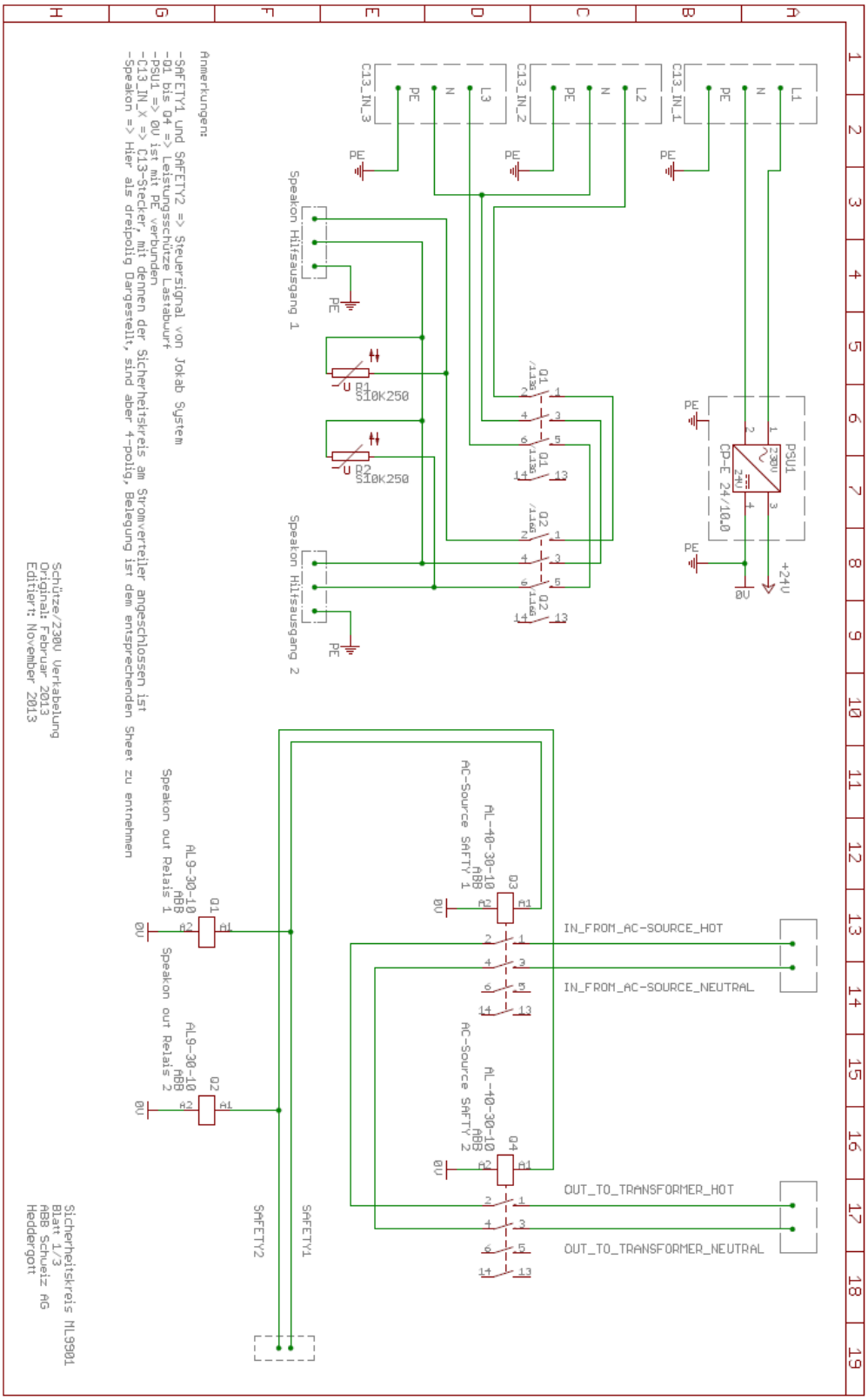


Fig. 56 Schematics of the Safety Circuit (page 1)

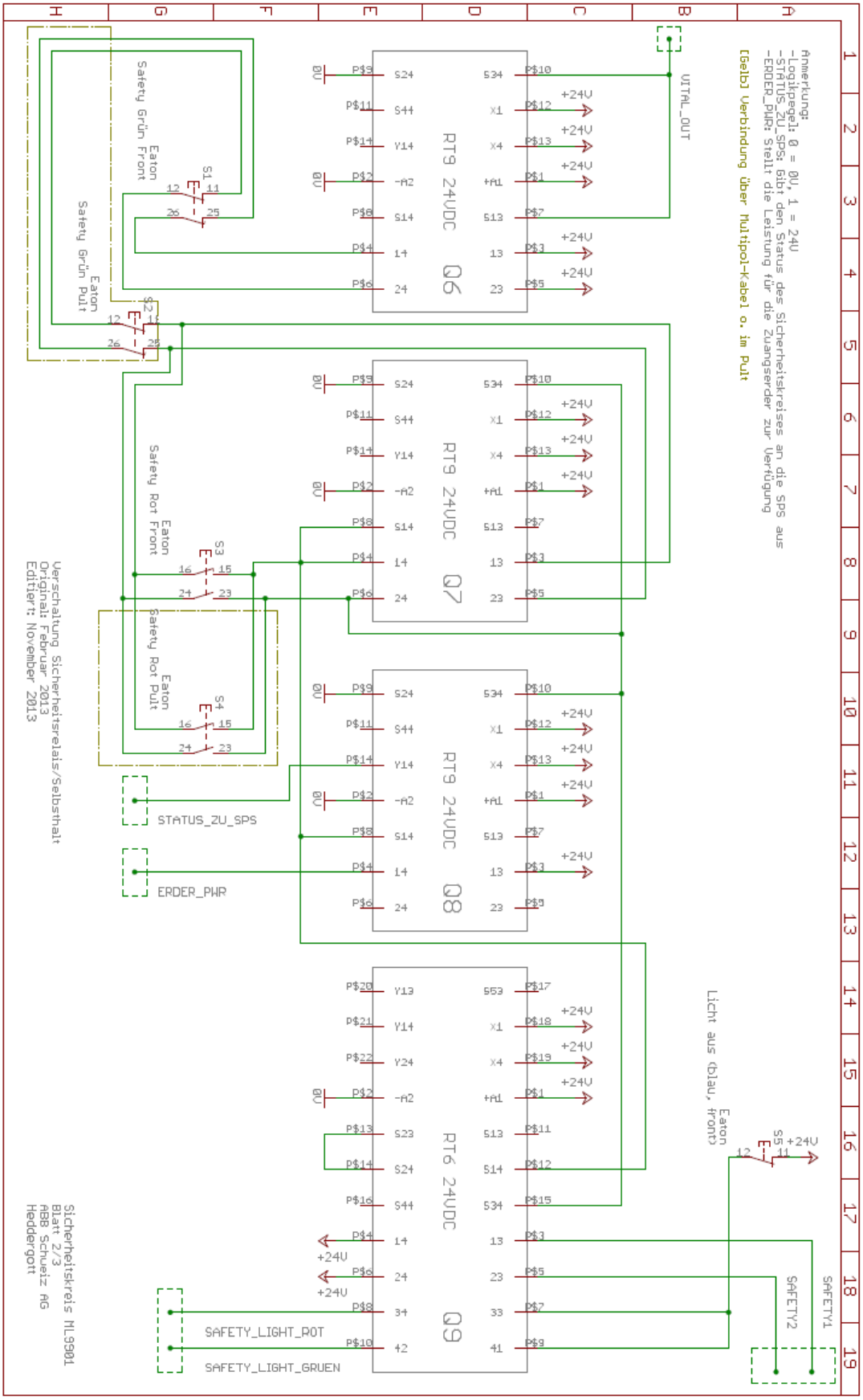


Fig. 57 Schematics of the Safety Circuit (page 2)

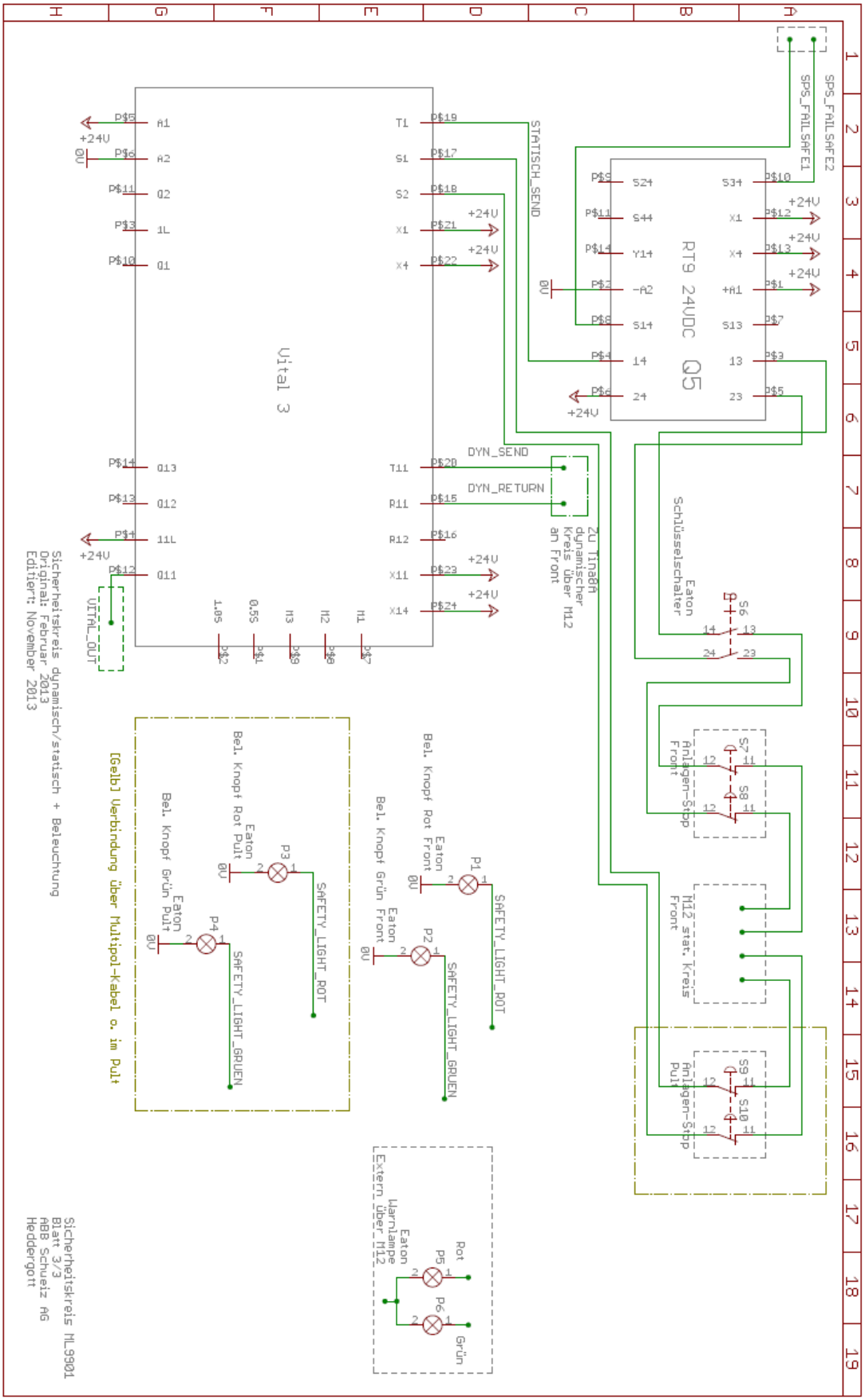


Fig. 58 Schematics of the Safety Circuit (page 3)