UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA

Performance of the TileCal High Voltage Upgrade System

Catarina José Ponte Pereira

Mestrado Integrado em Engenharia Física

Dissertação orientada por:
Prof. Guiomar Gaspar de Andrade Evans
Prof. Agostinho da Silva Gomes

2022

Acknowledgements

First and foremost, thank you to my parents for everything they have done for me, you are the reason I stand here today and I am forever grateful. To my sister, who helped me stay grounded and kept me company when I felt the most alone and to all my family for all the words of encouragement and support, thank you.

Thank you José Ramos for your unconditional love and support, for being always there when I needed it the most; you were my guiding light throughout these years, and I love you.

To my closest group of friends, thank you. You managed to make these challenging years my best, and you were there every step of the way. Even when things were hard, we always stood together. Whatever the future has in store for us, I will always have each and every one of you in my heart.

I want to give a special thank you to Professor Guiomar Evans, my advisor, for her continuous support, patience and empathy that was extremely important and helped me throughout the development of this thesis, not only academically but personally as well. Thank you for always being understanding and kind, I will always cherish all your advice and support.

I want also to thank Professor Agostinho Gomes for being my advisor and for welcoming me onto this project and his laboratory. To Luís Gurriana, Filipe Cuim and Filipe Martins for being always available to answer my questions. A special thank you as well to Rui Marques, who is already included in my friend acknowledgements; but this time for the companionship and help while understanding and performing our parts in this project.

I would also want to acknowledge LIP (Laboratório de Instrumentação e Física de Partículas) and INESC-ID (Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento) for hosting me as their collaborator and FCT (Fundação para a Ciência e Tecnologia) for their scholarship under the contract CERN/FIS-PAR/0026/2021.

Abstract

The work done under the scope of this thesis is part of the HV Remote project, whose main purpose is to create a remote high voltage (HV) regulation and distribution system for ATLAS Tile Calorimeter's (TileCal) photomultiplier tubes (PMTs). This system is made up of a high voltage supply board, the HV Supplies board, coupled to a high voltage regulation and distribution board, the HV Remote board, which will be connected to the PMTs via 100 m cables. Both boards will be controlled by a Field-Programmable Gate Array (FPGA) development board using the Serial Peripheral Interface (SPI) protocol. To supply all of the existing PMTs in TileCal, 256 pairs of HV Remote and HV Supplies boards will be manufactured, with each pair of boards and their respective FPGA board supplying a dedicated HV to 48 PMTs.

The second prototype of the HV Remote board has already been produced and, at the moment, a third prototype is in production. This work will mainly focus on the functional, stability and load tests of the second prototype of the HV Remote, both in the laboratory environment and in the CERN Test Beam.

The cables that will connect the HV Remote to the TileCal PMTs need to be tested in order to characterise their behaviour and influence on the HV transmitted. As such, another important part of this work is the creation of a Cable Test board, which, when used in conjunction with a selection board, the HV Reading board, will allow the user to select a cable, supply it with a specified HV and perform continuity and real-time voltage readings at the terminals of each one of the cables. In addition, it will allow measurements of the voltage of each of the cables' ground and shield signals and supply an HV to be used during the tests generated by a DC/DC converter. The board's schematics and Printed Circuit Board (PCB) were developed using the Altium Designer Tool, version 22.

The Cable Test board control software was also developed, as well as a Graphical User Interface (GUI), which will allow the user to perform quick readings and data acquisition of the different parameters. The aforementioned software was written in the Python 3 language with object-orienting programming, specifically designed to function with a Raspberry Pi development board.

Keywords: TileCal, HV Distribution System, PCB design, Cable Test, SPI

Resumo

O detetor ATLAS é o maior dos quatro detetores do Grande Colisor de Hadrões (*Large Hadron Collider* ou LHC) e já providenciou importantes descobertas, como o Bosão de Higgs. De forma a ir mais além, pretende-se aumentar a luminosidade do LHC em cinco vezes no que será chamado o LHC de alta luminosidade (*High Luminosity* LHC ou HL-LHC). A preparação para este aumento de luminosidade inclui toda uma atualização do sistema do detetor ATLAS, que também prevê alterações ao seu calorímetro hadrónico, o *Tile Calorimeter* (TileCal). O TileCal é constituído por uma matriz de ferro onde se encontram telhas cintilantes de plástico, que emitem luz ao interagirem com as partículas resultantes de uma colisão. Essa luz, predominantemente azul, é depois coletada por um conjunto fibras óticas que reemitem luz verde. Finalmente, tubos fotomultiplicadores (PMTs) detetam a luz emitida pelas fibras óticas e geram um sinal elétrico cuja tensão é proporcional à intensidade da luz coletada. Estes PMTs necessitam de uma alimentação de alta tensão negativa que neste momento é fornecida por um conjunto de 16 unidades independentes, conhecidas como HVPS (*High Voltage Power Supplies*), cuja distribuição e ajuste individual são efetuados pelas placas eletrónicas HVMICRO e HVOPTO. As placas de distribuição estão colocadas em "super-gavetas" no raio externo do TileCal e, como tal, estão sujeitas a níveis de radiação elevados, que irão aumentar por um factor de 5 no HL-LHC. Além disso, as placas foram fabricadas nos anos 90 e muita da eletrónica utilizada encontra-se obsoleta. Todos estes fatores tornam a manutenção e substituição de componentes uma tarefa complicada e, portanto, pretende-se substituir este sistema.

O atual sistema de alta tensão será então substituído por uma solução remota, a HV Remote, apresentada pela equipa portuguesa da experiência ATLAS, na qual o sistema de distribuição e regulação de alta tensão estará localizado numa sala técnica da caverna USA15 do detetor ATLAS e ligado aos PMTs através de cabos de 100 metros. Embora próxima do detetor, a caverna USA15 está bem protegida da radiação, pelo que operações de manutenção de equipamento no seu interior são seguras mesmo quando o LHC está em funcionamento. Como tal, este novo sistema não tem de ter eletrónica resistente à radiação.

O sistema, no seu núcleo, é constituído por uma placa que efetua a distribuição e ajuste individual da alta tensão de cada PMT, a HV Remote, e por uma placa que fornece à HV Remote as altas e baixas tensões primárias, a HV Supplies, conectadas por uma placa de interface. A HV Supplies fornece alta tensão através de conversores DC/DC. Cada placa HV Remote consegue regular e distribuir individualmente a tensão de 48 PMTs. Para alimentar todos os PMTs do TileCal, serão necessários 256 pares de placas HV Supplies e HV Remote. Estes pares serão distribuídos por 16 estruturas (*crates*), cada uma contendo uma placa intermédia que permite a conexão dos pares de placas anteriormente referidos e fornecerá as tensões primárias de 3.3 V @ 3 A, 12 V @ 15 A, -12 V @ 10 A e 24 V @ 20 A e os sinais digitais de controlo. Cada estrutura terá 16 conjuntos de placas HV Remote e HV Supplies e será controlada por uma placa de desenvolvimento FPGA através de protocolo SPI.

O trabalho desenvolvido no contexto desta dissertação tem como um dos seus objetivos o teste e avaliação do comportamento da placa HV Remote em termos funcionais e de estabilidade. Foi testado o segundo protótipo da placa tanto no Laboratório de Instrumentação e Física de Partículas (LIP) como no laboratório de ensaios com feixe da experiência ATLAS no CERN. Este segundo protótipo tinha como requisitos:

- O controlo individual das altas tensões de 48 PMTs, entre -360 V e -1000 V;
- A opção de ligar/desligar cada grupo de quatro PMTs, maualmente ou por controlo digital;
- A leitura da tensão aplicada a cada PMT;
- A leitura de uma tensão de teste de -1.2 V e de dois transdutores de temperatura;
- Uma variação da alta tensão fornecida aos PMTs com um desvio padrão máximo de 0.5 V face ao valor expectável.

Os testes realizados no contexto desta dissertação utilizaram uma placa de desenvolvimento Raspberry Pi para o controlo digital ao invés de uma FPGA. Verificou-se a funcionalidade da aplicação do sistema de controlo digital desenvolvida previamente no contexto deste projeto em conjunto com a HV Remote e, através dela, efetuou-se toda a aquisição de dados referente a todos os testes. Foi efetuada uma calibração para cada um dos canais de alta tensão da placa, fazendo corresponder as contagens dos seus conversores Digital-Analógico (DAC) e Analógico-Digital (ADC) às tensões medidas com um voltímetro externo. Os resultados desta calibração foram utilizados posteriormente como referência nos testes de estabilidade efetuados nos ensaios com feixe no CERN.

Os testes de estabilidade consistiram na monitorização de cada um dos canais de saída de alta tensão da HV Remote através do seu ADC ao longo de diferentes períodos de tempo, nunca inferiores a 5 horas. Foram também realizados testes de resposta a diferentes cargas: sem carga (*float*), passiva (ou resistiva), ativa e PMT. Consistiram na conexão das cargas a cada uma das saídas da HV Remote, que foram monitorizadas através de um osciloscópio. Ainda no contexto do teste da placa HV Remote, foi também efetuado um mapeamento que associa cada um dos canais de saída de alta tensão da placa a um PMT do TileCal. Os testes efetuados e os seus resultados estão descritos no quinto capítulo.

Foi também desenvolvido um sistema de teste para os cabos de 100 metros, a placa Cable Test, cujos esquemáticos e placa de circuito impresso (PCB) foram desenhados com a ferramenta Altium Designer, versão 22. O sistema de teste, descrito no terceiro capítulo, consiste essencialmente no fornecimento de uma alta tensão conhecida aos diferentes cabos e em medições da tensão nas suas extremidades. De forma a diminuir a complexidade e o custo de produção do sistema, a placa Cable Test foi desenvolvida para funcionar em conjunto com a placa HV Reading, uma placa composta por matrizes de relés, que servirá para automatizar os testes à HV Remote, cujo primeiro protótipo foi desenvolvido no âmbito de outra tese [1]. Um segundo protótipo está a ser desenvolvido por outro estudante [2]. Ambas as placas foram desenvolvidas para comunicar com uma placa Raspberry Pi e partilham os mesmos sinais SPI.

Ambos os protótipos da HV Reading serão responsáveis pela distribuição da alta tensão e seleção dos cabos, através de duas matrizes de 48 relés cada, que serão utilizadas pela placa Cable Test para selecionar entre, no máximo, 96 sinais (alta tensão e de massa) provenientes dos 48 cabos. Para além disso, a HV Reading fornecerá as tensões primárias da placa Cable Test, $+24$ V e $+3.3$ V, e os sinais SPI.

A placa Cable Test é responsável pelo fornecimento e monitorização de uma alta tensão negativa e regulável, proveniente de um conversor DC/DC com uma tensão máxima de saída de -1500 V. A placa

tem dois modos de funcionamento: modo de teste laboratorial e modo de funcionamento no laboratório de ensaios com feixe do CERN.

No modo de teste laboratorial, a alta tensão negativa estará conectada diretamente à placa HV Reading que irá proceder à sua distribuição pelos cabos, conectados a ambas as placas. A Cable Test, por sua vez, monitoriza as tensões às extremidades dos cabos através de um conjunto de divisores de tensão, de multiplexadores (MUXs) analógicos e de um ADC. Os divisores de tensão estão conectados diretamente à entrada dos cabos na placa e permitem alternar, com o uso de pequenos condutores, *jumpers*, entre diferentes tipos de carga (sem carga, passiva, ativa e PMT) a testar.

No laboratório da experiência ATLAS no CERN, os cabos alimentarão diretamente os PMT, pelo que não será possível efetuar medições nas suas extremidades. Como tal, a alta tensão negativa estará conectada a um divisor de tensão existente na placa e cuja saída estará conectada à placa HV Reading que novamente distribui a alta tensão pelos cabos e, por consequência, terá apenas como carga os PMTs. Nesta configuração só é possível determinar se existe ou não continuidade no cabo. É possível alternar facilmente entre os dois modos de funcionamento manualmente através da conexão/desconexão dos pontos de ligação dedicados recorrendo a *jumpers*.

Foi também desenvolvido um sistema de controlo digital e aquisição de dados na linguagem de programação Python 3 com um estilo de programação orientada a objetos. Uma interface gráfica de utilizador (GUI) foi desenvolvida através da ferramenta Qt Designer, que permite a conversão de elementos visuais em código Python. A utilização conjunta destes elementos permite ligar e desligar cada uma das fontes de alimentação, fornecer automaticamente alta tensão a um cabo de cada vez, regular a alta tensão fornecida e efetuar leituras de tensão para cada um dos cabos remotamente. A descrição do código desenvolvido está contida no quarto capítulo deste trabalho.

Palavras-Chave: TileCal, Sistema de Distribuição de Alta Tensão, Desenho de PCBs, Teste de Cabos, SPI

Table of Contents

List of Tables

List of Figures

Acronyms

AC Alternating Current

ADC Analog-to-Digital Converter

ALICE A Large Ion Collider Experiment

ATLAS A Toroidal LHC ApparatuS

BOM Bill of Materials

BT Barrel Torroid

CAN Controller Area Network

CERN European Organization for Nuclear Research

CMS Compact Muon Solenoid

CPHA Clock Phase

CPOL Clock Polarity

CS Chip Select

CSM Central Solenoid Magnet

CT Cable Test

CTI Comparative Tracking Index

DAC Digital-to-Analog Converter

DC Direct Current

DCS Detector Control System

DRC Design Rules Check

EB Extended Barrel

ECT End-Cap Torroid

EEPROM Electrically-Erasable Programmable Read-Only Memory

ERC Electrical Rules Check

FET Field-Effect Transistor

FPGA Field Programmable Gate Array

GND Ground

GPIO General Purpose Input/Output

GUI Graphical User Interface

HL-LHC High-Luminosity Large Hadron Collider

HV High Voltage

HVPS High Voltage Power Supplies

I/O Input/Output

IC Integrated Circuit

ICON Input/Output Configuration

IODIR Input/Output Direction

LAr Liquid Argon

LB Long Barrel

LED Light-Emitting Diode

LHC Large Hadron Collider

LHCb Large Hadron Collider beauty

LIP Laboratório de Instrumenção e Física de Partículas

LSB Least Significant Bit

LV Low Voltage

MISO Master Input Slave Output

MOSFET Metal–Oxide–Semiconductor Field-Effect Transistor

MOSI Master Output Slave Input

MUX Multiplexer

OOP Object-Oriented Programming

PCB Printed Circuit Board

PMT Photomultiplier Tube

POR Power-On Reset

RC Resistor-Capacitor

SCLK Serial Clock

SCT Semiconductor Tracker

SMD Surface Mount Device

SPDT Single-Pole Double-Throw

SPI Serial Peripheral Interface

SPST Single-Pole Single-Throw

TileCal Tile Calorimeter

TRT Transition Radiation Tracker

UI User Interface

USB Universal Serial Bus

UV Ultra-Violet

VME Versa Module Eurocards

WLS Wavelength Shifting

Chapter 1

Introduction

The Large Hadron Collider (LHC) is a 27-kilometre ring particle accelerator that is part of the European Organization for Nuclear Research (CERN) accelerator complex. It is the largest and most powerful particle accelerator in the world, able to collide protons with a centre of mass energy of up to 14 TeV and design luminosity of $10^{34}\text{cm}^{-2}\text{s}^{-1}$. [3]

Collisions between particles are controlled to occur along the LHC in specific locations corresponding to the four main experiments: A Toroidal LHC ApparatuS (ATLAS), Compact Muon Solenoid (CMS), A Large Ion Collider Experiment (ALICE) and Large Hadron Collider beauty (LHCb). [4]

Right now, the ATLAS experiment is undergoing an upgrade program that includes changes to its hadronic barrel calorimeter, known as Tile Calorimeter (TileCal).

TileCal is a cylindrical-shaped sampling calorimeter with a tile structure that uses steel as an absorber and scintillators as an active medium. Particles resulting from collisions interact with the scintillating tiles, producing light with a spectrum peaking in the blue that is then collected by wavelength-shifting fibres that convert it into green light. These fibres are coupled to Photomultiplier Tubes (PMTs) which convert the light into an analogue electrical signal that is amplified, shaped, digitized by sampling it every 25 ns and stored until a decision is received from the trigger system to either store the event for further analysis or to discard it. [5]

The PMTs need a HV supply system capable of regulating the individual high voltages to be applied at each PMT. The Portuguese ATLAS team is developing an HV remote supply and regulation system that will replace the current HV regulation system under the TileCal upgrade program.

The new system will consist of 256 pairs of boards, each pair composed of one HV distribution board, **HV Remote**, and one power supply board, **HV Supplies**. Each set of 16 pairs of boards will be controlled by one Field Programmable Gate Array (FPGA) development board through SPI communication, while communication with ATLAS Detector Control System (DCS) will be made through an Ethernet protocol. The system will be connected to the TileCal PMTs via two hundred fifty-six 100 m long HV cables, each cable containing up to 48 pairs of wires.

The HV Remote board is able to individually regulate and distribute the HV of 48 PMTs. It has the option to enable/disable each 4-PMT group and to perform readings of the voltage applied to each PMT, a test voltage of -1.2 V and two temperature transducers placed in different parts of the HV Remote board.

This work will focus on testing the second prototype of the HV Remote board, setting up a test system to evaluate the response of PMTs to a light source and working on the integration of the HV system in the CERN test systems to evaluate the performance of the detector's response when powered by the new system. Another main focus of this work is the creation of a Printed Circuit Board (PCB)

tester for the cables connecting the HV Remote to the PMTs, the Cable Test board. It will work in conjunction with a selection board, the HV Reading board, and will allow the supply of a specific test HV with which continuity tests and voltage readings at the terminals of each one of the cables will be performed. Cables' Ground (GND) and shield wires will also be evaluated.

Digital control software for this board was also developed, as well as a Graphical User Interface (GUI), which will allow the user to perform quick readings and data acquisition of the different parameters. The aforementioned software was written in the Python 3 language with object-orienting programming, specifically designed to function with a Raspberry Pi 3 Model B+ development board.

1.1 The ATLAS Detector

The ATLAS detector at CERN is the largest running detector at the LHC and seeks to observe high-energy phenomena which cover a wide range of physics, including the measurement of the Higgs boson properties and the search for extra dimensions and particles that can make up dark matter. [6]

The ATLAS detector mainly consists of an **inner detector**, a set of **calorimeters** (electromagnetic and hadronic), a **magnet system** and a **muon spectrometer**. A view of the overall detector layout is shown in Figure 1.1.

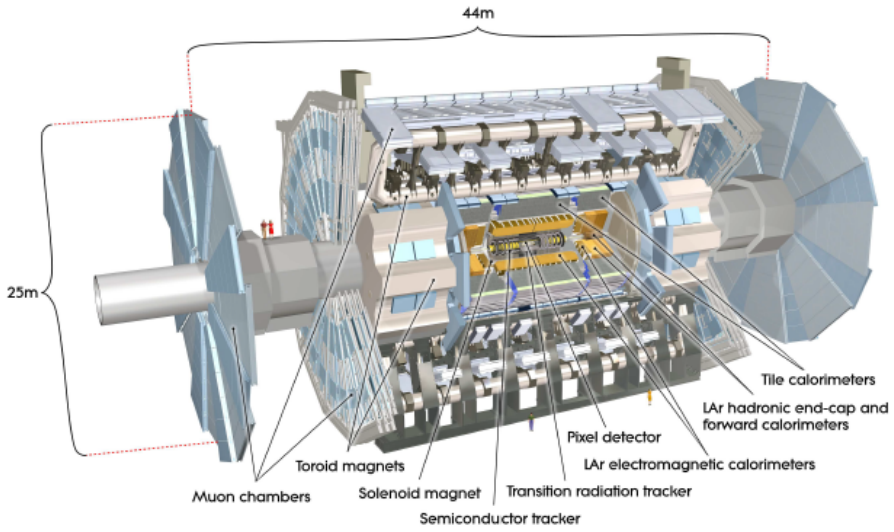


Figure 1.1: Sectional view of the ATLAS detector. [7]

Inner Detector

The inner detector consists of three main systems of detectors: Pixel Detector, Semiconductor Tracker (SCT) and Transition Radiation Tracker (TRT). These are responsible for measuring the direction, momentum and charge of electrically charged particles produced in each collision. [8]

Calorimeters

The primary function of a calorimeter is to measure the energy of charged and neutral particles, by stopping or "absorbing" the particles resulting from a collision, forcing them to deposit all of their energy within the detector. They can stop most known particles except muons and neutrinos, which

rarely interact with matter and, therefore, lose much less energy per unit path in matter than hadrons[1] and electrons. Calorimeters can also provide a "missing transverse energy" estimate corresponding to all particles that crossed them without depositing energy which is then compared to data obtained with different detectors such as the muon spectrometer.

The ATLAS calorimetry system consists of an electromagnetic calorimeter, a Liquid Argon (LAr) Calorimeter, and a hadronic calorimeter, the TileCal, located in the central barrel region. Electromagnetic calorimeters such as the LAr calorimeter measure the energy of electrons and photons as they interact with matter. On the other hand, hadronic calorimeters sample the energy of hadrons as they interact with atomic nuclei.

Since TileCal is the subject of this work, a more in-depth description of the calorimeter is featured in subsection 1.2.

The Magnet System

The magnet system is divided in three main sections: **Central Solenoid Magnet (CSM)**, **Barrel Torroid (BT)** and **End-Cap Torroids (ECTs)**. The overall dimensions of this system are 26 m in length and 20 m in diameter.

The system configuration is centred on the CSM, which is an inner 4.5 cm thick superconducting solenoid surrounding the inner detector that is aligned with the beam axis. It has a central magnetic field of 2 T and is responsible for bending charged particles for momentum measurement in the inner detector.

Outside the calorimeters, there are three large superconducting air-core toroids that consist of eight coils assembled radially around the beam axis. These are the BT and the two ECTs which are inserted in it at each end and lined up with the CSM. They generate a magnetic field of 4 T for the muon spectrometer.

Muon Spectrometer

Charged particles, mostly muons, usually escape the inner detector and the calorimeters. The muon spectrometer is located in the outer radii of the ATLAS detector and is responsible for detecting them and measuring their momenta. It is made up of 4000 individual muon chambers. [9]

1.2 Tile Calorimeter

Tile Calorimeter (TileCal) covers the most central region of the ATLAS Experiment and it is designed to detect hadrons and to measure accurately jets[2], tau-particles and missing transverse momentum. It is divided into three cylindrical sections along the beam direction: a fixed central Long Barrel (LB) and two movable Extended Barrels (EBs) located on each side of the LB. These sections can be seen in Figure 1.2.

Each section is subdivided azimuthally into 64 wedge modules constructed of alternating layers of steel plates and plastic scintillator tiles, placed perpendicularly to the colliding beams, as illustrated in Figure 1.3. Steel works as a passive absorber of energy and the scintillator tiles as an active material that radiates light when exposed to a charged particle.

[1] Particles that contain quarks, such as protons and neutrons

[2] In this context, the word jet is used to refer to a concentrated cone of particles that result from the interactions of quarks and gluons.

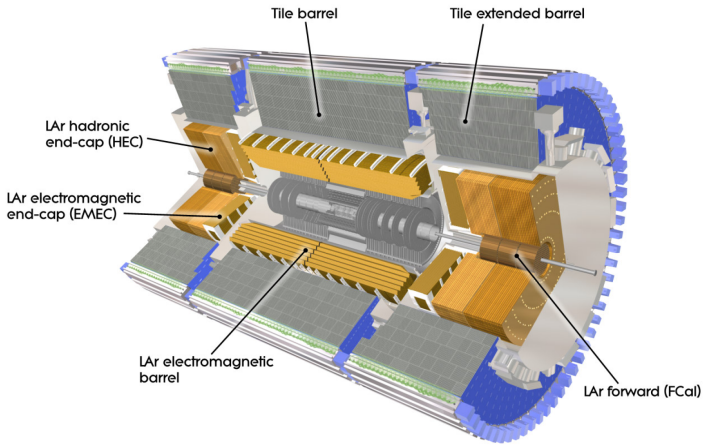


Figure 1.2: Sectional view of the calorimetric system. [7]

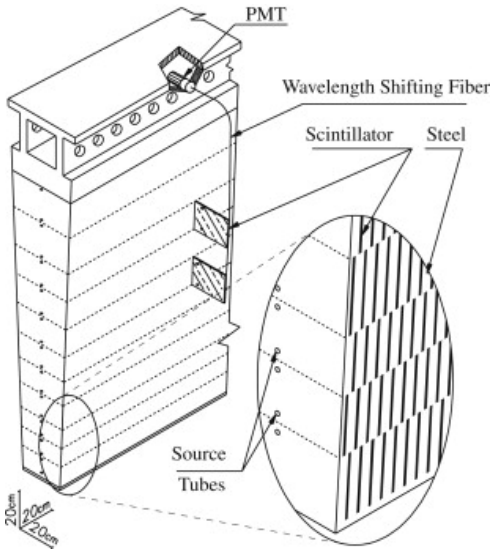


Figure 1.3: Schematic illustrating the mechanical assembly and the optical readout components of one wedge module of the Tile Calorimeter. [10]

Particles resulting from collisions interact with the tiles as described above, producing Ultra-Violet (UV) and blue light that is then collected by Wavelength Shifting (WLS) fibres on both sides of each module which subsequently emits light whose wavelength is shifted to green.

Readout cells are defined by grouping bundles of these fibres coming from adjacent tiles. Each cell is coupled to two PMTs which convert the visible light into an analogue electrical signal. The PMTs are located in the outer radius of the wedges in 2.60 m long drawers that also contain the front-end electronics.

In total, 9852 PMTs are used to readout the TileCal cells and each of their signals are amplified, shaped and digitized at a 40 MHz sample rate. The samples are stored until a decision is received from the ATLAS trigger system which selects approximately 1000 events from the 1.7 billion collisions that occur each second at the centre of the detector. [11]

1.3 TileCal's Present HV Distribution System

Each Photomultiplier Tube (PMT) has different characteristics and requires its own dedicated high voltage supply to allow for individual regulation, in order to work properly and stably. The HV supply system currently used consists of two parts: a front-end system that regulates and distributes the different voltages and an external HV power supply located in the ATLAS technical cavern USA15. [12]

The front-end system is located in super-drawers inside the girders at the outer radius of TileCal's wedges. Each super-drawer contains a controller card, **HVMICRO** and two distributor cards, **HVOPTO**. The boards are connected to each other by a short flexible bus and also two long **HVbus** boards that distribute the regulated HVs to the PMTs, according to the configuration illustrated in Figure 1.4. The connection between the HVMICRO and HVOPTO cards is ensured by a Versa Module Eurocards (VME) bus.

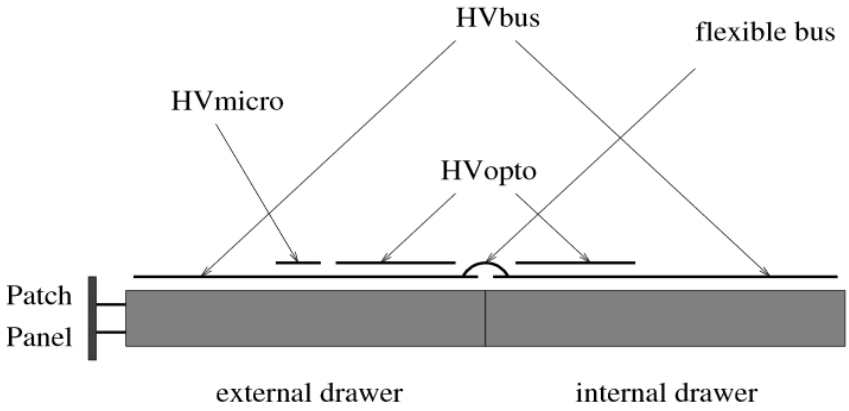


Figure 1.4: Location of the current HV distribution system cards in a super-drawer. [13]

Figure 1.5: Picture of one of the drawers containing the front-end electronics.

The HV power supply system consists of 16 independent units known as High Voltage Power Supplies (HVPS), each one with 16 output channels that provide HV input (HV_{in}) values of -830 V or -950 V to each super-drawer.

HVMICRO

This controller card features the Motorola MC68376 microcontroller as its main component, working at a frequency of 20 MHz. Software written in C programming language allows this card to manage the two HVOPTO cards, to read the HV applied to each PMT, to apply an order to each HV channel, to receive and transmit data to the ATLAS DCS via a Controller Area Network (CAN) bus and to

communicate with a console via an RS232 interface [3]. [13]

This board is also responsible for monitoring with a 0.1°C accuracy seven temperatures at the boards which are essentially temperature transducers integrated on the HVMICRO, HVOPTO and HVbus boards. A picture of the board is shown in Figure 1.6.

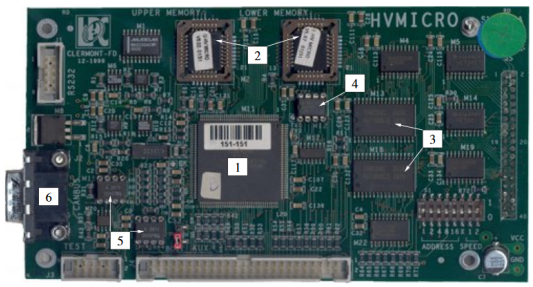


Figure 1.6: Picture of the HVMICRO board. (1) Motorola MC68376 micro-controller; (2) 256 kB flash memories; (3) 256 kB Random Access Memory (RAM); (4) 2 kB EEPROM; (5) optocouplers; (6) connector for the CANbus cable. [12]

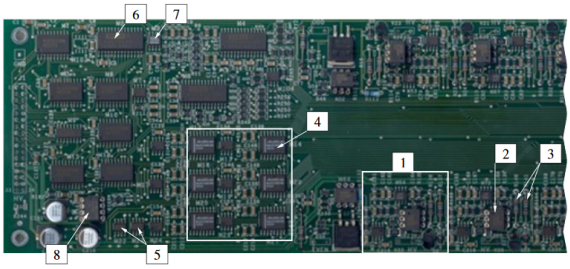


Figure 1.7: Partial picture of the HVOPTO board. (1) Regulation loop; (2) optocoupler; (3) two 100 MΩ HV resistors; (4) six 4 channel-DACs; (5) two -5 V regulators; (6) ADC; (7) the AD589 voltage reference; (8) the 2 kB EEPROM. [12]

HVOPTO

The HVOPTO distribution card (Figure 1.7) is able to switch on/off the 12 PMT groups of even or odd channels and to regulate up to 24 PMTs with a HV output (HV_{out}) that can vary between $HV_{in} - 360$ V and HV_{in}. Since there are 48 PMTs in each super-drawer, two boards are necessary for each one.

The board contains 24 regulation loops that apply the output voltages (HV_{out}). It also contains six 12 bit Digital-to-Analog Converters (DACs) with 4 channels that convert the HV digital value sent by the HVMICRO board, a 2 kB Electrically-Erasable Programmable Read-Only Memory (EEPROM), that stores the initial board conditions, and a 12 bit Analog-to-Digital Converter (ADC) that converts the analogue input signal from an analogue multiplexer with 32 input channels into a digital signal. Two digital control signals control the enable/disable switches of the two PMT groups' channels (even or odd). The 24 HV_{out}, the values of the temperature transducers, the Low Voltages (LVs) of the DACs and a fixed negative voltage are measured for test purposes.

1.4 TileCal Upgrade for the High-Luminosity LHC

Luminosity is the main indicator of the performance of an accelerator and the LHC has, so far, exceeded expectations and surpassed the luminosity targets set out, with a maximum luminosity of $10^{34}\text{cm}^{-2}\,\text{s}^{-1}$.

This high performance has provided a large amount of data to each running experiment and its analysis has resulted in interesting discoveries such as the Higgs Boson. As such, CERN's scientists intend to increase LHC's instantaneous luminosity by a factor of five to ultimately obtain a ten-fold increase in the dataset. The High-Luminosity Large Hadron Collider (HL-LHC) is expected to be operational around 2028.

In order to meet these luminosity requirements, ATLAS is undergoing an upgrade program with 3 stages: phase 0, I and II. Regarding TileCal, in phases 0 and I, there were mostly improvements to allow

[3] Standard for serial communication of data, introduced in the 1960s.

it to contribute to the muon trigger. Right now, the upgrade program is entering phase II and it includes improvements to TileCal's HV supply and regulation system.

The electronics of the HV system currently used in the experiment are dated to the beginning of the ATLAS experiment operation and some of its components are no longer manufactured. Aside from that, since all the electronics of the HV system are located in TileCal's modules, they are exposed to large amounts of radiation that can damage some of its parts. In order to carry out maintenance work on the system, it is necessary to have a long experiment shut down to lower the radiation levels.

These factors make maintenance work very difficult. Therefore, the Portuguese ATLAS team is developing a remote solution, **HV Remote**, in which the high voltage distribution and regulation system will be in the ATLAS technical cavern USA15 and connected to the PMTs via 100 m long cables. An in-depth description of the proposed HV Remote system is featured in chapter 2.

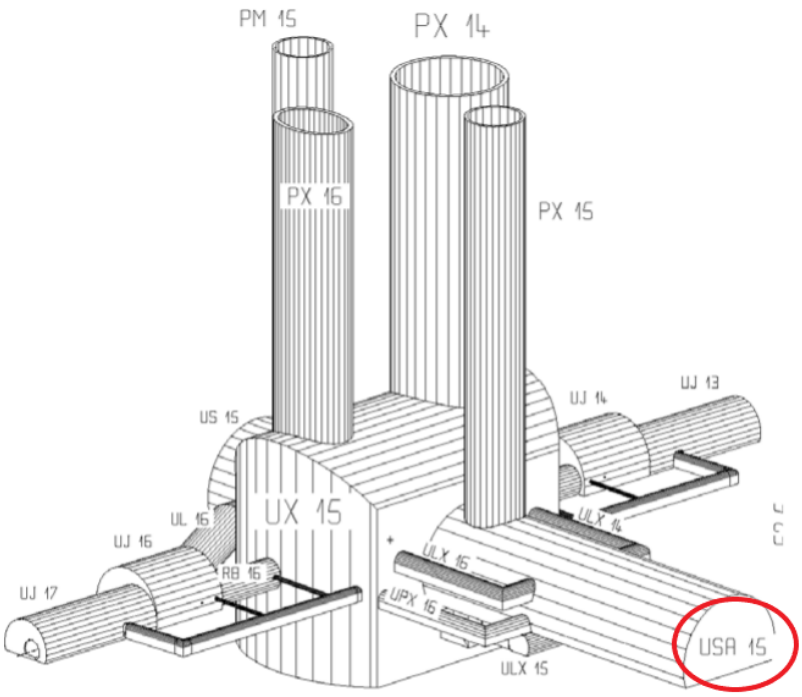


Figure 1.8: Depiction of ATLAS technical caverns and location of the USA15 cavern in respect to the detector cavern (UX15). [14]

The USA15 cavern was designed so that the radiation levels inside are low enough for people to be able to access it even when the LHC is running, despite being next to the experimental ATLAS cavern, as shown in figure 1.8.

This new system and configuration will facilitate maintenance work and decrease the amount of radiation to which the electronic circuits are exposed, so there will be no need for radiation-hard electronics. However, the use of 100 m long cables may subject the transmitted voltages to extra noise. Therefore, the cables must be tested beforehand to ensure the system is stable and reliable. A substantial part of this thesis work was dedicated to testing the continuity and response of these cables, through the development of a test board and software, whose development is described in chapters 3 and 4, respectively.

Chapter 2

Description of the HV Remote System

The new PMT HV supply, regulation and distribution system consists of 16 crates, each crate containing 16 pairs of boards HV Remote and HV Supplies, a FPGA development control board, its interface bus board and a midplane board (Figure 2.1) that will distribute the crate's primary power supplies (24 V @20 A, 12 V @15 A, −12 V @10 A and 3.3 V @3 A) and the digital control signals.

The complete system will be connected to the ATLAS DCS over Ethernet and the communication between the boards and the FPGA will be established via SPI protocol, described in section 2.1.2. Its architecture is shown in Figure 2.2.

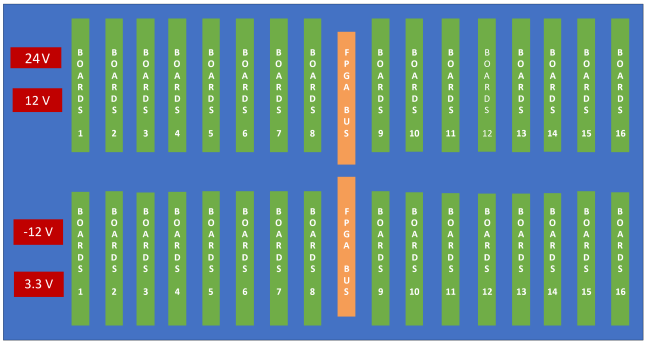


Figure 2.1: Midplane connectors distribution.

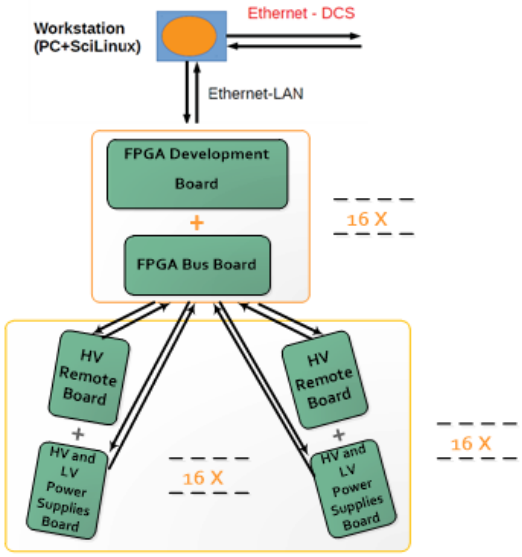


Figure 2.2: HV distribution system control architecture.

Each HV Remote board will be responsible for enabling, distributing and measuring the necessary HVs to 48 PMTs. The HV Supplies board will provide and monitor the primary HV and LV power supplies needed. 256 pairs of HV Remote and HV Supplies boards will be produced to power all of the existing PMTs at TileCal. The HV Remote System will be connected to each TileCal's electronics mini drawer via 100 m long cables.

2.1 HV Remote Board

The HV Remote board is the main board of the HV distribution system being developed and its second prototype (figure 2.3) is the main subject of this work. It was developed according to the results of tests carried out on the first prototype. Its primary function is to distribute, monitor and regulate the HV of 48 channel outputs. It can also monitor a reference test voltage and two temperature transducers, essential to determine the state of the board and of the crate's cooling system.

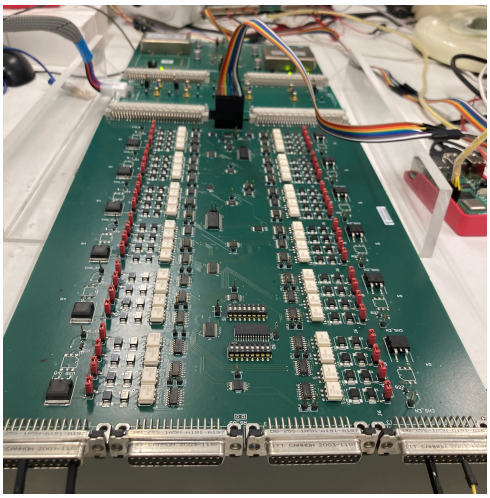


Figure 2.3: Picture of HV Remote's second and current prototype.

The first HV Remote board prototype had as main objectives:

- Individual control of the high voltages for 48 PMTs (between -360 V and -1000 V);
- Option to turn on/off each PMT individually;
- Reading of the voltages applied to each PMT;
- Reading of a test reference voltage of -1.2 V to test its stability;
- Reading of two temperature transducers in different parts of the HV Remote;
- Revision of individual HV control loops to mitigate noise; the HV variation to the expected value should not exceed 0.5 V (rms).

The first prototype was able to perform the individual regulation and reading of the high voltage of each channel and was stable, with a typical HV output variation of 0.1 V. However, one of the issues discovered was that the use of remote on/off switches for each channel limited the maximum HV output value to approximately -900 V, due to current constraints. Since the board must be able to deliver a higher HV output (in absolute value), changes had to be made to this configuration. Considering this issue and PCB design optimisation, it was decided that the second prototype would have the 48 output channels arranged in groups of 4, with the ability to remotely switch on/off each group.

On the other hand, hardware switches for each channel were introduced in the second prototype to ensure all PMTs can be disabled individually. The ability to turn off channels at any given moment is important to prevent malfunctioning PMTs from jeopardizing the remaining ones during LHC runs.

Furthermore, to reduce noise on the board, the board's signal planes were readjusted and a single ground plane was established on the second prototype's PCB rather than the previous analogue, digital and HV ground planes. A 16-bit hardware address and interlock fault circuitry were also introduced. The second prototype's features and overall description follows in the next sections.

2.1.1 Operating Conditions

The HV Remote board needs two HV power supplies (HV1 & HV2) able to provide up to -950 V @10 mA each and three LV power supplies of $+12$ V, -12 V and $+3.3$ V in order to operate properly. These HVs and LVs will be provided by a dedicated HV Supplies board that will be discussed in subsection 2.2 and the recommended values for each power supply board are detailed in table 2.1.

		Min	Nom	Max	Unit
Supply Voltage	HV1 & HV2	-360	—	-950	V
	VN12	-10.8	-12	-15.8	V
	VP12	10	12	15	V
	3V3	3.1	3.3	4.3	v
Supply Current	HV1 & HV2	—	0.01	0.01	A
	VN12	0	1.5	2	A
	VP12	0	0.9	1.1	A
	3V3	0	0.9	1	A

Table 2.1: HV Remote's voltage and current recommended values for each power supply board. [15]

2.1.2 Digital Control System

The Digital Control System of the HV Remote is responsible for distributing and sending all the signals needed to control the board. Each crate will have its own digital control system, which will be implemented on a chip architecture in a FPGA development board. As of now, instead of a FPGA, a Raspberry Pi development board is being used, with an adapted software. [16] Communication between the crate digital control system and the HV Remote board is established via SPI protocol.

SPI Protocol

The Serial Peripheral Interface (SPI) is an interface protocol used to establish connections between micro-controllers and peripheral integrated circuits such as ADCs and DACs. It is a synchronous interface composed of one master and one or multiple slaves, where both the master and one of the slaves can send data at the same time. [17]

SPI protocol has 4 signals: Master Output Slave Input (MOSI), Master Input Slave Output (MISO), Serial Clock (SCLK) and Chip Select (CS). MOSI and MISO are data transmission lines: MOSI transmits data from the master to the slave, whereas MISO transmits data from the slave to the master. Figure 2.4 shows the SPI connection between a master and multiple independent slaves, which illustrates the HV Remote crate communication architecture, where the HV Remote boards are independent slave devices and the FPGA development board is the master device.

The master device starts communication by enabling the CS signal which is typically an active low signal and thus needs a logic low signal to be sent in order to enable it. Afterwards, the master generates the SCLK signal that synchronises the data transmitted and received.

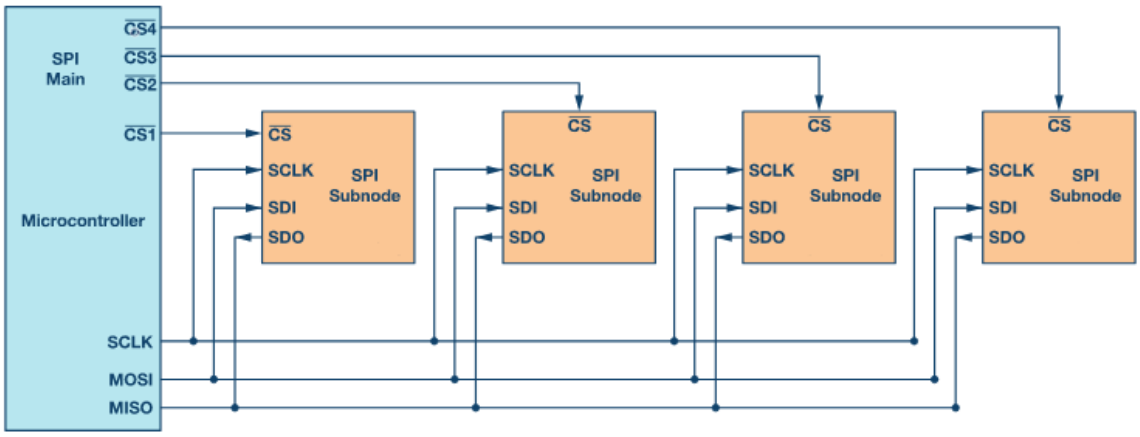


Figure 2.4: SPI configuration between one master device (in blue) and multiple independent slave devices (in orange). [17]

The CS signal has 4 modes which allow the user to choose whether to sample and/or shift data on the rising or falling edge of SCLK. These modes can be chosen by sending a 2-bit digital word, where the most significant bit represents the value of Clock Polarity (CPOL) and the least significant the value of Clock Phase (CPHA) [16]. Table 2.2 and figure 2.5 describe each mode.

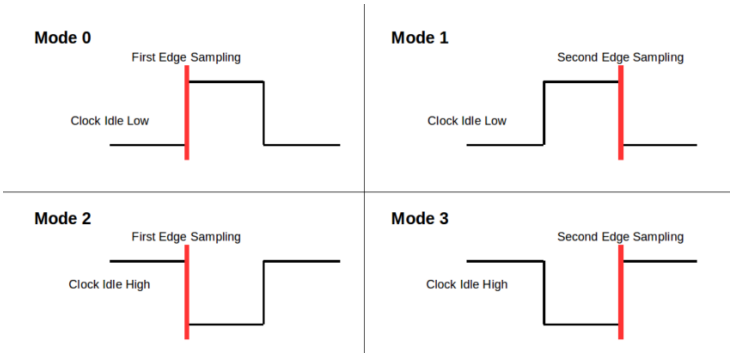


Figure 2.5: Data sampling for each SPI mode. [18]

SPI MODE	CPOL	CPHA	Digital Word
0	0	0	00
1	0	1	01
2	1	0	10
3	1	1	11

Table 2.2: Description of the four different SPI modes.

In modes 0 and 1, CPOL is equal to 0 and, therefore, the clock polarity in the idle state is a logic low. These two modes differ by the CPHA, which determines if the data is sampled on the rising edge and shifted out on the falling edge (CPHA= 0) or if the data is sampled on the falling edge and shifted out on the rising edge of the SCLK signal (CPHA= 1). Modes 2 and 3 also differ from each other by their CPHA, however, they have a CPOL of 1, which means the clock polarity in idle state is a logic high. Note that the SPI mode set for the HV Remote board is mode 0 (CPOL= 0 & CPHA= 0), which is the mode recommended by the manufacturers of the digital components used on the board.

This type of interface is appropriate for this project since it has a fast transmission speed, is easy to implement, both in hardware and software, and allows the MOSI, MISO, and SCLK signals to be shared between slaves, as shown in the configuration of figure 2.4. All of the digital components used in the HV Remote board are SPI controlled.

Functional Description

All HV Remote boards in a given crate will share the SPI MOSI, MISO, and SCLK signals. Differentiation between boards will be accomplished by the use of unique CS signals sent by the FPGA development board, CS_CARD. Figure 2.6 presents a functional block diagram of HV Remote's digital control system, which describes the aforementioned board differentiation method and all of the board's functionalities.

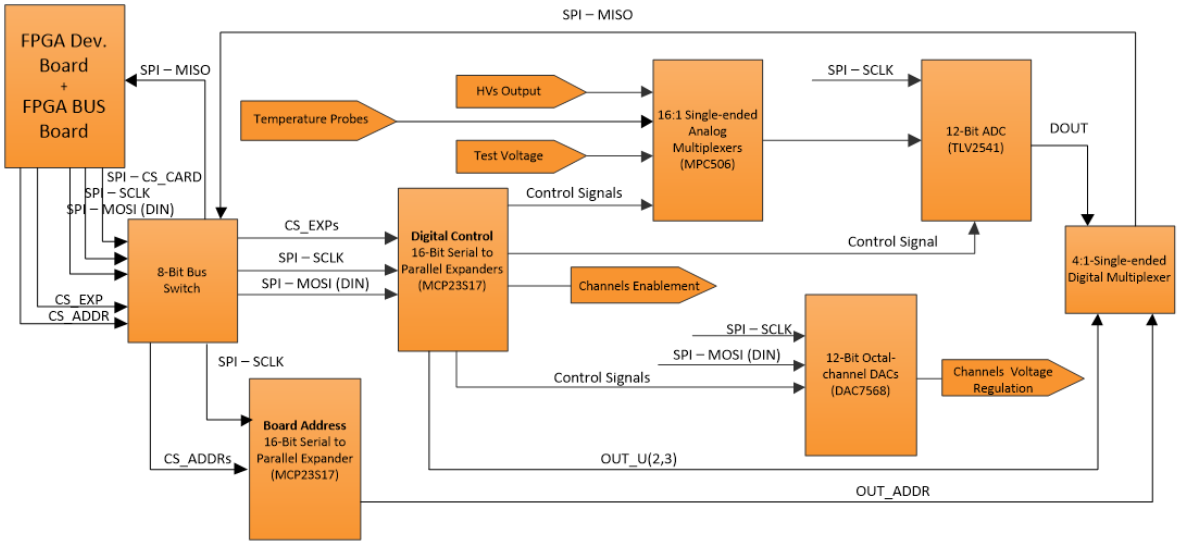


Figure 2.6: Block diagram representing the SPI communication signals.

The CS_CARD signal is the SPI CS signal and, in addition to initiating SPI communication, it selects from 16 boards one to be controlled, by enabling its 8-bit Field-Effect Transistor (FET) bus switch. CS_EXP and CS_ADDR are active low signals shared by every HV Remote board in a given crate and enable the two control 16-bit Input/Output (I/O) serial to parallel port expanders (MCP23S17) and one hardware address serial to parallel I/O port expander, respectively. These signals are always-on but will only be transmitted to the board's respective Integrated Circuits (ICs) if the bus switch is enabled.

Once the bus switch is enabled, communication is established between the FPGA and the three MCP23S17 I/O port expanders. The two 16-bit port expanders, described in figures 2.7 and 2.8, will perform writing sequences to their General Purpose Input/Output (GPIO) pins as they will be responsible for enabling the twelve 4-PMT channel groups, the six 12-bit DACs, the analogue Multiplexers (MUXs), the digital MUX and the 12-bit ADC and for controlling the different MUX selector inputs. All analogue MUXs will share the same selector inputs, as only one MUX at a time will be enabled. The third port expander is responsible for the board's 16-bit hardware address and is further described in this subsection.

The MCP23S17 port expanders have a unique 3-bit hardware address, represented by pins A0, A1 and A2 in figures 2.7 and 2.8. Each pin must be externally biased as a logic high (3.3 V) or a logic low (0 V), forming a 3-bit digital word that must be different from the other port expanders on the board. Since the control port expanders share the same CS signal (CS_EXP), the hardware address pins are important to differentiate between port expanders when communicating. The device also features a hardware reset pin ($\overline{\text{RESET}}$) which sets all of its registers to their default values when set to a low state. To ensure normal operation, the pin must be externally biased with 3.3 V. The placement of a jumper in connectors RS_U2 and RS_U3 in figures 2.7 and 2.8 allows a hardware reset function.

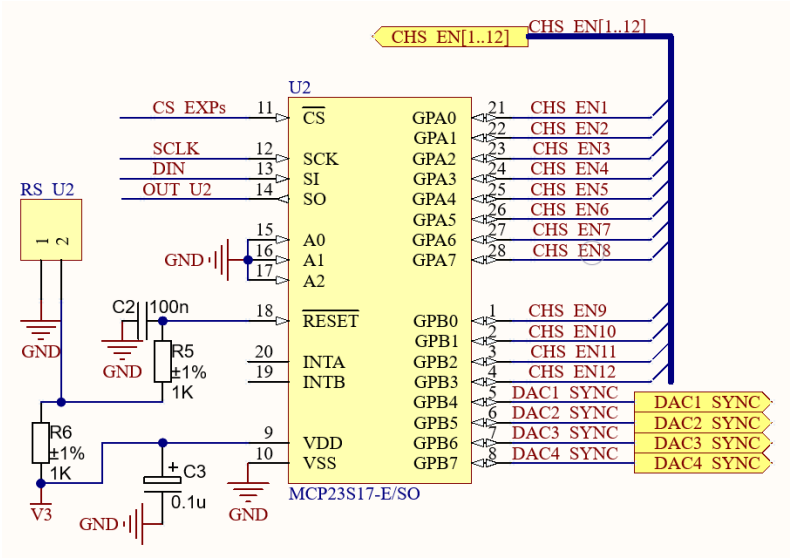


Figure 2.7: Port expander responsible for enabling the 12 PMT channel groups and 4 out of the 6 DACs.

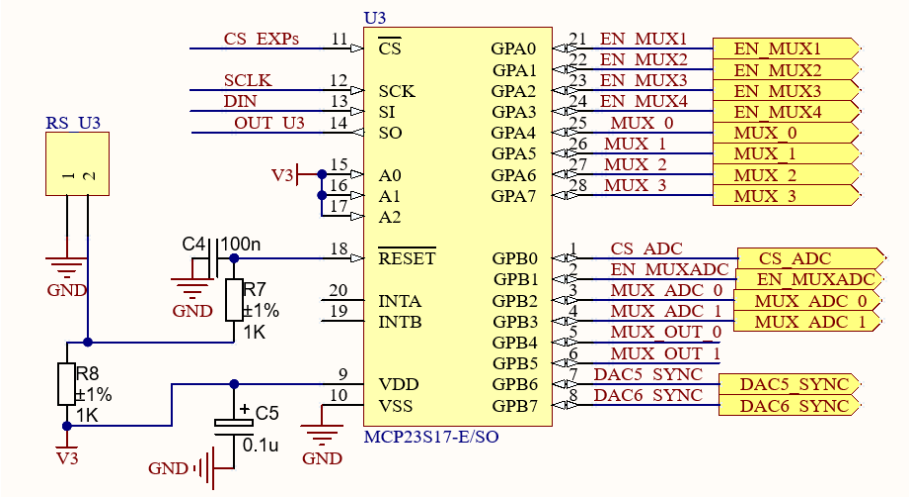


Figure 2.8: Port expander responsible for enabling the analogue MUXs, the digital MUX, 2 DACs and the ADC and for supplying the MUX selector signals for both the analogue and digital MUXs.

Communication with the MCP23S17 I/O Port Expander

The MCP23S17 I/O serial to parallel port expander is at the core of the HV Remote digital control. It controls all of the board's features by converting a given serial 16-bit digital input into 16 1-bit signals that determine the state of the device's GPIO pins. These pins can be configured as either inputs or outputs: the control port expanders pins will be defined as outputs and the hardware address expander as inputs, as they will be externally biased and only meant to be read.

Communication with the MCP23S17 port expander is a bit more complex than just sending a 16-bit word. It is started by lowering the $\overline{\text{CS}}$ line and then sending three 8-bit (1 byte) sequences which must be sent without ever lifting the $\overline{\text{CS}}$ line. The first byte (control byte) contains the 3-bit word address of the device it is intended to communicate with (corresponding to the pins A2, A1 and A0) and the read/write bit which defines the type of operation, as illustrated in figure 2.9.

The second byte corresponds to the address of the register to be changed. The device has 22 individual 8-bit registers which form 11 register pairs that allow the user to configure its different functionalities.

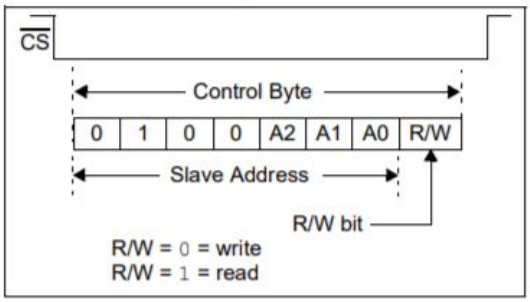


Figure 2.9: Control byte composition, where the first 4 bits are fixed and A2, A1 and A0 correspond to the device hardware address. [19]

Table 2.3 describes each register and their addresses according to the mapping chosen through the BANK bit of the Input/Output Configuration (ICON) register, whose default or Power-On Reset (POR) value is 0. Only the ICON, Input/Output Direction (IODIR), and GPIO registers are used to perform the operations required for the HV Remote digital control system.

Register	Address (binary) IOCON.BANK = 1	Address (binary) IOCON.BANK = 0
IODIRA	00000000	00000000
IODIRB	00010000	00000001
IPOLA	00000001	00000010
IPOLB	00010001	00000011
GPINTENA	00000010	00000100
GPINTENB	00010010	00000101
DEFVALA	00000011	00000110
DEFVALB	00010011	00000111
INTCONA	00000100	00001000
INTCONB	00010100	00001001
IOCON	00000101	00001010
IOCON	00010101	00001011
GPPUA	00000110	00001100
GPPUB	00010110	00001101
INTFA	00000111	00001110
INTFB	00010111	00001111
INTCAPA	00001000	00010000
INTCAPB	00011000	00010001
GPIOA	00001001	00010010
GPIOB	00011001	00010011
OLATA	00001010	00010100
OLATB	00011010	00010101

Table 2.3: MCP23S17 registers' address mapping. The POR value of the IOCON.BANK bit is 0.

The ICON register allows the user to configure, for example, the way registers are addressed through its BANK bit, the mode of operation (sequential or byte) with the SEQQP bit and the hardware address feature via the HAEN bit. On the other hand, the IODIR register allows the user to configure the GPIO pins as inputs or outputs by setting the corresponding bits as either 1 or 0, respectively. The GPIO register represents the pin value, where 1 represents high and 0 low.

The third and final byte is the data byte, which is the data to be written on the selected register. However, if it is a read operation, this byte is discarded, as it is intended to obtain a byte from the device. Once the third byte is sent, the selected register is successfully altered and the $\overline{\text{CS}}$ line is lifted.

All of the board's functions are controlled by the described digital control system, except for the interlock fault, described in section 2.1.3.

Board Address

Each board features a unique 16-bit hardware address to identify each HV Remote board. This address is implemented through two hardware switches, that allow to physically set a hardware address with ground and 3.3V signals, as shown in figure 2.10. These signals are then converted by a 16-bit serial to parallel I/O port expander, which is then sent to the crate digital control through the SPI MISO signal.

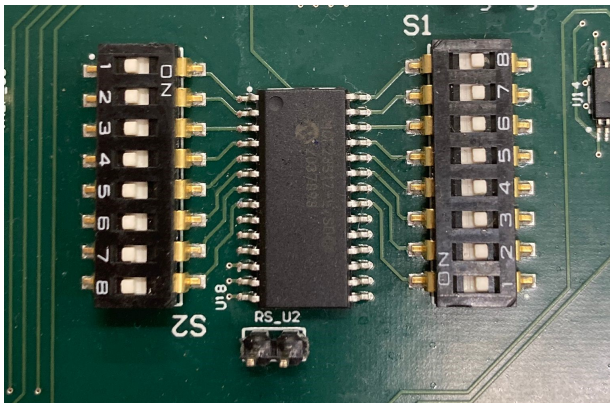


Figure 2.10: Picture of HV Remote's hardware address mechanism.

2.1.3 Interlock Fault

The interlock circuitry consists of a line linking two connector pins. If this connection is not made, it means the HV Remote board is not connected to the crate's midplane and an interlock fault is generated in the HV Supplies board. Following an interlock malfunction, the board's power supplies are turned off and a signal is sent to the TileCal DCS via its dedicated and independent digital control system and buses.

After the connection is correctly established between the HV Remote board and the crate, the HV Supplies and HV Remote boards can be restarted with a TileCal DCS enable. The boards' restart parameters will be memorised in the FPGA control board in order to start the configuration properly.

2.1.4 HV Regulation System

The HV regulation system is composed of key parts: regulation loops, on/off switches and monitorization loops, as described in figure 2.11. The HV channels' individual regulation is performed by 48 regulation loops that use the two primary voltages supplied by two DC/DC converters from the HV Supplies board, HV1& HV2, that supply each side of the board individually, due to current constraints described in section 2.2. The DC/DC converters that supply these voltages have a maximum HV output value of -1000 V, however they are not intended to operate above -950 V. The regulation system is controlled by the HV Remote digital control system, which will later be controlled by the TileCal DCS.

An individual PMT HV is defined by the user in the HV Remote GUI, which sends a serial 16-bit word through the MOSI SPI signal to a set of six octal-channel, SPI controlled, 12-bit DACs (DAC7568) that have an internal reference of 2.5 V. Figure 2.12 shows one of the boards' DACs circuitry. Each DAC is enabled through its active low $\overline{\text{SYNC}}$ pin which is connected to the respective DAC_SYNC signal provided by the control port expanders, described in figures 2.7 and 2.8.

The D_{IN} pins of each DAC are directly connected to the SPI MOSI signal. This means that only one DAC can be enabled at a time and the CS_EXP signal must be high when transmitting commands.

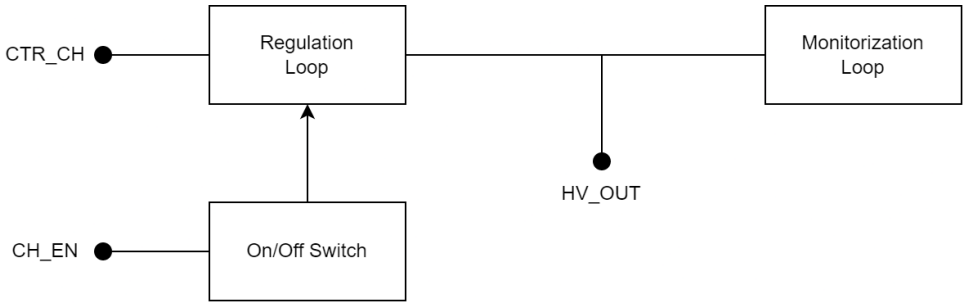


Figure 2.11: HV regulation system diagram.

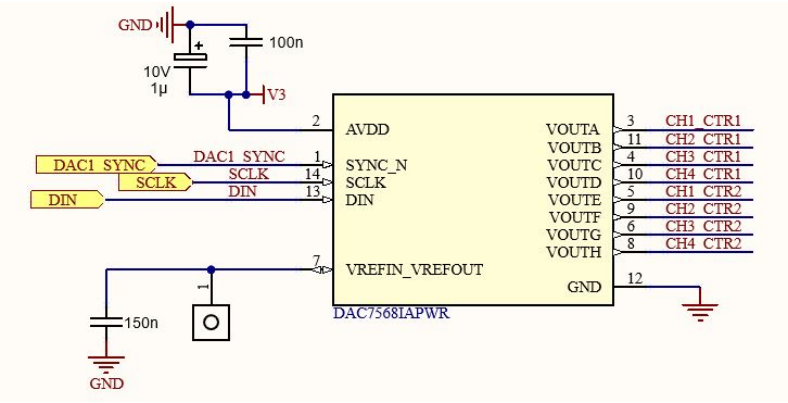


Figure 2.12: Circuitry schematic of the 12-bit octal-channel DAC7568.

Communication with the DAC7568 is made by commanding specific ”write and update” sequences, described in the device’s datasheet [20], to its 32-bit wide input shift register.

Once a digital command is written to the DACs’s input shift register, it will be converted into an analogue voltage between 0 V and the DACs’ internal reference voltage. This analogue voltage is the input signal of the HV regulation loop, labelled as CH_CTR in the diagram featured in figure 2.13, which describes the circuitry of the regulation loops.

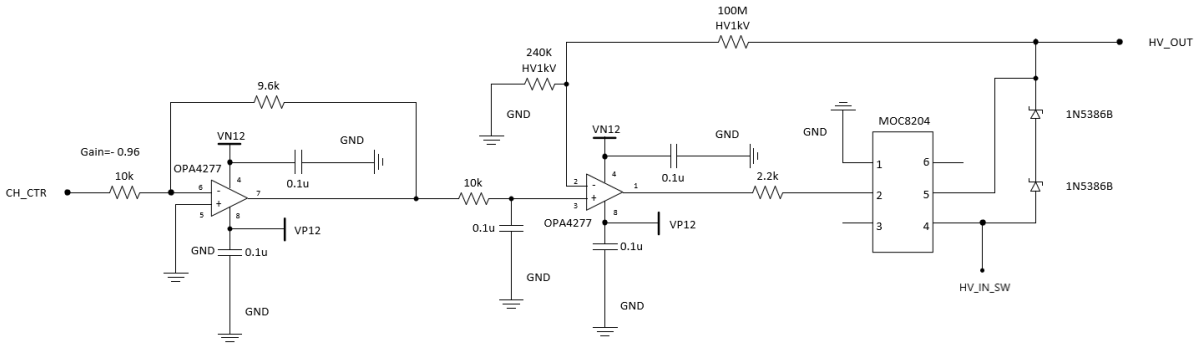


Figure 2.13: Schematic of the regulation loop circuitry.

CH_CTR is the positive analogue signal that sets the desired negative HV output. An inverter voltage amplifier with an adjusted gain of 0.96 and a passive low-pass filter adjust the CH_CTR low voltage signal. This signal is adjusted by taking into account a fraction (a voltage divider with two resistors of 100 MΩ and 240 kΩ was used) of the negative output HV, HV_OUT.

An optocoupler performs the decoupling between the output adjusted low voltage signal and HV_OUT. An optocoupler is an electronic component that uses light to transfer electrical signals between

two isolated circuits. It is typically made up of a Light-Emitting Diode (LED) and a phototransistor. The one used (MOC8204) consists of an infrared emitting diode and phototransistor.

Two zener diodes with a zener voltage (V_Z) of 180 V [21] are placed at the optocoupler's output, to ensure the output voltage is in the interval [$HV_{in} + 360$ V, HV_{in}], where HV_{in} is one of the primary voltages supplied by the HV Supplies board.

The on/off switch circuit, illustrated in figure 2.14, is responsible for switching on/off the primary HV from HV Supplies to the HV regulation loop. Since the channels are now grouped in sets of four, each circuit remotely switches on/off sets of four regulation loops at a time. The circuitry also features a manual switch for each channel which is not featured in figure 2.14.

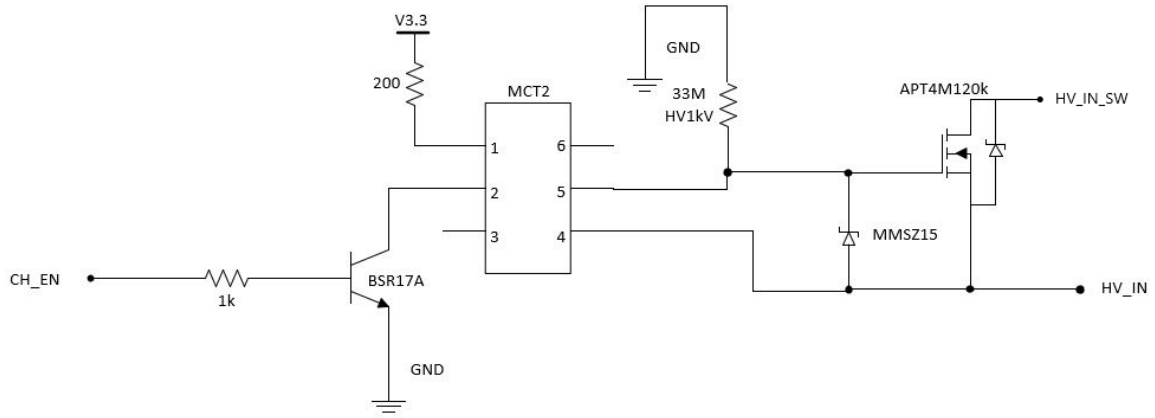


Figure 2.14: Schematic of the on/off switch circuitry.

When an active-high enable signal is sent, the NPN bipolar transistor is in its saturation region and consequently, behaves as a closed switch, switching the optocoupler (MCT2) on. Connected to the optocoupler's output is a Zener diode and a HV N-channel Metal–Oxide–Semiconductor Field-Effect Transistor (MOSFET) that switches on the HV.

The output HV signal, HV_OUT, is also monitored by the digital control system. This is possible by connecting it to a voltage divider, a low-pass filter and a buffer, as described in figure 2.15. The voltage divider converts the negative HV to a negative low voltage, while the low-pass filter and the buffer are used to filter and decouple the resulting analogue signal.

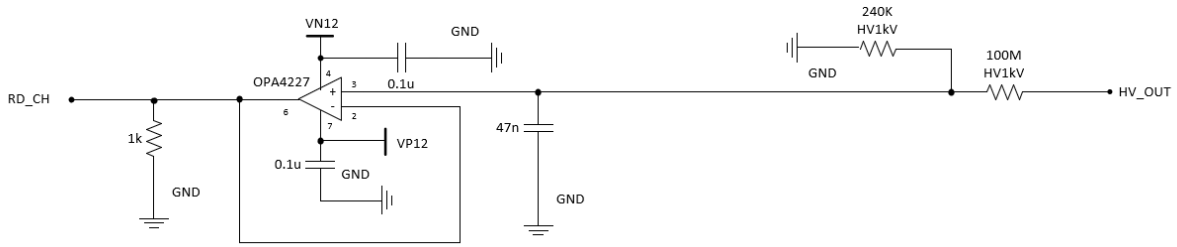


Figure 2.15: Schematic of the circuitry responsible for the conversion of the HV to a monitoring low voltage signal.

The output of each monitorization loop, RD_CH, is connected to an input pin of one of three analogue MUXs, as shown in figure 2.16. The analogue MUX used for this purpose is the MUX36S16, a 16:1 MUX with a dual power supply of up to ± 18 V.

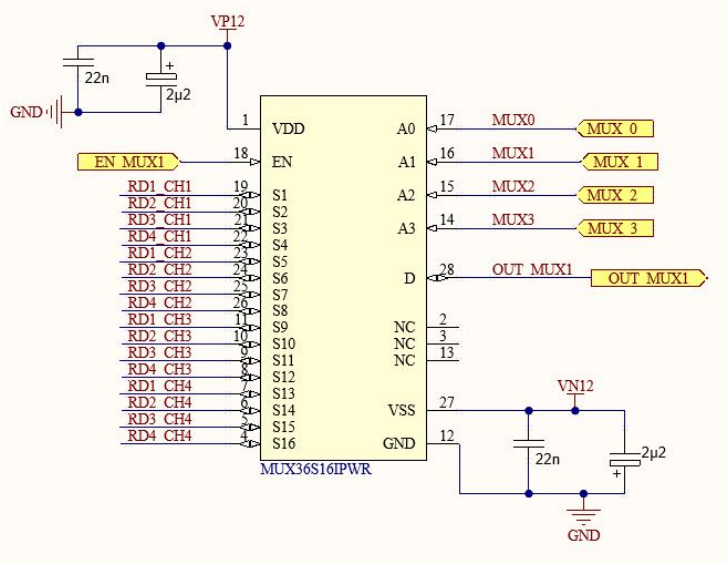


Figure 2.16: Circuitry of one of the board's three MUX36S16 analogue multiplexers.

Each of these MUXs outputs are connected to a 4 : 1 analogue multiplexer, that selects between the 48 individual HV outputs, the temperature probes output voltages and the stability test voltage. Its output is connected to an inverter voltage amplifier without gain, which converts it into a positive analogue voltage transmitted to an ADC. Figure 2.17 describes the circuitry of the aforementioned description.

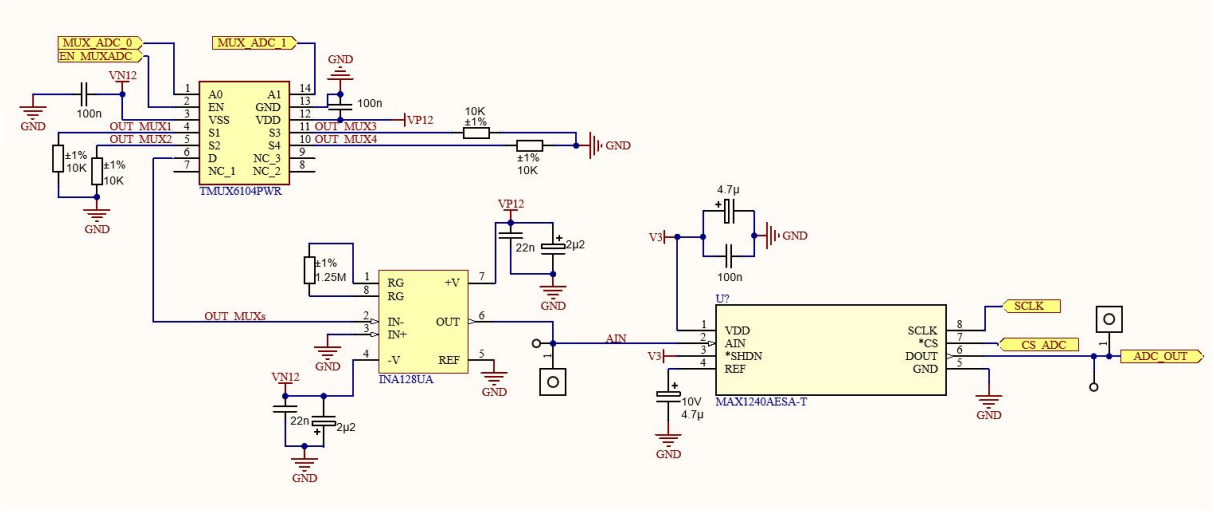


Figure 2.17: Circuitry responsible for the HV monitorization: a 4:1 analogue multiplexer connected to an inverting amplifier circuit and an ADC.

The MAX1240 [22] 12-bit ADC then converts the selected analogue voltage to a digital signal to be transmitted via SPI to the FPGA. It has an internal reference voltage of 2.5 V, which is enabled by pulling its $\overline{\text{SHDN}}$ pin high. The full scale of the input voltage (A_{IN}) is defined by the reference voltage.

The ADC is enabled by lowering its $\overline{\text{CS}}$ line and conversion begins. While conversion is ongoing, its digital output (D_{OUT}) is set to low. In order to get the 12-bit of data, two consecutive 8-bit (1 byte) reads are required. The first byte consists of a leading 1 followed by seven bits resulting from the conversion. The second byte contains the remaining five bits and three zeros which are to be ignored. Figure 2.18 shows the timing in which the transmission is made. The data bits are interpreted as a decimal number between 0 and 4095 ($2^{12} - 1$) which corresponds to a scale between 0 and 2.5 V, that is, the ADC's Least

Significant Bit (LSB) equals to a voltage of approximately 0.61 mV.

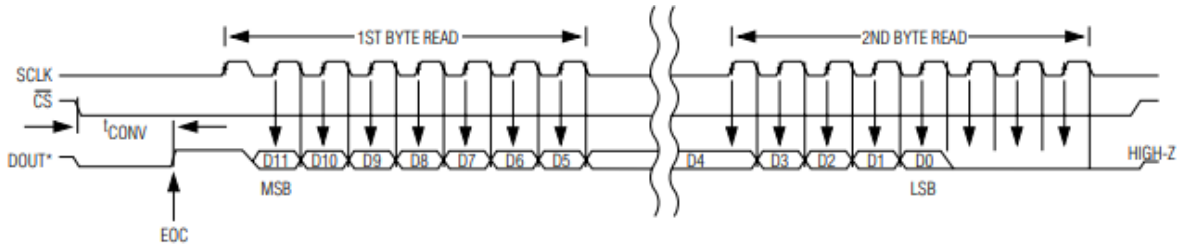


Figure 2.18: MAX1240 Serial Interface timing and data transmission. D11 to D0 are the data bits, CONV is the conversion period, MSB is the Most Significant Bit and LSB is the Least Significant Bit. [22]

A final 4 : 1 digital MUX then selects the signal to be read by MISO between the ADC digital output and the three port expanders output registers. This allows the user to perform different voltage readings as well as monitor the commands in place upon request.

2.1.5 HV Remote Mapping and Input Pin Configuration

Figure 2.19 shows the composition of each HV output channel group and their localisation in the board. Each of HV Remote's output channels was assigned to a TileCal PMT, in order to be more easily identified when testing. The mapping performed is described in table 2.4 and is the reference used by the TileCalDCS.

HV Connector 1			HV Connector 2			HV Connector 3			HV Connector 4		
PMT	Pin	Channel	PMT	Pin	Channel	PMT	Pin	Channel	PMT	Pin	Channel
—	25-14	GND	—	25-14	GND	—	25-14	GND	—	25-14	GND
—	13	SHIELD	—	13	SHIELD	—	13	SHIELD	—	13	SHIELD
1	12	5	13	12	29	25	12	44	37	12	17
2	11	6	14	11	30	26	11	43	38	11	18
3	10	7	15	10	31	27	10	42	39	10	19
4	9	8	16	9	32	28	9	41	40	9	20
5	8	13	17	8	37	29	8	33	41	8	9
6	7	14	18	7	38	30	7	34	42	7	10
7	6	15	19	6	39	31	6	35	43	6	11
8	5	16	20	5	40	32	5	36	44	5	12
9	4	21	21	4	48	33	4	25	45	4	1
10	3	22	22	3	47	34	3	26	46	3	2
11	2	23	23	2	46	35	2	27	47	2	3
12	1	24	24	1	45	36	1	28	48	1	4

Table 2.4: Assignment of HV Remote output channels to each of the PMTs to be powered.

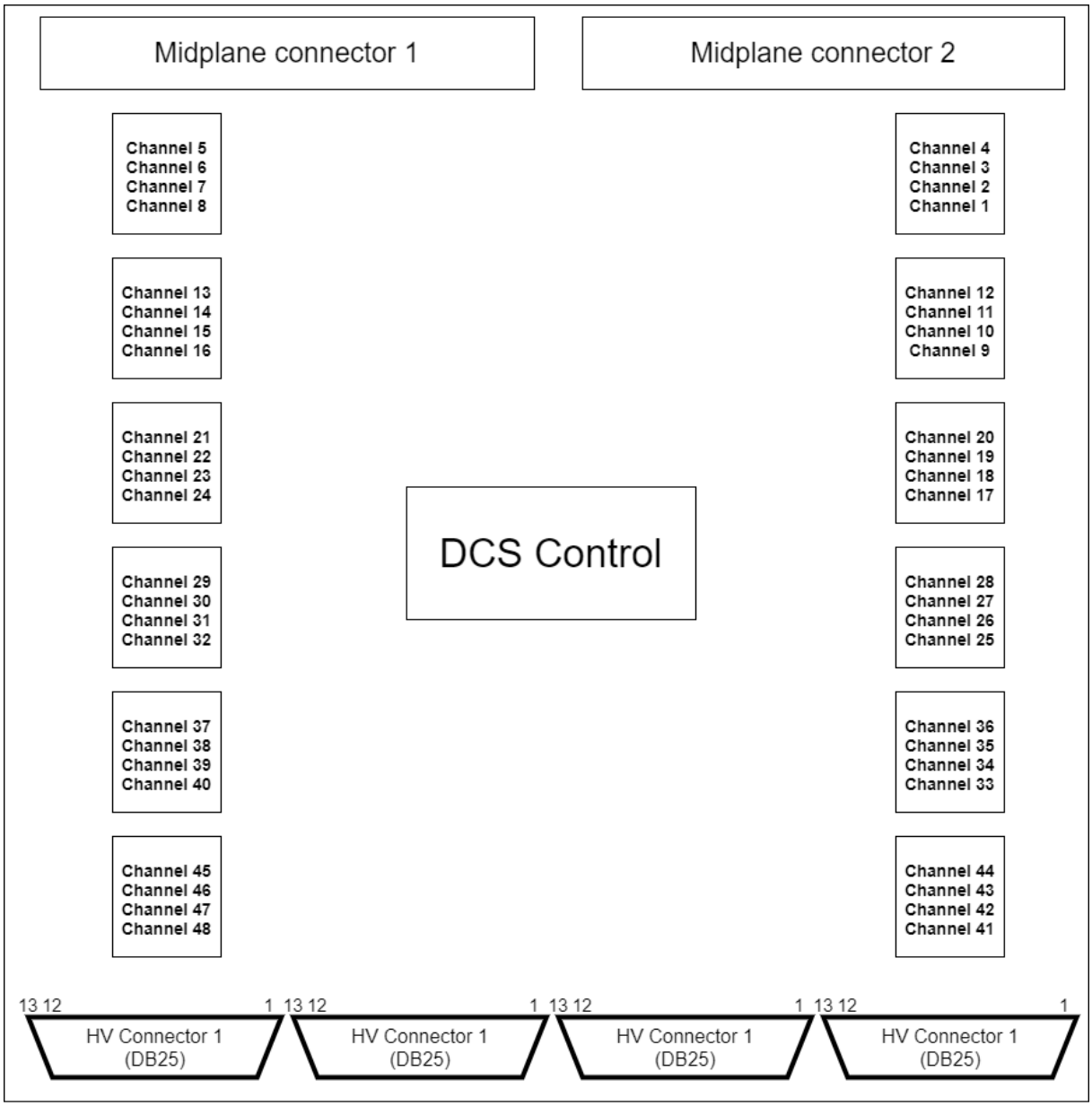


Figure 2.19: Pin configuration and HV output channel mapping.

2.2 HV Supplies Board

The HV Supplies board (Figure 2.20) is designed to supply the HV Remote board with the primary HVs and LVs needed for its regulation loops, enabling and reading circuitry, temperature monitoring, test voltage and digital control system. Since this work focuses on the HV Remote board, only a short description of the HV Supplies will be featured.

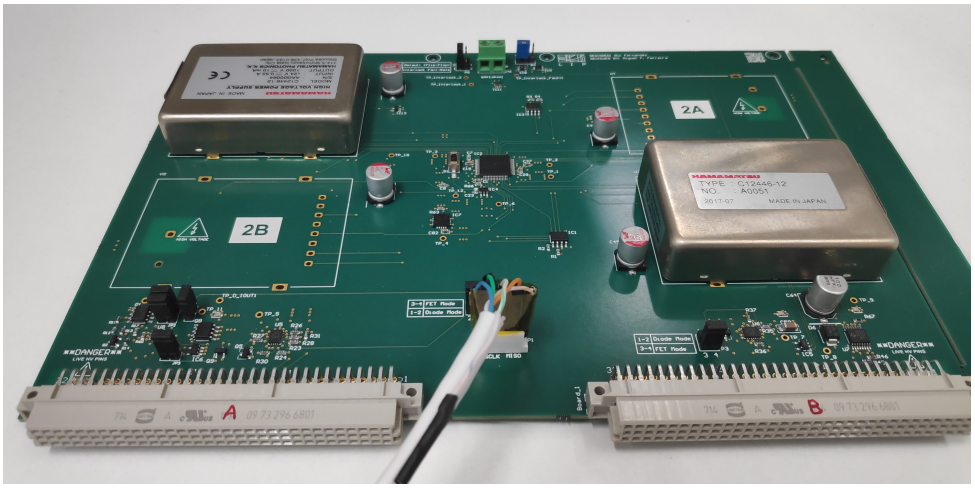


Figure 2.20: Picture of the first prototype of the HV Supplies board.

In order for the HV Supplies board work properly, it needs 4 different primary voltages, them being 24 V @ 20 A, 12 V @ 15 A, −12 V @ 10 A, and 3.3 V @ 3 A, supplied by the crate's power supplies to which the HV Supplies boards will be connected via a midplane board as shown in Figure 2.1.

Regulation of its HV outputs is possible through two independent DC/DC boost converters with a 24 V input and a programmable output voltage between 0 and −1000 V. A DC/DC converter is an IC that converts one Direct Current (DC) voltage into another. It can also be designated as a linear or switching regulator, depending on its conversion method. The converters used in this application are switching regulators that regulate voltage by temporarily storing its input power, via inductors, transformers or capacitors, before releasing it to its output at a different voltage. This type of regulator can either output a higher voltage (Step-up or Boost converter) or a lower voltage (Step-down or Buck converter). They are more commonly used since they are more adaptable, power-efficient and stable. [23]

The chosen DC/DC converter for this application has a maximum output current of 10 mA and each PMT consumes a current of approximately 0.4 mA. Thus, two DC/DC converters were implemented to supply all the 48 PMTs connected to each HV Remote/HV Supplies pair.

In addition to regulating its HV outputs, the board also allows readings of current and voltage consumption, both for high and low voltage. The board also contains two temperature transducers, placed in different positions, that allow careful temperature monitoring.

Hot swaps, which allow the protection, fault detection and switching of all the power supplies in case of replacement or addition of boards without shutting down all crate cards, were also implemented at the inputs of all feeds. It is also possible to turn the power on/off, using software or hardware, thus ensuring permanent control of the board. [24]

The board also includes the implementation of an interlock system in which an interlock line is made available and its current is monitored. It is responsible for sending an interlock fault to the TileCal DCS in case the HV Remote board is not connected to the HV Supplies board correctly, which switches off all power supplies of these boards. After an interlock fault, the boards may be restarted through a

TileCal DCS enable, if there is an appropriate connection of the boards. The HV Supplies board has an independent SPI bus, allowing the monitoring of the primary power sources and their fault detection, and the HV Remote's digital control to work simultaneously.

Chapter 3

Cable Test Board

As described in chapter 2, each HV Remote board will be connected to the PMTs via 100 m cables. The use of these cables results in an extra source of noise and alters the transmitted signal, albeit in a very small way. Its impact must be quantified so that the voltage supplied to each PMT is adjusted accordingly.

Each cable has a different resistance that must be quantified. In addition, there are different types of interference or noise that are introduced by the use of long cables, such as electromagnetic induction and parasitic capacitance.

Since these cables must be tested beforehand, the Cable Test (CT) board was designed to perform continuity and insulation tests. Continuity tests consist of determining whether a circuit is closed or not, in this case, if the cables transmit the applied voltage. On the other hand, insulation tests consist of supplying a cable with HV and measuring the voltage of the neighbouring channels, thus calculating the magnitude of the caused interference.

With this in mind, the cable test system must be capable of distributing negative HV to a set of 48 cables and measuring the cables' voltage with a resistive circuit. Each long cable has two twisted wires, one for HV and another one for its GND. Both wires and their interference on each other must be tested, which will take up 2 signals/connector pins for each cable. As such, the system must perform readings of 96 signals: 48 HV + 48 GND wires.

To select and distribute HV to a single cable at a time, it would be necessary to implement a matrix of relays as programmable switches. Components such as relays take up a lot of space on a PCB and are very expensive. When studying all the viable options for implementing this system, it was noted that the board developed by Pedro Rato [1] for the HV Remote test bench, HV Reading, has two independent relay matrices, each one comprising 48 relays that could be used.

In order to minimise the complexity and cost of the cable test system, it was determined that it would be composed of the simultaneous use of the two boards (HV Reading and Cable Test). In this configuration, the CT board will be responsible for supplying negative test HV to the HV Reading board and for carrying out continuity and voltage readings for each of the connected cable's wires. The HV Reading board, described in section 3.1, will supply the CT board with the required DC voltage supplies (+24V and +3.3V) and SPI control signals (MOSI, MISO, SCLK & CS) as well as distribute the test HV to each cable connected through relay matrices. These relay matrices will be selected via MUXs which are controlled by a dedicated digital control system, further described in this chapter.

The test system must be prepared for use in a laboratory environment at Laboratório de Instrumenção e Física de Partículas (LIP) and in the ATLAS laboratory facilities, where tests with laser and beam will also be performed. As such, the cable test board was designed to be prepared for both setups. The CT board will supply adjustable high voltage to the HV reading board in both of them. The most significant difference between the two setups is the termination of the long cables.

In the LIP laboratory setup, the cables will connect the HV Reading board to the CT board, which will measure the signal at the cable terminal and verify the signal changes in response to resistive circuits, PMT emulators, and the PMT itself. However, in the ATLAS facilities configuration, the cables will connect the HV Reading board directly to the TileCal PMTs or to a replica of them for test purposes. Since the cables will not be connected to the CT board, it will not be possible to take voltage measurements, thus only continuity tests will be performed in this configuration. This will be achieved by monitoring a single signal while supplying HV to a cable at a time. Figures 3.1 and 3.2 show simple block diagrams that illustrate the LIP laboratory and ATLAS facilities setups, respectively.

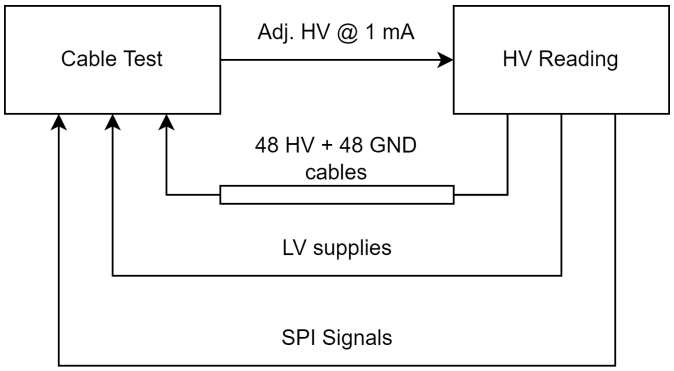


Figure 3.1: Block diagram describing the LIP laboratory setup of the cable test system.

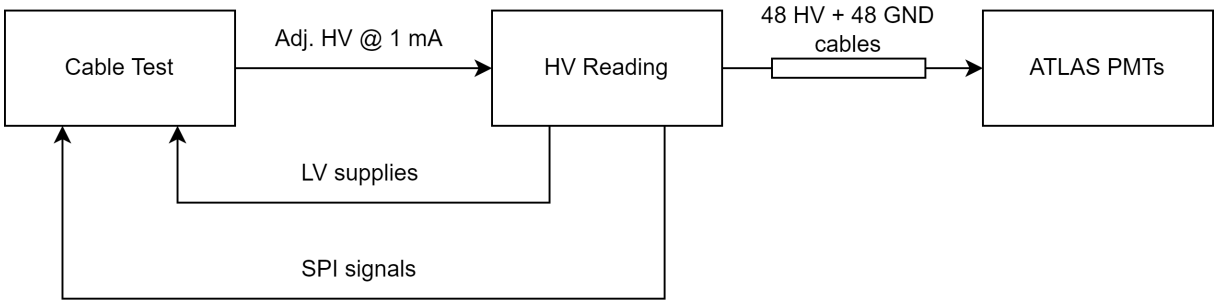


Figure 3.2: Block diagram describing the setup of the cable test system in the ATLAS facilities.

In this chapter, the development of the board will be described in detail. Dedicated interface and control software was also developed for this board and will be described in chapter 4.

3.1 HV Reading

The HV Reading board is, at the time of writing this thesis, in production, with its second prototype already being developed.[2] Only the board's relevant functions to the cable test configuration will be discussed. It is mainly composed of relays acting as programmable switches able to select one HV input or output at a time.

A relay is an electrical component made up of a coil and one or more switches connecting isolated circuits. When there is current in the coil's circuit, a magnetic field is induced, forcing the switch to close an external circuit. If there is no current in the coil, there is no magnetic field and thus the switch is open. Figure 3.3 describes the most common types of relays and their function.

In the HV Reading board, each HV channel has its own dedicated relay. There are two relay matrices, each one comprising 49 relays connected to one common terminal, named COM, and the remainder necessary circuitry to enable each relay at a time. [1] Figure 3.4 shows the schematic of one relay matrix, in which the relays are represented by only their switches, for simplicity.

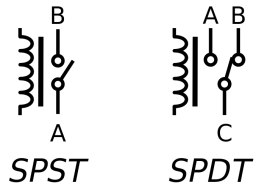


Figure 3.3: Circuit symbols of the most commonly used relays: Single-Pole Single-Throw (SPST) and Single-Pole Double-Throw (SPDT). [25]

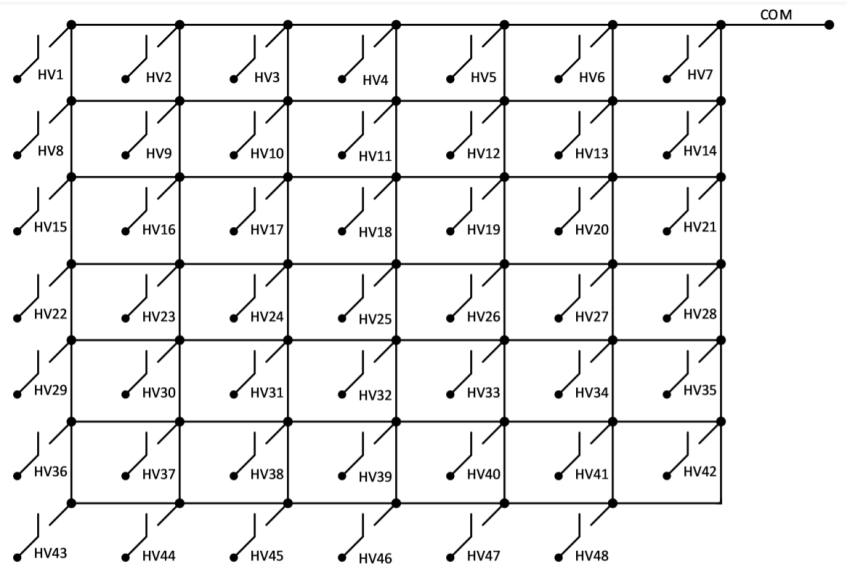


Figure 3.4: Schematic of one relay matrix. [1]

The HV Reading board would, in its primary mode, be used to test in real-time one or two HV Remote boards via its DB25 connectors, receiving up to 96 HV inputs, selecting one through its relay matrices, and performing voltage readings.

In cable test mode, the operation will be nearly reversed, with the HV input being just one, coming directly from the CT board and distributed by the relay matrices to the HV Reading's DB25 connector pins, which will now act as HV outputs to which the ground and HV wires of the long cables will be connected. At any given time, only one output signal can have a HV signal and be monitored.

With this configuration in mind, connectors for the test HV input from the CT board and the necessary power supplies were added to the HV Reading board. This way, the HV Reading board was altered to allow two modes of operation: the HV Remote test mode and the Cable Test mode. The selection between the two modes is made by another relay and, depending on the mode selected, the board will be either reading or distributing voltage.

3.2 Project Design

The board's project design began with component selection and the development of schematics of the board's key parts, starting with the circuitry at each input of the board. All HV and GND wires of each cable will be connected to the CT and HV reading boards through DB25 connectors, where the two wires corresponding to each of the cables must be arranged so that there is as little separation as possible between them when connecting to the boards.

To perform voltage readings on each cable using an ADC and SPI communication, all HV wires must be connected to a voltage divider, with resistors sized according to the maximum HV that can be supplied and the reference voltage of the chosen ADC. Therefore, the component responsible for supplying the HV, a DC/DC converter, and the ADC were chosen beforehand. The chosen DC/DC converter has a maximum voltage supply of -1500 V and the ADC has a voltage reference of 2.5 V. Both components and their circuitry will be further discussed in this chapter.

Since design specifications for HV resistors usually require low inductance and temperature coefficients, the resistors composing the voltage divider were chosen carefully. The divider is composed of a 680 kΩ HV resistor and a 1 kΩ resistor. With a maximum HV supply, the output voltage will be -2.2 V.

It was determined that all GND wires should also be connected to a similar voltage divider, instead of being directly connected to the ADC, to prevent any damage that could be caused to the board if a HV wire is accidentally connected to an input assigned to a GND wire or if the cables' insulation is compromised.

In addition to measuring cable continuity, it is important to analyse how different types of loads affect cable performance. For this purpose, the HV wire's inputs are also connected to pin headers that allow alternating between only resistive load, by not connecting any of the pin headers, a capacitive load that simulates a PMT or a PMT by connecting the respective pin headers with jumper or the PMT itself. Figures 3.5 and 3.6 show the circuitry to be implemented at the input of the HV and GND wires, respectively.

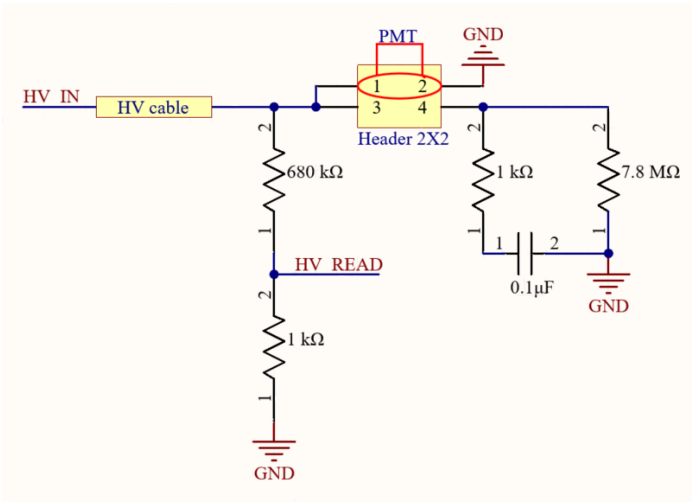


Figure 3.5: Schematic of the circuitry at the input of each HV wire, where pin headers 1 and 3 are connected to the input HV (HV_IN), pin 2 is not connected and pin 4 is connected to the PMT emulator circuit.

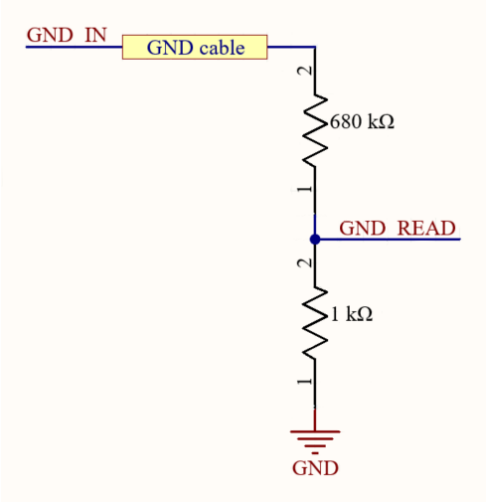


Figure 3.6: Schematic of the circuitry at the input of each GND wire, that only contains a resistive voltage divider.

The HV cables are shielded, which means that all of its wires (HVs and GNDs) are enclosed by a common conductive layer, which must also be tested. Therefore, each of the four shields will be connected to a pin header with a resistive voltage divider shown in figure 3.7, with a 2×2 pin header right before the voltage divider, to allow the user to connect the shield either to a GND signal or to the voltage divider.

When performing cable tests at CERN, the setup will be significantly different. The cables will be connected directly from the HV Reading board, which performs the cable selection, to the PMTs located in the ATLAS facilities. In this setup, the CT board will only supply the adjustable HV to the HV Reading board. Therefore, it will not be possible to carry out insulation tests, as in the aforementioned configuration.

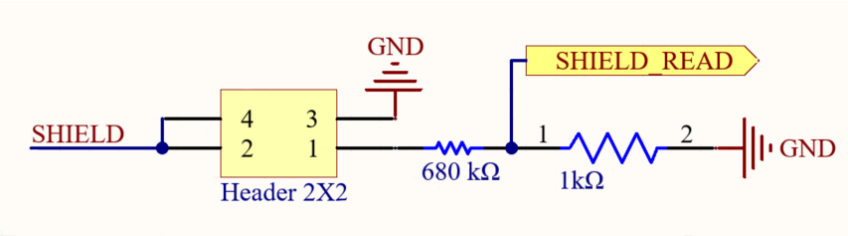


Figure 3.7: Schematic of the circuitry at the input of each shield connection, in which the 2×2 header allows to toggle connection between ground and a resistive voltage divider.

However, it is possible to verify whether or not there is continuity in the cables by measuring the current that passes through a voltage divider connected to the HV output that goes through the selected cable. If there is no current flowing through the cable, the voltage divider branch will have a current of 1 mA, which is the fixed output current of the DC/DC converter that supplies the HV. This implies that the voltage divider output will be maximised, with its value depending on the resistor dimensions and the DC/DC converter output HV. The resistors were dimensioned so that the voltage divider output never exceeds (in absolute value) the value of the ADC's internal reference, which is 2.5 V.

On the other hand, if there is current flowing through the cable, ie, there is continuity, then the input currentwill be split into two branches. Since the resistance of each cable is approximately 15 Ω, which is very small in comparison to the resistance of the voltage divider, the majority of the current will flow through the cable. This means that the voltage divider's output voltage will be smaller, as there will be a significant voltage drop in the divider's first resistor. Figure 3.8 shows the described circuit, which uses precision HV resistors to allow cable resistance measurement.

The implementation of a 2×2 connector to the HV output of the DC/DC converter allows for the selection between the LIP laboratory and ATLAS setups. As a result, this output HV can be connected directly to the HV Reading board (LIP setup) or to a resistive voltage divider used to make current measurements (ATLAS setup).

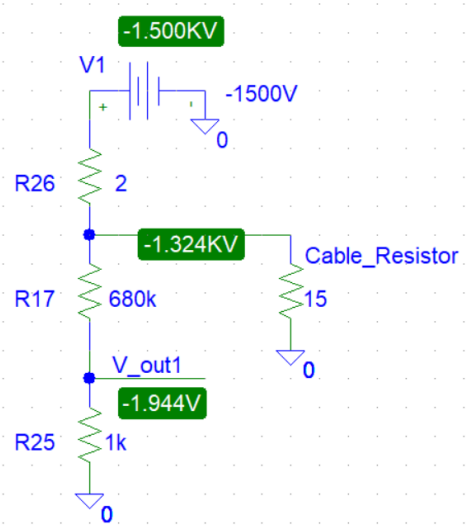


Figure 3.8: Simulation of the impact of cable continuity in voltage readings (V_out1), the 15 Ω resistor emulates the cable resistance.

3.2.1 Power Supplies

The required negative HV is supplied by the 1.5M24-N1 by Advanced Energy, a DC/DC converter with a maximum output voltage of -1500 V @ 1 mA. It has an input voltage of 24 V, supplied by the HV Reading board.

The DC/DC converter's functional block diagram is presented in figure 3.9, where the Voltage Monitor pin (pin 6) provides a LV proportional to the output HV (-1V per -1kV output) via a voltage divider. The relation between the two voltages is linear across the entire range, allowing any error in the system to be easily compensated.

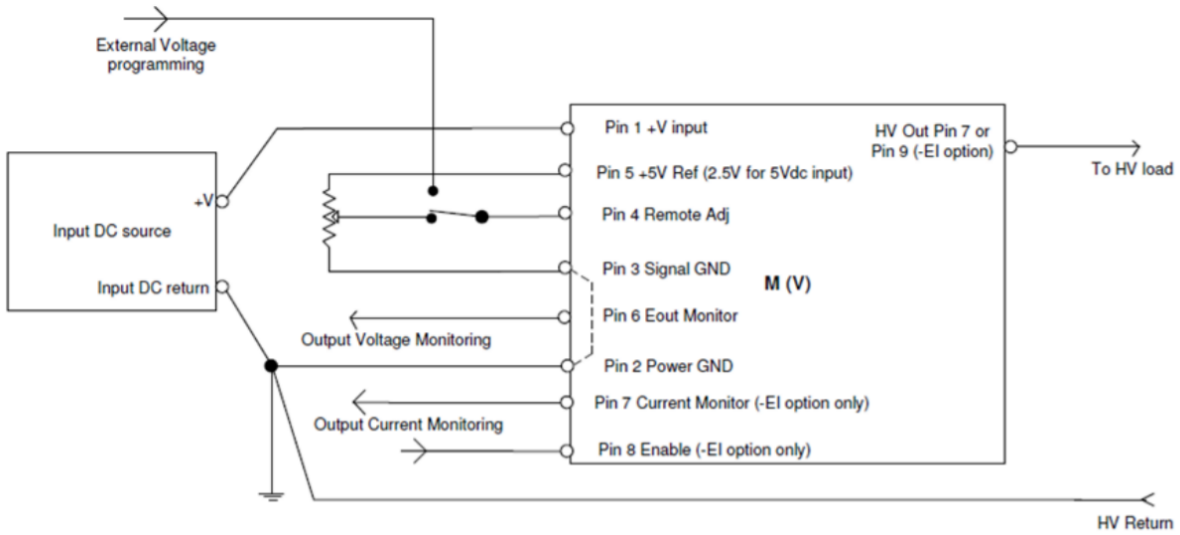


Figure 3.9: Functional diagram and pinout of the 1.5M24-N1 Advanced Energy DC/DC converter [26], which has not the -EI option available, therefore the corresponding -EI pins must be disregarded.

The Remote Adjust Input (Pin 4) allows a LV analogue signal, between 0 V and 5 V, to control the output HV. The control signal can be generated by a DAC or a potentiometer-controlled voltage connected to the converter's internal reference (pin 5), which is 5 V. If 0 V is supplied or if no connection is made to the remote adjust pin, there will not be any output HV. It was decided to use both control options, using pin headers and jumpers to switch between them. According to the device's datasheet [27], the potentiometer's value must be between 10 kΩ and 100 kΩ, so a 100 kΩ potentiometer was selected.

A 10-bit DAC with an output signal between 0 V and 5 V was also implemented in order to control the output HV. However, this DAC requires a 5 V voltage supply, which the HV Reading board does not provide. As a result, this voltage must be generated using a step-down DC/DC converter that takes 24 V as input and is connected as shown in figure 3.10a.

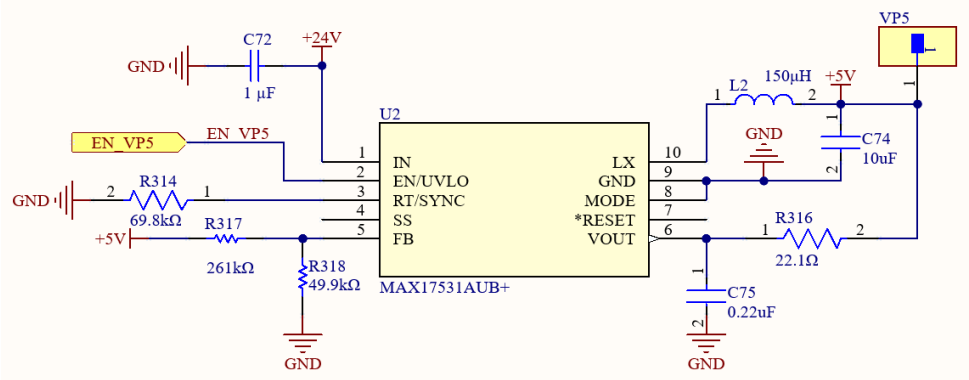
Furthermore, it must be considered that the output voltages of each of the voltage dividers at the inputs of the HV and GND cables' wires are negative; yet the ADC can only convert positive analogue voltages. This presents two options: either place inverter circuits at the output of each voltage divider (a total of 97 inverter circuits) and use digital multiplexing, or use analogue multiplexing capable of taking as input the negative voltage values and place a single inverter circuit right before the ADC.

It is preferable, and simpler, to implement analogue multiplexing. After searching the different analogue MUXs existent on the market, the MUX506 was chosen. However, it requires a dual supply voltage of ± 12 V, which is also not provided by the HV Reading board, so it must be generated by the CT board with another two step-down DC/DC converters.

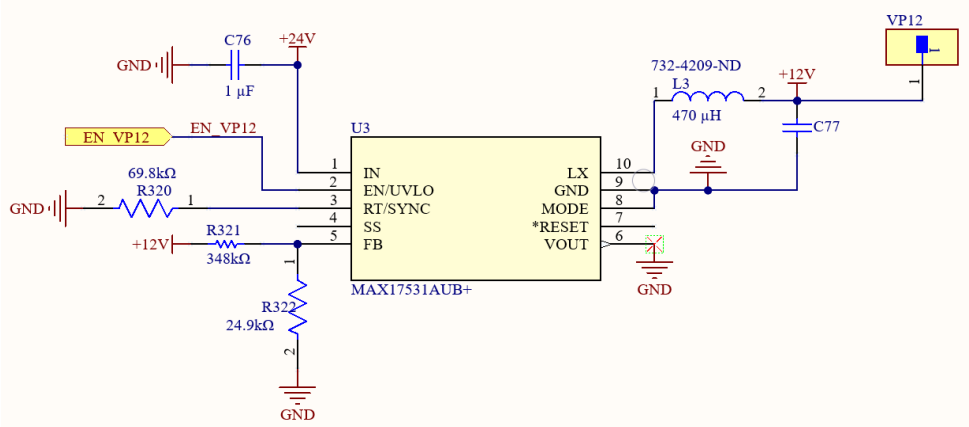
While searching for the converters to be used, it was found that the voltage supplies of the DAC and the MUXs can be generated using the same component: the MAX17531 from Maxim Integrated [28], a step-down DC/DC converter that provides an adjustable 0.8 V up to $0.9 \times V_{IN}$ voltage output with 50 mA output current.

Current consumption calculations were carried out for each of the necessary low voltage sources (5 V, +12 V and −12 V), to verify if the MAX17531 output current is sufficient to cover the consumption of all supplied components. The 5 V source will only power the DAC, which has a maximum consumption of 0.16 mA; the +12 V and −12 V sources must provide an output current of at least 1.25 mA and 1.08 mA, respectively.

Consequently, the CT board has four DC/DC converters: one 1.5M24-N1 from Adavanced Energy, that can generate up to −1500 V @ 1 mA, and three MAX17531 from Maxim Integrated that generate 5 V @ 50 mA, +12 V @ 50 mA, and −12 V @ 50 mA, depending on their configurations. Figures 3.10a and 3.10b describe the application circuits for the 5 V and the 12 V outputs, respectively, as recommended by the manufacturer.



(a) 5 V output.



(b) 12 V output.

Figure 3.10: Implemented circuitry of the MAX17531 DC/DC converter for 5 and 12 V voltage outputs. [28]

In the two schematics of 3.10, the FB pin of the MAX17531 is connected to a resistor voltage divider between the output voltage and GND. Equation 3.1 describes the relation between the chosen resistors (R_1 and R_2) and the output voltage (V_{out}). It is recommended to chose a resistance value for R_2 between 25 kΩ and 100 kΩ and then calculate R_1 resistance value. [28]

$$R_1 = R_2 \left[\frac{V_{out}}{0.8} - 1 \right] \tag{3.1}$$

The -12 V configuration is slightly different. According to a Maxim Integrated application note [29], any step-down DC/DC converter can be used as an inverter with no changes to the operating schematic. The only difference between the normal step-down application, shown in figures 3.10a and 3.10b, and the inverting operation, shown in figure 3.11, are the connection pins. What used to be a V_{out} pin in a normal step-down application is a GND in the inverter application; and the GND is now a $-V_{out}$ in the inverter.

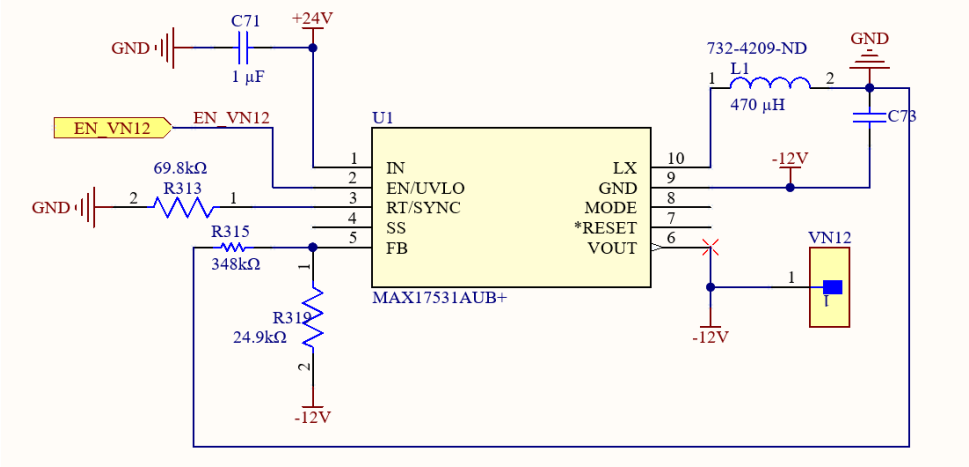


Figure 3.11: Implemented circuitry of the MAX17531 step-down DC/DC converter connected as a negative-output voltage inverter.

However, this configuration has some restrictions. The voltage difference between the input and negative output must be less than the maximum operating input voltage of the step-down DC-DC converter. For the desired purposed this is not a problem, since the step-down DC/DC converter used has a maximum input voltage of 42 V, therefore it can be used to convert 24 V to -12 V.

The use of bypass capacitors at the inputs and outputs of each component is recommended throughout the different datasheets since they provide a low impedance path to GND for high-frequency signals, which prevents noise or high current transients from being passed to other components through the common DC power connection. This ensures all power supplies stability and minimises their noise.

Digital Control

For safety reasons, it is intended to remotely turn on and off each of the power supplies generated by the DC/DC converters. To accomplish this, digital control circuits must be implemented following the specifications of each converter.

Since the chosen HV DC/DC converter lacks an enable/disable option, a hardware switch was placed at the 24 V primary supply voltage input to switch it on and off. However, the converter's output voltage can also be disabled by using a DAC, which will be controlled via SPI, or a potentiometer, to input 0 V signal into the converter's Remote Adjust Input pin, displayed in figure 3.9.

The three MAX17531 DC/DC converters, that will generate the 5 V, $+12$ V and -12 V @ 50 mA power supplies, can be enabled or disabled through its enable pins (EN/UVLO), shown in figures 3.10 and 3.11. This pin is an active-high input, which means that pulling it to GND disables the converter output. To enable the converter, a voltage superior to 1.25 V must be supplied to the EN/UVLO pin, which initiates a soft-start sequence whose duration depends on the connection made to the SS (Soft-Start) capacitor pin, which will be left unconnected for a default internal soft-start of 5.1 ms.

The distribution of the necessary control signals will be made by the output registers of two SPI-controlled 16-bit I/O port expanders (MCP23S17). Since the SPI control signals are shared between the HV Reading and the CT boards, all port expanders must have different hardware addresses. These addresses are defined by the A0, A1 and A2 port expander pins, which can be connected to either GND or to the port expander's supply voltage, which is 3.3 V, thus forming a 3-digit binary word. The first version of the HV Reading board has two port expanders, whose hardware addresses are 000 and 111. The second version will have only one port expander with 001 as its hardware address. Considering this, the chosen CT board port expanders' addresses are 010 and 101, as shown in figure 3.12.

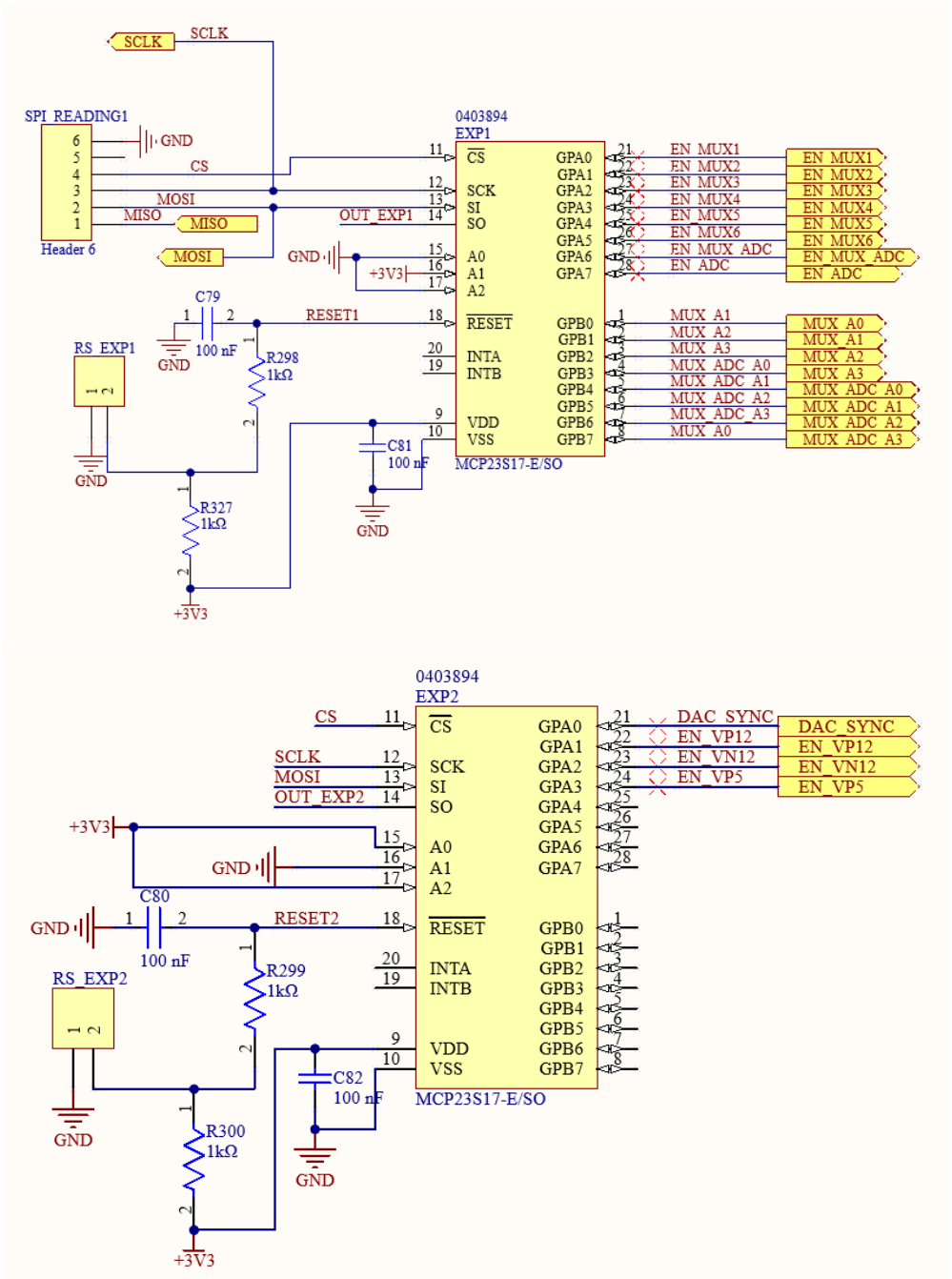


Figure 3.12: Cable Test Board port expanders.

The 96 HV and GND signals will be connected to six 16:1 analogue MUXs that will select one signal to be read at a time. The MUXs' outputs, the cable shield signals, the monitoring voltage of the HV DC/DC converter and the CERN setup cable continuity monitor signals will be connected to a final 16:1 analogue MUX, whose schematic is described in figure 3.13.

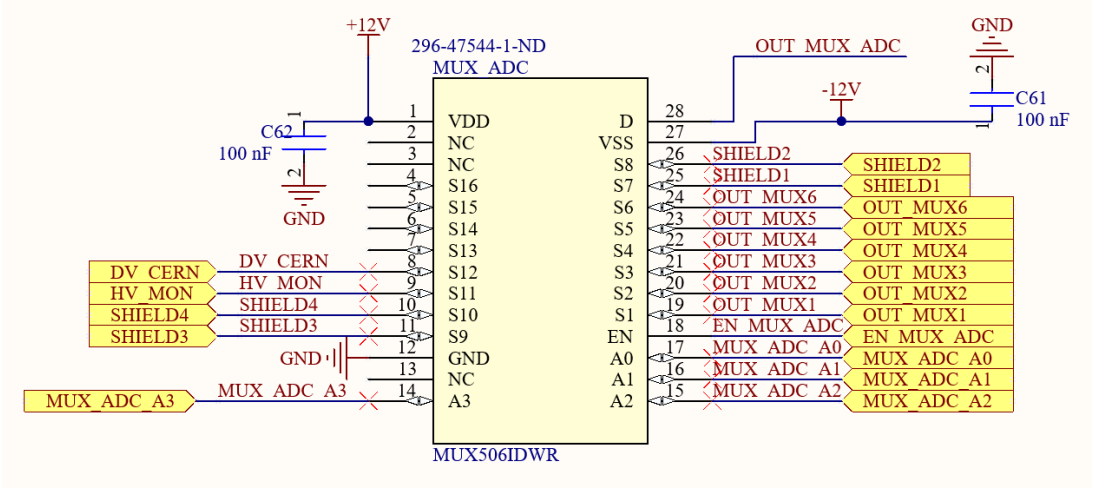


Figure 3.13: Schematic of the final 16:1 analogue MUX, responsible for selecting and outputting the analogue signal to be converted by the ADC (OUT_MUX_ADC).

The negative output of the final MUX is connected to an instrumentation amplifier (INA828) [30] configured to invert the negative signal so that it can be read by the ADC, and mitigate signal noise by subtracting the GND signal. The used ADC is the MAX1240, a 12-bit ADC with an internal voltage reference of 2.5 V. [22] A 3×3 pin header (figure 3.15) is connected to the ADC and the two output registers of each port expander so that the user can choose what to read via the MISO signal by connecting its pins with a jumper. The block diagram in figure 3.14 illustrates the digital control of the CT board.

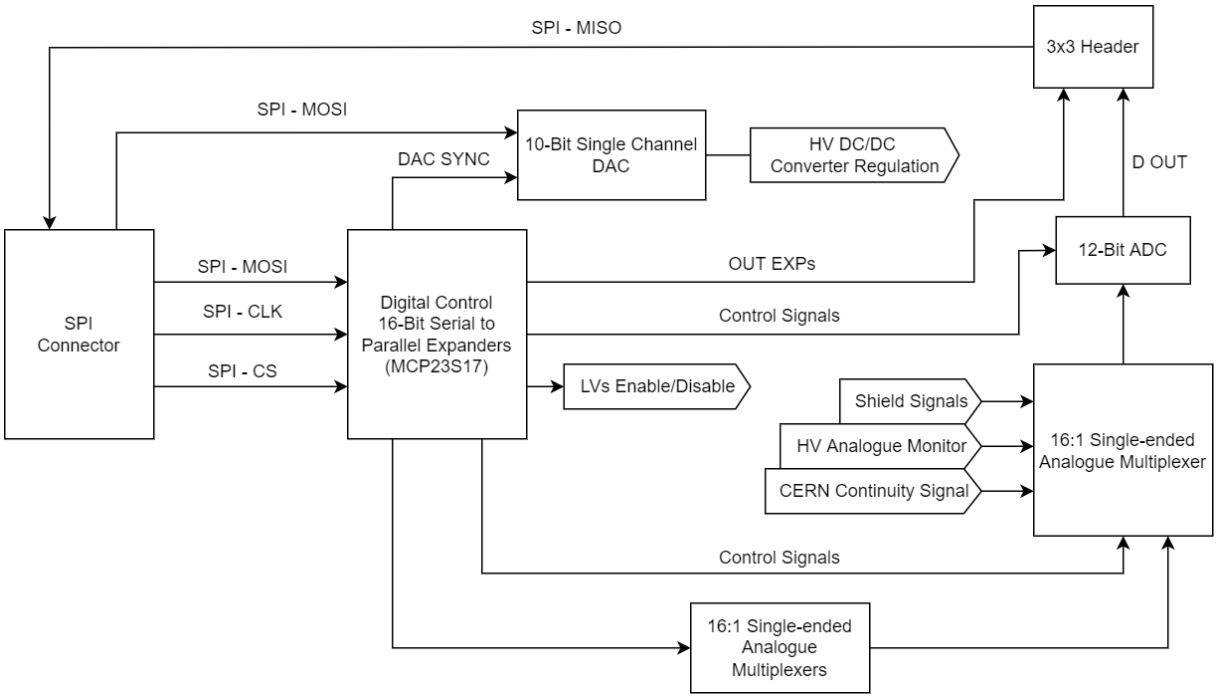


Figure 3.14: Block diagram of the Cable Test board digital control system.

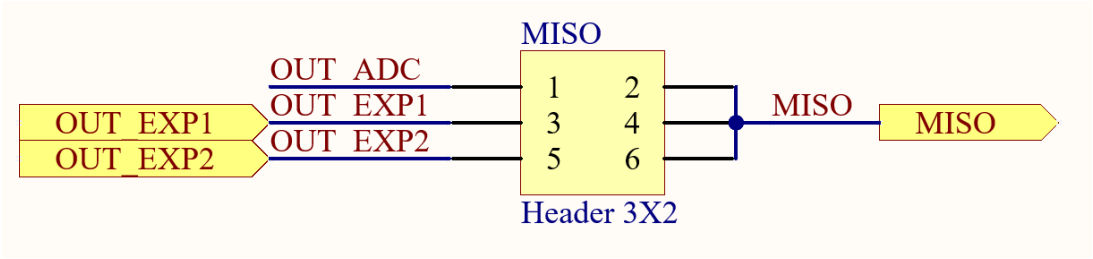


Figure 3.15: MISO 3×3 pin header.

To function properly, all digital inputs and outputs must be correctly set. To ensure this, each enable signal was connected to a a pull-down or pull-up circuit depending on the value set when the signal is not active. These circuits provide a defined default state for digital input and output pins, preventing them from floating between HIGH and LOW signal values when the digital control is not enabled or when it is started.

The controlling SPI signals will be provided by the HV Reading board through a connection cable, which will be, in turn, controlled by a Raspberry Pi. The boards will share the same SPI bus and have similar digital components, such as the I/O port expanders and the ADC. A software interface and a GUI were developed to control the CT board. Both developments will be described in the following chapter. All of the board's schematics and chosen components can be found in appendixes A and B, respectively.

3.3 Printed Circuit Board

Once the board's project design was complete, it was time to design the board's PCB, using the Altium Designer software (version 22) from Altium Limited. To ensure that the design works as intended, good design practices had to be learnt. According to Altium Limited [31], here are some of the design guidelines which have been followed:

1. **Design rule definition**: since there will be multiple HV signals, clearance and creepage distances must be defined beforehand to follow safety requirements and prevent component shifting and re-routing later on;

2. **Component grouping and placement**: components must be placed so the board can be easily routed, with as few layer transitions as possible. It is also important to respect mechanical constraints such as connector placement and enclosure limitations;

3. **Placement of power and ground planes**: establishing one ground plane is recommended. For power connections, it is recommended to use common rails in a signal plane, or a full plane. Creating daisy chain [1] power lines from part to part are not a good practice;

4. **Simple routing**: when possible, traces between components should be short and direct. If the trace routing on one side of the board must be horizontal, then the routing on the opposite side should be vertical.

[1] A daisy chain is a wiring scheme that connects several devices in series. A series of power strips, for example, plugged into one another to form a single long line of strips.

3.3.1 Design Rules

As the board operates at high voltage, PCB trace spacing requirements such as clearance and creepage must be defined carefully, to prevent signal integrity issues and arc discharge. When dealing with HV arcing is a real risk, since it can occur if the potential difference between two conductive elements exceeds the breakdown voltage of the board's insulator [2].

Clearance and creepage both describe the distance between two conductors through the air and along the surface of the insulation material, respectively. Figure 3.16 illustrates the difference between the parameters.

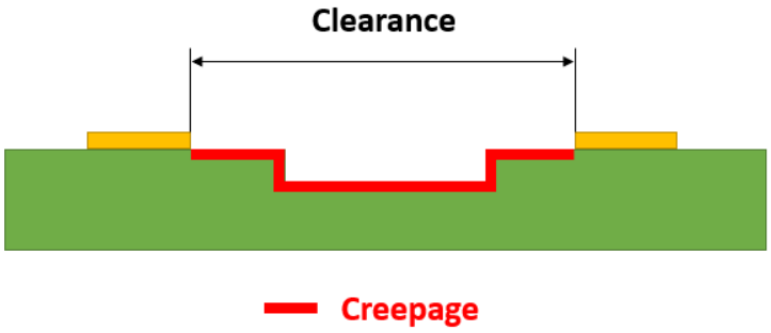


Figure 3.16: Illustration of creepage versus clearance in a PCB. [32]

Environmental conditions must be taken into account when defining trace spacing distances. For example, humidity increases the risk of arcing, since it changes the breakdown voltage of air; similarly, dust particles deposited on the PCB's surface can shorten the distance between conductors. [32] Creepage can be increased without changing any routing, by simply adding a slot between tracks, as shown in figure 3.16, or a vertical barrier of insulation.

However, clearance impacts routing, and choosing the appropriate trace spacing distances is not an easy task. Fortunately, there are well-established safety standards for PCB trace spacing distances such as the IPC-2221B, which is the most generic and used standard. After consulting it, a minimum clearance constraint of 0.76 mm (30 mils in imperial units, most commonly used in PCB design) between all of the board's HV traces was added.

3.3.2 Component Grouping and Placement

Component placement is a difficult task in PCB layout design as it requires creativity and meticulousness to create a board that can be easily routed while complying with the aforementioned design rules.

When placing components, it is important to always imagine how to route them and to leave enough room between parts, especially ICs, since they usually have a lot of pins to be connected. Furthermore, considerations must be made regarding the assembly and manufacturing process:

- Components should be placed in the same orientation to be easily assembled, inspected and tested; [33]
- Breakaway tabs should be included to allow easy gripping of the board while manufacturing;

[2] Minimum voltage that causes an insulator to become electrically conductive.

- All Surface Mount Device (SMD) components should be placed on the same layer to minimise manufacturing costs.

The first step in designing the PCB layout was to place the larger components, such as the DB25 connectors and DC/DC converters. Since the connectors had to be placed at one of the board's ends, the remaining large components were placed at the opposite end. This ensures that the heaviest components are evenly distributed across the board, preventing it from wobbling. The HV DC/DC converter was placed in one of the board's corners, thus being as far away as possible from the remaining circuitry, as it will dissipate a lot of heat on the board. At this point, it was decided to define the location of the different component groups throughout the board to facilitate further placing and routing, as described in figure 3.17.

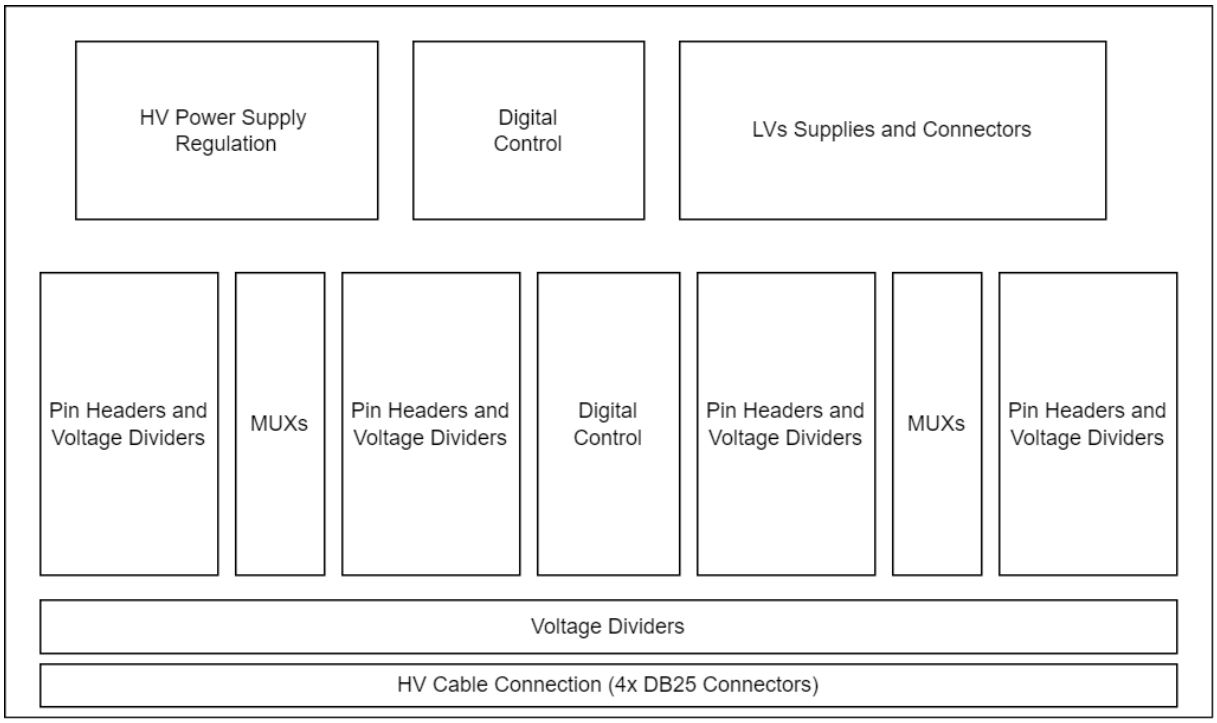


Figure 3.17: Cable Test board component grouping.

Pin headers and voltage divider resistors were placed near the DB25 connectors to allow an easier routing process. MUXs were placed between the respective pin headers groups, as they have a large number of connections and this simplifies routing. All of the digital components such as ADC, DAC, I/O port expanders and MUXs, as well as the instrumentation amplifier were placed as close as possible to each other as they share multiple signals. The power supply and SPI signal connectors were placed next to each other, between the digital components and the DC/DC converters to provide easy access to each of them.

Finally, decoupling capacitors were placed as close as possible to their respective components, as recommended by the components' manufacturers to reduce power supply noise significantly.

3.3.3 Placement of Power and Ground Planes

After completing the primary component placement, it was time to route power, ground, and signal traces. The final component placement will be determined during the routing process. The board has a total of 6 layers, as shown in figure 3.18, consisting of 4 signal layers and the top and bottom layers:

- Top layer: Component placement and analogue and digital signal routing;
- Layer 1: GND plane;
- Layer 2: HV connections and power; +5 V, +3.3 V, +12 V and +24 V power signals;
- Bottom layer: analogue and digital signal routing.

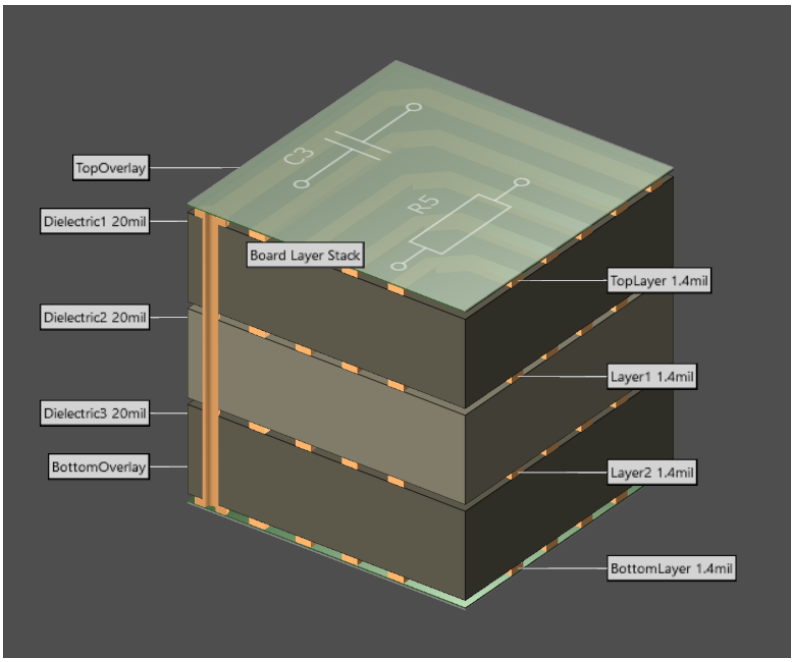


Figure 3.18: Cable Test board layer stack.

Internal planes in a multi-layer PCB should be separated appropriately and have a dedicated high voltage layer, so the current flows without interfering with the other components on the board. [34] With this in mind, the dielectric material thickness throughout the board was increased to 0.508 mm. The selected dielectric material is FR-4, a common material used in HV PCB manufacturing since it has a high dielectric profile. It has a Comparative Tracking Index (CTI) of 175, which is the maximum voltage at which a material withstands 50 drops of contaminated water without forming conductive paths. [35]

After establishing the materials and each layer, it was time to create the different signal polygons on layer 3, to which components will be connected through vias. Figure 3.19 describes the position of each polygon.

3.3.4 Signal Routing

The routing process started by defining the appropriate trace width. Even though this is a HV PCB layout, the current flowing through the board will not exceed 0.3 A, even the HV signals that have a current limit of 1 mA. Since there is not any specific impedance or high current specification and considering the standard trace thickness of 1 oz./sq. ft. copper (approximately 35 µm), a default 0.254 mm (10 mil, in imperial units) trace width is considered appropriate for routing the board's analogue and digital signals. [31] Some of the power supplies current can go up to 0.3 A, however, the connections between them and the components are made through planes and polygon net signals, so trace width is not an issue in these cases.

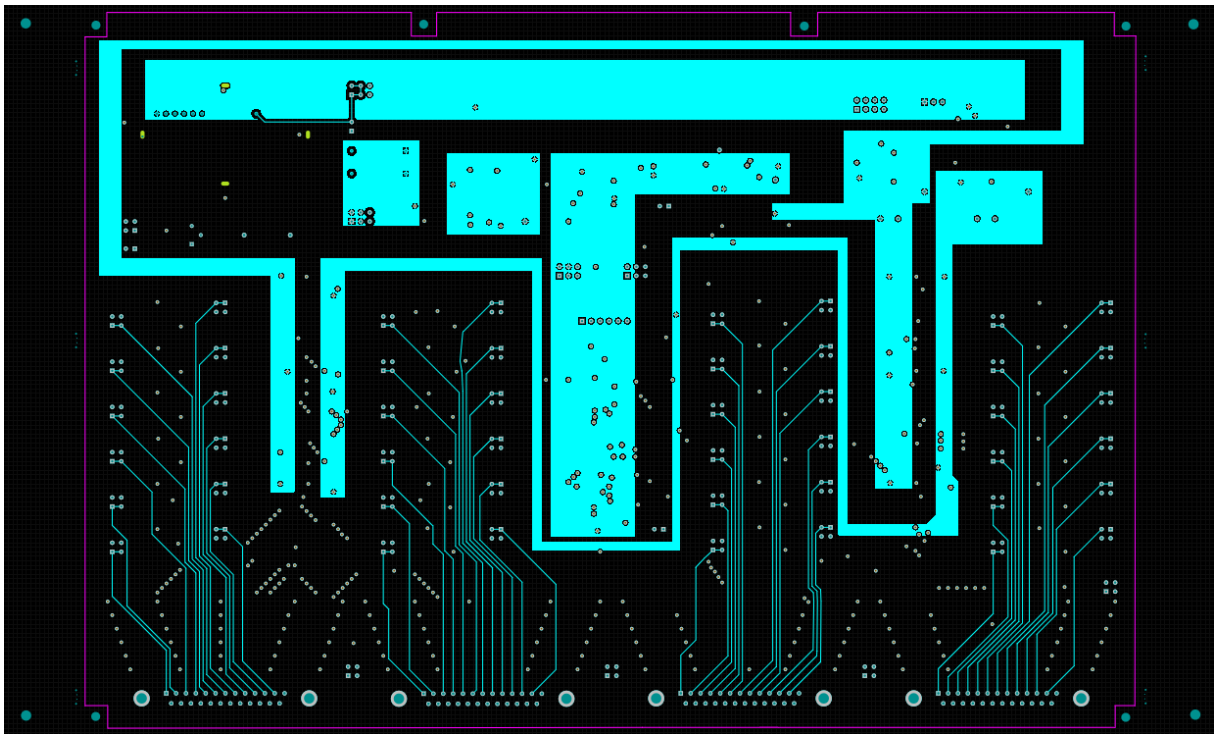


Figure 3.19: Power supply polygon distribution of the Cable Test Board.

After connecting every component to their respective net signals, all that was left was the routing of control and HV input signals. The routing strategy adopted for the control signals consisted of alternating horizontal and vertical traces in the top and bottom layers. However, there was not enough space in the top and bottom layers to route the HV signals while respecting the established clearance rules. They were then routed freely in layer 3, as shown in figure 3.19. Figures 3.20 and 3.21 describe, respectively, the final PCB layout and the rendered 3D preview of the Cable Test board.

With the project finished, it was time to validate all connections using the Electrical Rules Check (ERC) and Design Rules Check (DRC) tools offered by the Altium Designer software and correct any errors that arose. Once all the errors were corrected, all the necessary files for PCB assembly and manufacturing and the board's Bill of Materials (BOM) were generated by the Altium Designer software. The board is ready to be assembled and manufactured.

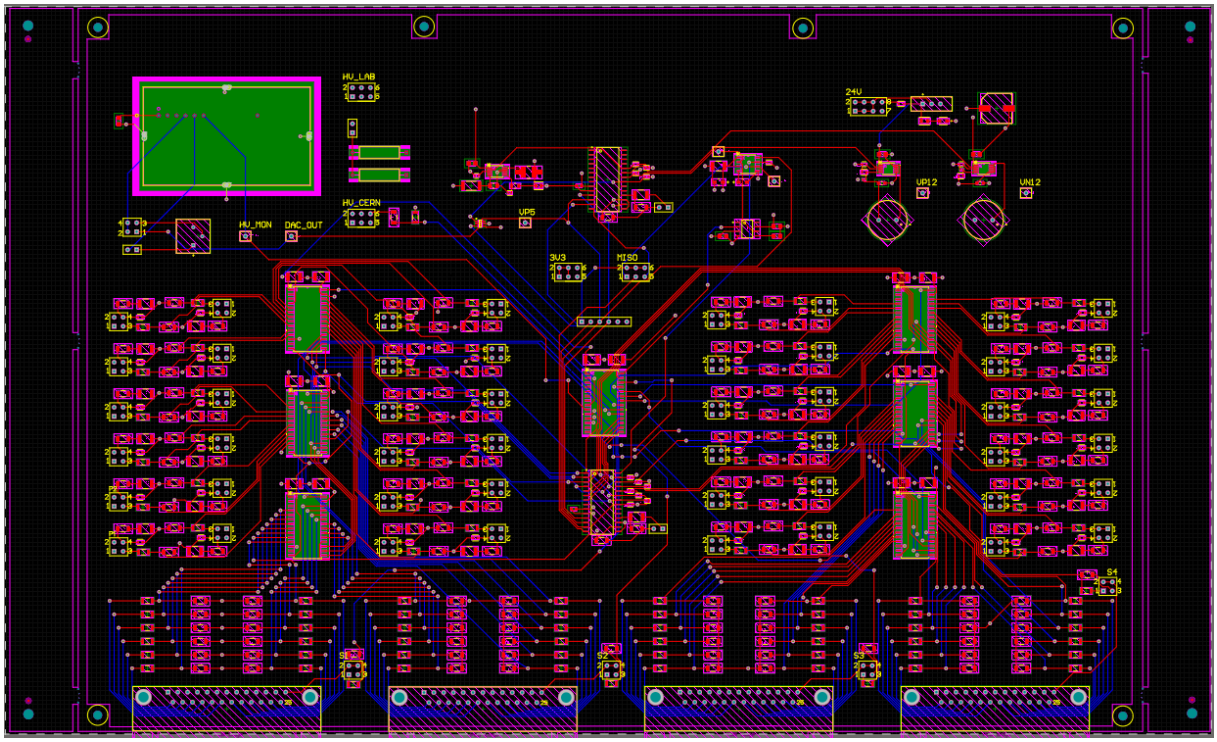


Figure 3.20: Cable Test Board final 2D PCB layout.

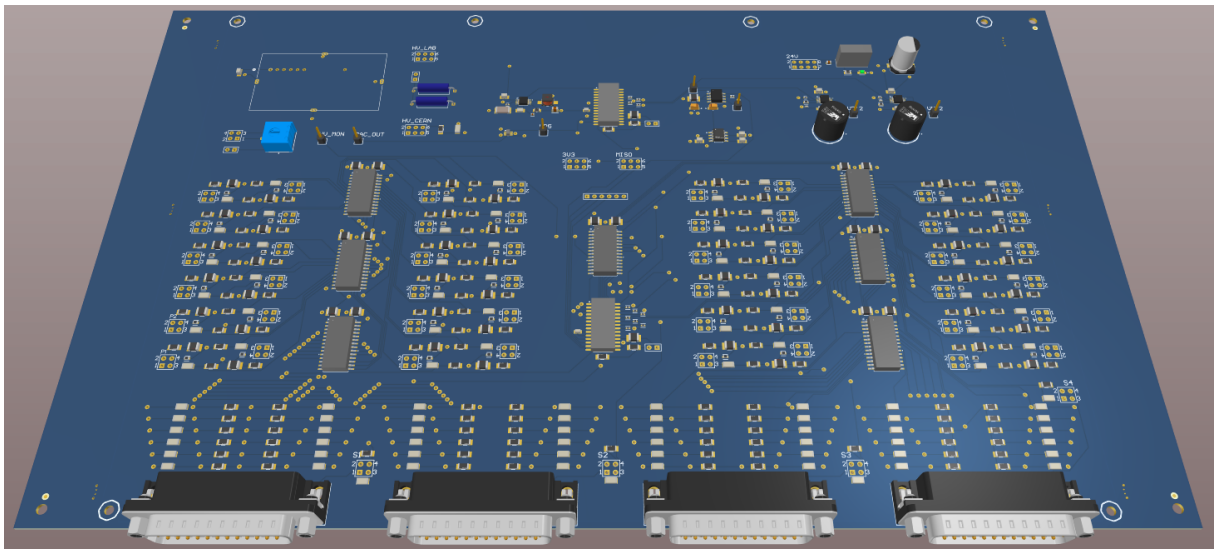


Figure 3.21: Cable Test Board PCB 3D preview.

Chapter 4

Cable Test Board Digital Control System and Graphical User Interface

This chapter describes the development of the dedicated digital control for the Cable Test (CT) board, and its GUI. It will allow the user to select each cable, by controlling the HV Reading board relays, control the input test HV and perform voltage readings of up to 96 cables.

The digital control of the HV Reading board was already written by Rui Marques [2], as part of the HV Remote functional tester software. As such, some of its classes were imported and used to develop the CT board software interface, since it was written using Object-Oriented Programming (OOP) and the boards share some integrated components, such as the 16-bit I/O port expanders and the ADC. This work will be briefly described in this chapter as well.

The digital control system for the CT board was written in Python 3 language and was developed to work with a Raspberry Pi 3 Model B+, which is already used to control the HV Remote system in preliminary tests and was, therefore, a familiar device to work with. Communication between the Raspberry Pi and the boards will be made through SPI protocol. The dedicated GUI was developed using Qt Designer and is also written in Python 3.

4.1 Raspberry Pi 3 Model B+ and SpiDev Module

The Raspberry Pi is a low-cost, small computer that can be connected to a computer monitor or television, through its HDMI video output, and operates with a standard keyboard and mouse via Universal Serial Bus (USB). Its operating system is the Raspbian OS and it is stored on a micro SD card. The model 3 B+ has 40 pins, 27 of which are GPIO pins. It also has several peripherals and the SPI peripheral is one of them.

The SPI peripheral enables the Raspberry to function as either a slave or a master. It will be used as a master in this application, implementing a common SPI protocol with two independent CS signals (CE0 and CE1). Figure 4.1 shows how the distribution of the different pins.

Each of the previously mentioned SPI pins will be directly connected to the board and controlled by a software interface that was developed in Python 3 with object-oriented programming. The Python language allows access to several useful libraries, one of which is SpiDev [36], which was developed to be used specifically to establish SPI transactions with a Raspberry Pi. It contains several methods and attributes, the most important of which are:

- **open(bus, cs)**: Method. Connects an object to a specified SPI device. Takes as arguments two

Figure 4.1: Raspberry Pi 3 B+ pinout. In green are the GPIO pins, in pink the SPI pins, in blue the inter-integrated circuit pins, in lilac the Universal Asynchronous Receiver/Transmitter pins and in turquoise the Pulse Code Modulation pins.

integers, bus and cs;

- **close()**: Method. Disconnects an object from the system SPI device;
- **max_speed_hz**: Attribute. Specifies the maximum bus speed in Hz. The default value is 125 MHz, although the maximum speed on the Raspberry Pi is 32 MHz;
- **mode**: Attribute. Sets the SPI mode and, therefore, CPOL and CPHA. Its default value is 0, but can take integer arguments between 0 and 3 (modes 0 to 3);
- **xfer2([values])**: Method. Performs a SPI transaction. Takes as argument a list of bytes (8-bit words) that are written to the SPI device and, as each byte in that list is sent out, the chip select signal is held active (lowered).

4.2 Development of the Cable Test Board Digital Control

It is necessary to identify the devices to be digitally controlled and detail their specifications before developing a digital control system. For the CT board, these devices are the MCP23S17 [19] I/O port expander, the DAC6311 [37] 10-bit DAC, the MAX1240 [22] 12-bit ADC, the MUX506 [38] analogue MUXs and the MAX17531 [28] low voltage DC/DC converters.

The only digitally controlled feature of the MAX17531 DC/DC converters is the enable pin, which is an active high pin that will be connected to one port expander output pin. The MCP23S17 I/O port expander and the MAX1240 ADC are well-known and used components in both the HV Remote and HV Reading boards and were already described in 3. Filipe Cuim [16] has already created classes for these components. Therefore, this section will not include a detailed description of those components and their implemented classes. Methods from these classes were incorporated into the classes that were developed from scratch and will be described further on: the **Cable Test** class, which includes all the board's command methods; and the **DAC6311** class that will control the single-channel DAC that regulates the HV DC/DC converter.

4.2.1 The Digital to Analogue Converter

The chosen DAC was the DAC6311, a 10-bit low-power, single-channel DAC. [37] This device has a 3-wire serial interface with a maximum clock rate of 50 MHz, which is compatible with standard SPI protocol. Figure 4.2 shows the component's implemented circuitry.

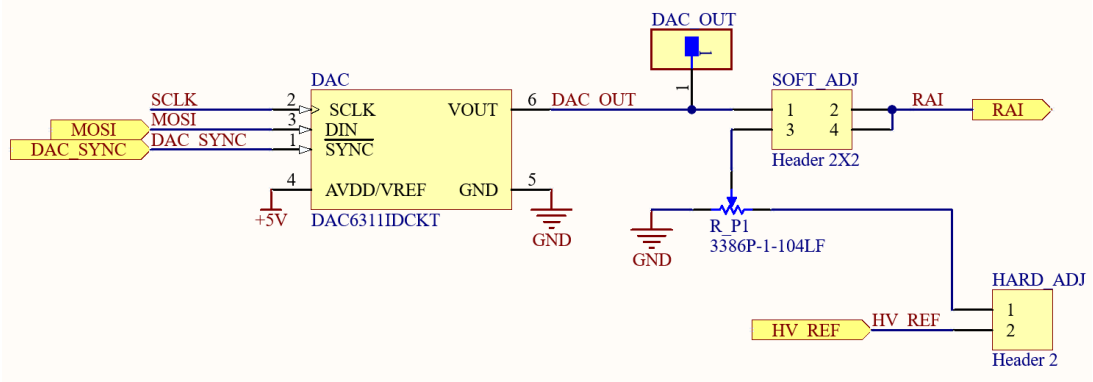


Figure 4.2: Implemented circuitry of the DAC6311. [37]

The signals labelled $\overline{\text{SYNC}}$, SCLK and D_{IN} are the serial interface pins. $\overline{\text{SYNC}}$ is a level control input signal which is active low and it will be connected to a dedicated signal, DAC_SINC, provided by one output signal of one of the port expanders of figure 3.12. The SCLK pin will be connected to the SPI SCLK signal and the D_{IN} pin to the SPI MOSI signal.

The V_{REF} signal, is the power supply input and acts as an external reference voltage, which determines the DAC's maximum output voltage. It will be supplied with 5 V. The DAC6311 ideal output voltage is given by equation 4.1, where N is the number of bits, which is 10, D is the decimal equivalent of the binary code received through D_{IN} and V_{REF} is the input power supply and the reference voltage.

$$V_{OUT} = V_{REF} \times \frac{D}{2^N} \tag{4.1}$$

This DAC also features a power-on reset. When it is initialised, its input register is filled with zeros, and the output voltage is 0 V. The input register remains in this state until a valid write sequence to the DAC is completed.

Furthermore, the DAC6311 has a set of programmable power-down modes that regulate its operation and consumption. The modes are defined by two bits of the control register (PD1 and PD0). Table 4.1 describes the different modes and establishes the correspondence between each of them and the bits of the control register.

PD1	PD0	Operating Mode
0	0	Normal Operation
0	1	Power-down 1: Output 1 kΩ to GND
1	0	Power-down 2: Output 100 kΩ to GND
1	1	Power-down 3: High-Z

Table 4.1: Modes of operation of the DAC6311.

In normal operation, the device has a power consumption of approximately 80 µA @ 2 V. All three power-down modes reduce the typical supply current to 0.1 µA @ 2V; the difference between them lies in the device's output impedance, as described in the table above.

The control bits that select between these modes are included in the DAC's input shift register, described in figure 4.3. The register is 16-bit wide and consists of one 2-bit control word (PD1 and PD0) and 14 data bits which comprise the 10-bit input data and the so-called *don't care* bits, which are discarded.

DB15	DB14										DB4	DB3			DB0
PD1	PD0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	X	X	X	X

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Figure 4.3: DAC6311 10-bit data input register; PD1 and PD0 (DB15 and DB14) are the control bits, D9 to D0 (DB13 to DB4) are the 10 data bits and the remaining bits (DB3 to DB0) are the discarded bits. [37]

The write sequence starts by lowering the $\overline{\text{SYNC}}$ line. On each falling edge of SCLK, data is clocked into the 16-bit shift register through the D_{IN} line. The programmed function is executed once the last data bit is clocked. At this point, the $\overline{\text{SYNC}}$ must be raised for at least 20 ns before the next write sequence so that the next write sequence is initiated by the falling edge of $\overline{\text{SYNC}}$.

In case the $\overline{\text{SYNC}}$ line is raised before the SCLK 16th falling edge, the shift register is reset and the write sequence is deemed invalid. Therefore, the $\overline{\text{SYNC}}$ signal is a write sequence reset. This feature does not update the content of the DAC register or change the operation mode. Figure 4.4 compares a valid and an invalid write sequence.

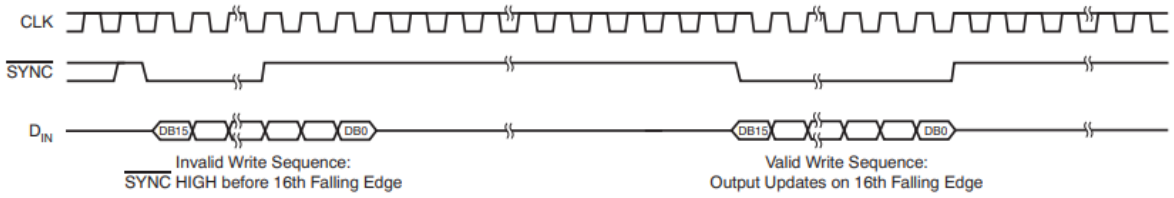


Figure 4.4: DAC6311 $\overline{\text{SYNC}}$ interrupt feature. [37]

The DAC6311 Class

The DAC6311 class was written from the ground up and allows the user to write a valid sequence in the DAC6311 input shift register and select the desired operating mode from the available power-down modes described in table 4.1. The class requires three arguments to be initialised: a SpiDev instance, the number of the port expander pin that will control the DAC $\overline{\text{SYNC}}$ signal, and an MCP23S17 instance representing the port expander that controls this pin. A dictionary of constant values was also defined to correspond each operation mode to an integer. Figure 4.5 shows the DAC6311 class constructor describing the aforementioned arguments and dictionary.

An additional attribute "spiMode" was created to define the SPI clock mode, which must be set to mode 1 (CPOL = 0 & CPHA = 1) so that data is clocked into the input shift register on the falling edge of the SCLK.

Aside from the constructor, the class has two more methods: **writeRegister** and **sendWord**. The **writeRegister** method, depicted in figure 4.6, accepts two integers as arguments (*op_mode* and *data*). The DAC's operating mode is determined by the *op_mode* argument, and *data* is the 10-bit data. Both arguments are to be written into the input shift register.

The **writeRegister** method takes the *op_mode* integer and matches it to a 2-bit word from the operation modes dictionary described at the beginning of this subsection and saves that value as the variable

```
class DAC6311(object):
    """
    This class is an abstraction of the DAC6311, which is a 10-Bit Single Channel DAC, for the
    Raspberry Pi. It requires the Python modules spidev and RPi.GPIO.
    """
    OP_MODES = {
        '0' : 0b00,
        '1' : 0b01,
        '2' : 0b10,
        '3' : 0b11
    }

    def __init__(self, spiDevice = spidev.SpiDev(), SYNC = -1, MCP = MCP23S17)
        """
        Constructor

        Keyword Arguments:
        SYNC -- pin of the SYNC -- pin of the MCP23S17 that will be used as SYNC.
                The write sequence starts by bringing SYNC low
        MCP -- the port expander that will control the SYNC pin of the DAC
        """

        self.__spi = spiDevice
        self.__spiMode = 0b01
        self.__SYNC = SYNC
        self.__MCP = MCP
```

Figure 4.5: DAC6311 class constructor, where spiDevice is the SpiDev instance, MCP is the port expander (MCP23S17) instance and SYNC is the port expander's pin number controlling the $\overline{\text{SYNC}}$ signal.

```
def writeRegister(self, op_mode, data): #alterar
    """
    Write into the input shift register

    Keyword Arguments:
    data -- 10-bit data to be written in the input register.
            Must be an integer less than 1024
    op_mode -- integer between 0 and 3 that determines the DAC's mode
                of operation. 0 is normal operation, 1-3 are power-down modes.
    """

    assert type(data) is int
    assert type(op_mode) is int
    assert op_mode in range(0,4)
    assert data < 1024


    prefix = DAC6311.OP_MODES[op_mode]
    feature = 0b0000

    self.__sendWord(prefix, data, feature)
```

Figure 4.6: The **writeRegister** method, which edits and sends the data to be written into the DAC6311 input shift register.

prefix. A new variable *feature* was created to store the ”don’t care” bits of the input shift register. The three variables are then transferred to the input shift register via the **sendWord** method.

Finally, the **sendWord** method, described in figure 4.7, performs the SPI transaction. It takes as arguments three variables, *prefix*, *data* and *feature*, combines and divides them into two 8-bit words and sends them to the DAC in their correct order, forming a 16-bit word. Before the word is sent, the DAC $\overline{\text{SYNC}}$ signal is activated by lowering its respective port expander pin. The SpiDev function **xfer2** then transmits the two words without lifting the $\overline{\text{CS}}$ line. Then the $\overline{\text{SYNC}}$ signal is deactivated by lowering the port expander pin. Figure 4.7 describes the method.

```
def __sendWord(self, prefix, data, feature):
    """
    Enables communication with the DAC by lowering the SYNC line,
    sends the word to the DAC via spidev xfer2 function which takes a list
    of bytes.

    Keyword Arguments:
    prefix -- 2-bit word that determines de mode of operation.
    data -- 10-bit data word to be sent
    feature -- 4-feature bits which are to be ignored
    """
    data1 = data >> 4
    first_word = (prefix << 6) | data1
    second_word = ((data & 0b00001111) << 4) | feature

    self.__MCP.digitalWrite(self.__SYNC, self.__MCP.LEVEL_LOW)
    self.__spi.mode = self.__spiMode
    self.__spi.xfer2([first_word, second_word])
    self.__MCP.digitalWrite(self.__SYNC, self.__MCP.LEVEL_HIGH)
```

Figure 4.7: The ”sendWord” method, responsible for performing the SPI transactions.

4.2.2 The Analogue Multiplexer

For analogue multiplexing, the MUX506 [38] was chosen. It has 16 input ports and one output port. It has four selector signals and works with a single supply of 10 V to 36 V or with dual supply between $\pm$ 5 V to $\pm$ 18 V. The device performs well with either symmetric or asymmetric dual supply.

This MUX will be powered by a dual $\pm$ 12 V power supply in this application, as shown in figure 3.13. If the MUX had to be used with a single power supply, connecting the V_{SS} to GND would suffice. The enable pin EN is an active high pin. When activated, the analogue inputs can be selected according to the truth table 4.2.

4.2.3 The CableTest Class

The **CableTest** class incorporates elements from the DAC6311 class, described in the previous subsection, and the MCP23S17, MAX1240 and HV Reading classes. It also imports the CTBMap, which contains the constant default values of the port expander’s pins and the dictionaries corresponding the HV, GND and shield channels to their MUXs and MUX addresses. This map file and its composition is described in subsection 4.2.3.1. To be initialised, the class requires four arguments: a SpiDev instance, a RPI.GPIO instance (python library that allows the control of the Raspberry Pi GPIO pins), the number of the Raspberry Pi pin that will act as a $\overline{\text{CS}}$ signal, and a number, 1 or 2, which indicates the HV Reading board version used. Figure 4.8 shows the CableTest class constructor with the aforementioned arguments.

EN	A3	A2	A1	A0	ON-CHANNEL	SIGNAL
0	X	X	X	X	All off	-
1	0	0	0	0	S1	OUT_MUX1
1	0	0	0	1	S2	OUT_MUX2
1	0	0	1	0	S3	OUT_MUX3
1	0	0	1	1	S4	OUT_MUX4
1	0	1	0	0	S5	OUT_MUX5
1	0	1	0	1	S6	OUT_MUX6
1	0	1	1	0	S7	SHIELD1
1	0	1	1	1	S8	SHIELD2
1	1	0	0	0	S9	SHIELD3
1	1	0	0	1	S10	SHIELD4
1	1	0	1	0	S11	HV_MON
1	1	0	1	1	S12	DV_CERN
1	1	1	0	0	S13	-
1	1	1	0	1	S14	-
1	1	1	1	0	S15	-
1	1	1	1	1	S16	-

Table 4.2: Example of a truth table, based on the configuration of the MUX described in figure 3.13.

```
class CABLETEST(object):

    def __init__(self, spiDevice = spidev.SpiDev(), iGPIO = GPIO, CS_PIN = 40, board = 1):

        self.__spi = spiDevice
        self.__GPIO = iGPIO
        self.__CS_PIN = CS_PIN
        self.__GPIO.setup(self.__CS_PIN, self.__GPIO.OUT)
        self.__HVRBoard = board
        self.__MCP = dict()
```

Figure 4.8: CableTest class constructor, where spiDevice is the SpiDev instance, GPIO is the RPI.GPIO instance and CS_PIN is the Raspberry Pi's GPIO pin number controlling the $\overline{\text{CS}}$ signal.

The constructor creates objects for each argument given, defines the selected CS pin as an output pin and creates an empty dictionary called MCP, where different instances of the MCP23S17 port expander will be included. The HV Reading board version must be defined as it is a necessary argument to initialise an HV Reading class instance. The CableTest class methods are described below and all the corresponding code can be found in appendix C.

CableTest class methods:

- **initBoard(self)**: Initialises the port expanders, ADC, DAC and HV Reading board;
- **enableCommunication(self, onOff)**: Enables/disables board communication. Keyword Arguments: **onOff** (boolean), if true communication is enabled, if false communication is disabled;
- **enableDisableSupplies(self, supply, onOff)**: Enables or disables one power supply. Keyword Arguments: **supply** (string), string with the name of the power supply; **onOff** (boolean), if true supply is enabled, if false supply is disabled;
- **readHVChannel(self, channel)**: Performs a reading of a specific HV channel. Keyword Arguments: **channel**, integer between 1 and 48. Returns: **value** (int), 12-bit integer reading from the ADC;
- **readGNDChannel(self, channel)**: Performs one reading of a specific GND channel. Keyword Arguments: **channel**, integer between 1 and 48. Returns: **value** (int), 12-bit integer reading from the ADC;

- **readSupplyHV(self)**: Performs a reading of the output voltage of the HV DC/DC Converter. Returns: **value** (int), 12-bit integer reading from the ADC;
- **readShield(self, shield_num)**: Performs a reading of a specific Shield channel. Keyword Arguments: **channel**, integer between 1 and 4. Returns: **value** (int), 12-bit integer reading from the ADC;
- **readCERN(self)**: Performs a continuity reading when the board is set to CERN mode via hardware. Returns: **value** (int), 12-bit integer reading from the ADC;
- **readAllHVs(self)**: Performs a reading of all HV channels. Returns: **[values]**, list of 48 reading from the ADC;
- **readAllGNDs(self)**: Performs a reading of all GND channels. Returns: **[values]**, list of 48 reading from the ADC;
- **setHV(self, hv)**: Adjusts the high voltage output of the DC/DC converter. Keyword Arguments: hv (int), a value between 0 and 1500 corresponding to the voltage in Volts (max. 1500 V);
- **setHVCable(self, cable)**: Selects the cable that transmits the HV, by controlling the HV Reading relays;
- **set_spiSpeed(self, speed)**: Sets the SPI communication speed (max. 32 MHz);
- **closeBoard(self)**: Disables the DAC and all low voltages.

4.2.3.1 Cable Test Board Map

A script, called Cable Test Board Map, in short, CTBMap, was created to simplify the use of channel variables, by corresponding each channel to its MUX pin. This correspondence is made through multiple dictionaries which correspond each channel to a list containing three variables: *MUX enable*, *MUX address* and *MUX output*.

The MUX enable is a 6-bit word to be written in the GPIO register of the port expander, and each bit corresponds to one MUX enable signal. The MUX address is a 4-bit word and the selector signal for the analogue MUXs that selects the cables to be measured. It is defined according to the MUX506 truth table and the Cable Test Board schematics, which can be found in appendix A. Finally, the MUX output variable is another 4-bit selector signal for the final analogue multiplexer, whose truth table is presented in table 4.2. The CTBMap can be found in appendix C.

4.3 Cable Test Graphical User Interface

Whenever a user wants to control a specific part of the board, they must send a specific digital command. Since the Cable Test board has multiple features, sending these commands repetitively becomes a slow and tedious process. To facilitate this feature selection process, a GUI was developed using the Python 3 programming language and the Qt Designer tool.

The Qt Designer tool allows the user to add and build GUIs using Qt Widgets, a catalogue of User Interface (UI) elements designed to create classic desktop-style UIs. This tool employs a what-you-see-is-what-you-get approach to assist in the creation and customisation of windows, and enables window testing with various styles and resolutions. [39]

The process began with the identification of the necessary visual features, namely the enable/disable of each DC/DC converter, the selection of the HV to be supplied to the cables, and the options for reading the voltages related to each cable's properties (HV, GND and shield).

After identifying the necessary features, a first layout of the graphical interface was made using the Qt Designer tool, which can be seen in figure 4.9. In this first phase, it was not necessary to write code to generate the observed visual aspects, as they are customisable widgets that you can simply drag and drop.

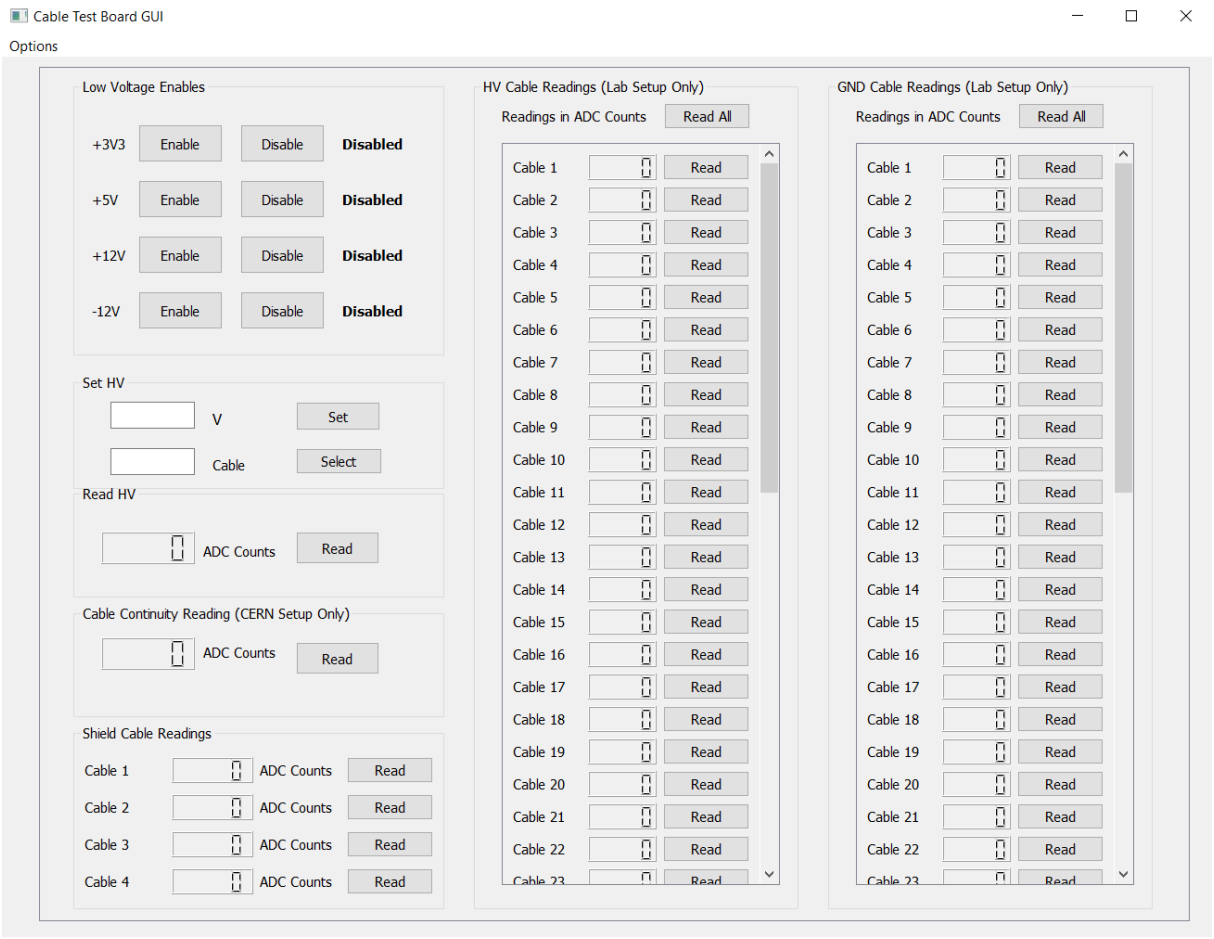


Figure 4.9: Cable Test Board GUI designed using the Qt Designer Tool.

All readings will be displayed in ADC counts, that is, a number between 0 and 4096 (2^{12} as it is a 12-bit ADC) will be displayed in each of the LCD number widgets.

Once the design was finished, the project was saved in a .ui extension file that was later converted into a python file using the pyuic5 tool. The generated python file contains a **Ui_MainWindow** class that aggregates all widgets and their properties in a single function, **setupUi**. This code must be copied and pasted onto another file so that further changes are not lost whenever the graphical design is altered.

Then, a link between each of the widgets and the functions of the Cable Test Board must be established. This is done by importing classes such as time, SpiDev, RPI.GPIO and CableTest. Most of the interactive widgets used are buttons and it is required that, by pressing each of them, a digital control system response is triggered. This response is achieved by the use of the *clicked* attribute and the *connect* method of the PyQt5 library, as exemplified in figure 4.10.

```
self.set_cable.clicked.connect(lambda: self.__setCable(self.textEdit_setCable.toPlainText()))

def __setCable(self, cable):
    self.__ctbBoard.setHVCable(cable)
```

Figure 4.10: Exemplification of the combined use of the clicked attribute and connect method of the PyQt5 library.

The *connect* method takes as an argument a lambda function, a small anonymous function that can take any number of arguments, but can only have one expression. Its expression is another function, defined within the **Ui_MainWindow** class which is to be called when the corresponding button is clicked. In the example featured in figure 4.10, the *set_cable* button will, when clicked, trigger the **Ui_MainWindow** method **__setCable** which takes as an argument the number of the cable to be selected, retrieved from the text written by the user in a text prompt.

The board must be initialised by selecting the *Options* tab, seen in figure 4.11, and then selecting the *Initialize Board* option. This will trigger a message box requesting the user to select the Raspberry Pi pin number which will provide the board with the CS signal, as shown in figure 4.12. It is followed by another message box requesting the selection of the HV Reading board version to be used, as displayed in figure 4.13.

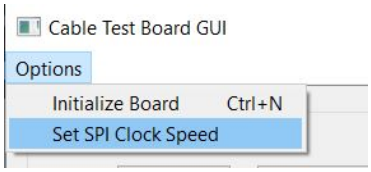


Figure 4.11: GUI Options tab.

There is also a *Set SPI Clock Speed* option, that displays a message box requesting the user to enter the desired SPI clock speed or frequency in kHz.

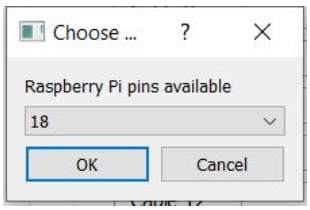


Figure 4.12: Raspberry Pi pin selection.

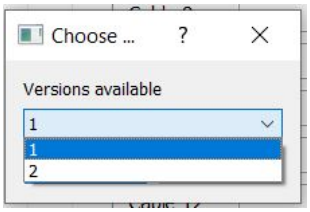


Figure 4.13: HV Reading board version selection.

Once the board is initialised, the user will be able to enable each of the low voltages, set the desired HV, select the cable to be powered and perform voltage and continuity readings, by editing the text prompts and clicking the buttons shown in figure 4.9.

The GUI was tested with simulated values and has been successful. However, this interface must be integrated with the Cable Test system to be effectively tested, which can only be done once the Cable Test board is manufactured. Also, the option of continuous data acquisition is missing from this version and will have to be implemented in the future to allow a more in-depth analysis of cable performance. The complete GUI code development is described in appendix D.

This chapter completes the description of the Cable Test system. The next chapter will describe the tests performed on the HV Remote board and characterise its performance.

Chapter 5

HV Remote Performance Tests

This chapter will discuss the performance tests carried out with the HV Remote board's second prototype, which is the last task of this work. Two boards were produced and tested in a laboratory environment at LIP and at CERN's Test Beam facilities.

When the boards arrived, they were connected to the HV Supplies board via an interface board shown in figure 5.1, since the designed crate was not yet operational, and could not be used.

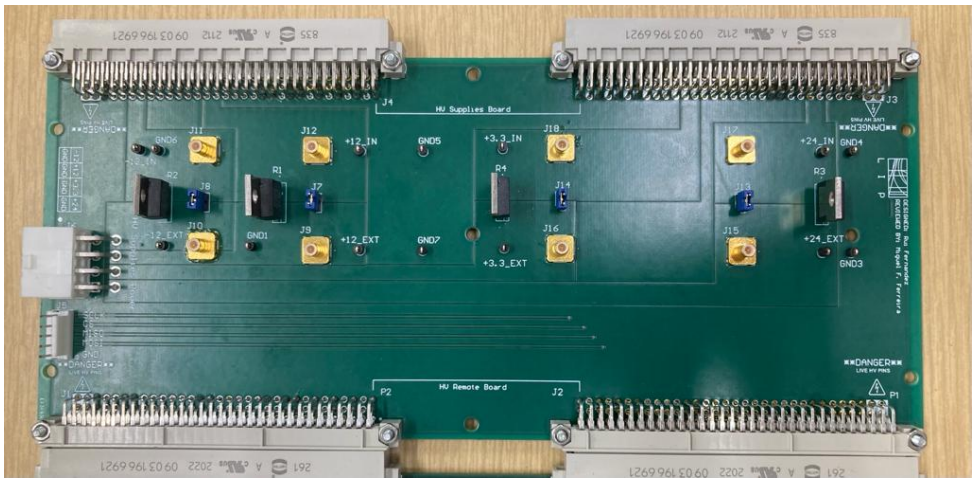


Figure 5.1: Interface board, which receives and distributes to the HV Remote board the SPI signals and HV supplies.

During the first functional tests, it was discovered that the on/off circuitry had significant design and PCB assembly errors. As a result, this functionality was rendered inoperable, and both boards can only be used with all HV channels enabled. However, a significant set of tests remained viable and will be described extensively in this chapter, including:

- Sequential tests and calibration at LIP;
- Stability tests at CERN Test Beam facility;
- Stability tests at LIP;
- Response tests to different types of load (without load, resistive and active loads, and PMT).

Throughout this chapter, and to simplify future calculations and comparisons, the HV values used for calibrations, calculations and graphs correspond to the absolute voltage value. All the HV values are negative.

5.1 Sequential Tests and Calibration

Before sending one of the boards to the CERN Test Beam facility, each of its channels was calibrated by monitoring the board's output HV values, using the board's monitoring functionality, also referred to in this text as ADC measurements, and an external voltmeter connected to an active filter. This calibration test consisted of setting different output voltage values through the board's regulation loops and taking multiple measurements of the output HV values over a time period, and then performing a linear regression to obtain the calibration and offset parameters of each channel. Voltages were set in the HV Remote GUI as DAC count values and two values were tested: 2500 and 3000 counts. Each channel was subjected to approximately 15-minute tests and the HV Supplies board was set to supply the HV Remote board with a primary voltage of 800 V. It is important to note that the HV Supplies board has a known offset of about 20 V. As a result, it applies a voltage of 780 V to the regulation loop circuit (HV_IN in figure 2.13 of chapter 2), instead of the requested 800 V.

Table E.1 in appendix E displays the mean values of the ADC and voltmeter measurements for each channel, as well as the calculated calibration and offset factors. These factors were calculated by comparing measurements made with the ADC and the voltmeter, as well as values commanded with the DAC and measurements made with the voltmeter, and then applying linear regression to each of the correspondences.

It was discovered that the board's channel 28 does not respond to the given commands. The same voltage value is displayed for both previously indicated input values, which indicates that the HV regulation system of this channel is not functioning properly. Channel 28 has a constant voltage of 420 V, which corresponds to HV_IN -360 V, the limit value that the regulation circuit imposes, as mentioned in chapter 2, subsection 2.1.4.

Channel 43 revealed itself as a problematic channel as well, as it did not respond immediately to the given input values, presenting in the first few seconds a voltage about 280 V above the value set. It then stabilised into the requested value, but this initial instability affected the calibration factors.

In general, with the exception of channel 28, the channels have similar calibration factors, namely 0.25 V per ADC count and 0.24 V per DAC count. Channels differ most in their offset factors.

Since all linear regressions were performed with only two points for each channel, it was decided to measure the response of two arbitrary channels to more input values. Channels 4 and 45, as well as extra DAC count values 2250, 2750, and 3250, were selected. Tables 5.1 and 5.2 display the average ADC and voltmeter values for each of the HV channels and input DAC values.

Ch 4		
DAC	ADC	Volt
2250	2193.399	543.832
2500	2435.667	604.152
2750	2677.739	664.519
3000	2919.160	724.832
3250	3160.795	785.139

Table 5.1: Mean values of the ADC and voltmeter measurements of HV channel 4, for different input DAC values.

Ch 45		
DAC	ADC	Volt
2250	2186.184	546.061
2500	2427.925	606.642
2750	2669.554	667.266
3000	2910.354	727.857
3250	3151.193	788.412

Table 5.2: Mean values of the ADC and voltmeter measurements of HV channel 45, for different input DAC values.

The linear regressions applied to data obtained, as shown in tables 5.3 and 5.4, suggest that the two-point linear regressions made for all the other channels are sufficient to calculate a calibration factor. Both

V as a function of DAC counts					
Ch 4	Calibration factor	Offset	Ch 45	Calibration factor	Offset
2 pt regression	0.2414	0.7552	2 pt regression	0.2424	0.5636
5 pt regression	0.2413	0.8719	5 pt regression	0.2424	0.7386

Table 5.3: Comparison between the two linear regressions performed for tension as a function of DAC counts: the first only using 2 DAC values (2500 and 3000) and the second using 5 DAC values (2250, 2500, 2750, 3000 and 3250).

V as a function of ADC counts					
Ch 4	Calibration factor	Offset	Ch 45	Calibration factor	Offset
2 pt regression	0.2496	-3.7878	2 pt regression	0.2513	-3.4006
5 pt regression	0.2495	-3.4281	5 pt regression	0.2512	-3.1160

Table 5.4: Comparison between the two linear regressions performed for tension as a function of ADC counts: the first only uses 2 ADC values corresponding to DAC values 2500 and 3000; the second uses 5 ADC values corresponding to DAC values 2250, 2500, 2750, 3000 and 3250.

calibration factors of the corresponding regressions differ by a tenth of a thousandth, which corresponds to less than a 0.1 V difference when dealing with the maximum operating HV, which is 950 V. Given this, the linear regressions were considered suitable and used in CERN's Test Beam facilities as a calibration tool.

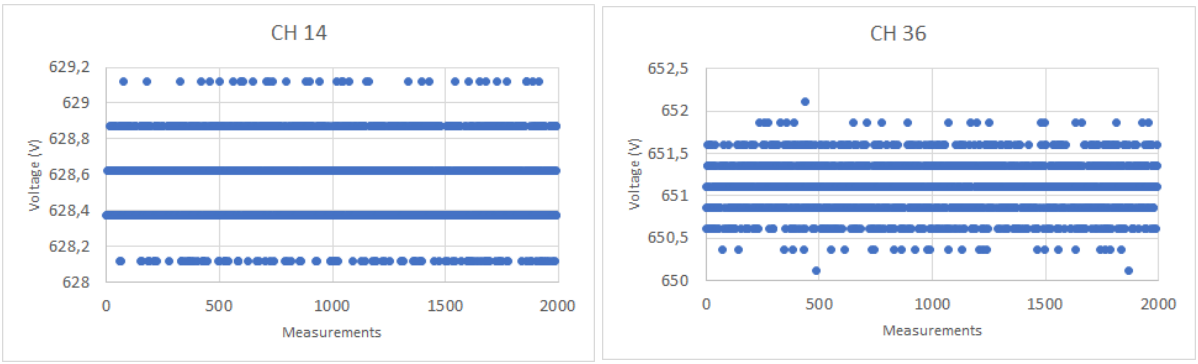
5.2 Tests at CERN Test Beam Facility

The HV Remote, HV Supplies and interface boards were tested at CERN Test Beam facility. The Test Beam setup consists of a set of three TileCal modules including one complete long barrel module equipped with 45 PMTs and instrumented with four 3-in-1 mini drawers. The PMT modules, which were installed on a remote rotating table depicted in figure 5.2, were connected to the HV Remote system via HV cables. The PMTs were then calibrated by equalising their current, which is induced by movable radioactive ^{137}Cs sources that cross every row of scintillating tiles near the edges. [40] Finally, the PMTs were subjected to different beam energies and monitored.

Figure 5.2: Picture of the Test Beam facility modules and the mobile table.

Once the installation of the modules was completed and communication between the TileCal DCS system and the digital control of the HV Supplies and HV Remote boards was established, a series of functional and stability tests began. The graphs displayed in figures 5.3, 5.4 and 5.5 result from a stability run of approximately 5 hours in which different HV input values were defined by a user. The PMTs had already been calibrated with a caesium 137 radioactive source.

The results showed, generally, variations between 1 V and 2 V that were not centred on the requested value. The formation of voltage bands was verified in all channels as well. The graphs in figure 5.3 demonstrate the generic behaviour of the HV channels.



(a) Channel 14 with a programmed voltage of 628.5 V

(b) Channel 36 with a programmed voltage of 650 V

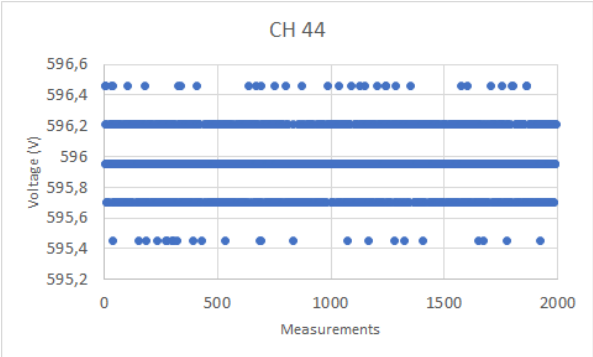
Figure 5.3: ADC voltage measurements acquired in a 5 hour period of two PMT channels.

Channels 44, 43, 42 and 41 showed the same amplitude of variations, but with a 5 V offset from the programmed value. It is important to bear in mind that these channels belong to a group of channels that share an on/off circuit. It was later discovered that the calibration values, used to acquire this data and described in the previous section, of these channels were not correctly used, which may be the cause of this offset. Even so, the group already appeared to have some instability, as described in section 5.1.

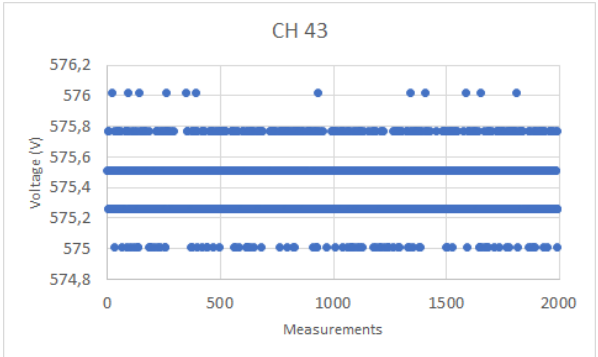
However, channels 15 and 16, shown in figure 5.5, presented voltage variations of approximately 10 V. These results were unexpected, as the data obtained during channel calibration, described in section 5.1, did not present any evidence of this type of fluctuation.

Once the PMT modules were removed from the Test Beam table, it was discovered that the two PMTs were not connected at all to the HV Remote board, which explains the observed fluctuations. Although this was a technical error, its result demonstrates how the HV Remote outputs respond when there is not any load connected.

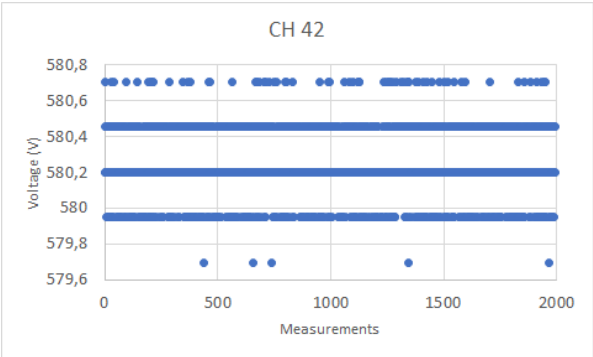
After reconnecting the PMTs to the board correctly and reinstalling the modules in the Test Beam setup, further stability tests were performed. In a 7 hour stability run, with all HV channel outputs set to 650 V, it was found that the fluctuations at the output of channels 15 and 16 decreased abruptly. The graphs of figure 5.6 show the recorded HV variations, which describe a behaviour in line with the remaining channels.



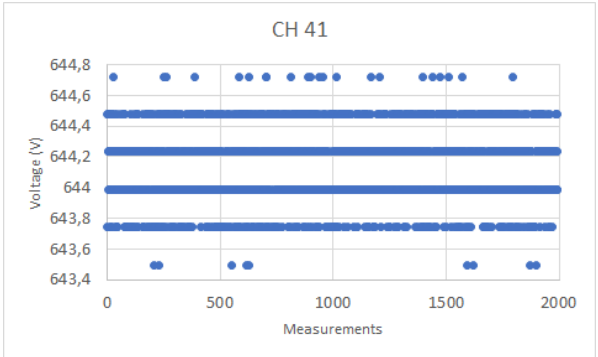
(a) Channel 44 with a programmed voltage of 600.7 V



(b) Channel 43 with a programmed voltage of 580.4 V



(c) Channel 42 with a programmed voltage of 584.4 V



(d) Channel 41 with a programmed voltage of 647.6 V

Figure 5.4: ADC voltage measurements acquired in a 5 hour period of 4 PMT channels with a significant offset.

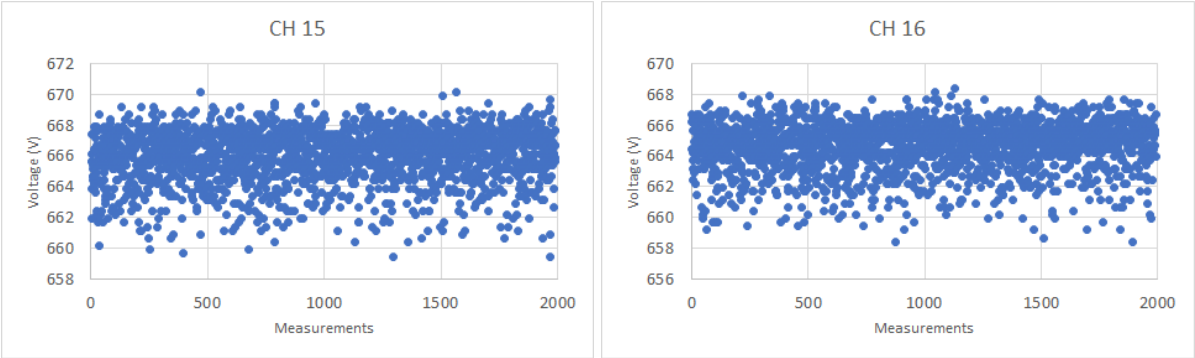


Figure 5.5: ADC voltage measurements in a 5 hour period of two PMT channel outputs for a programmed voltage of 650 V.

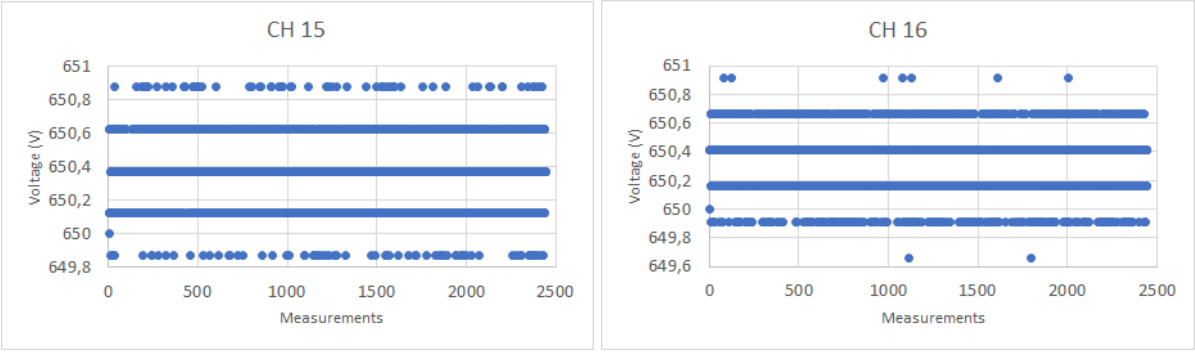
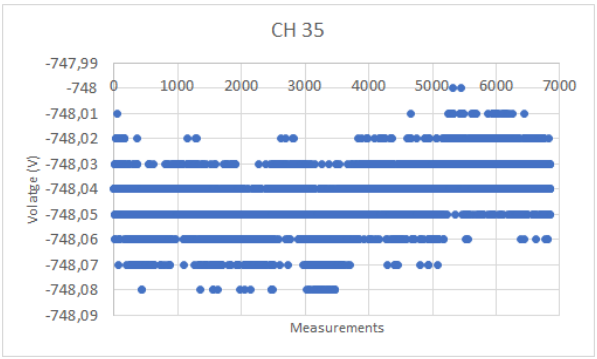


Figure 5.6: ADC voltage measurements in a 7 hour period of two PMT channel outputs for a programmed voltage of 650 V.

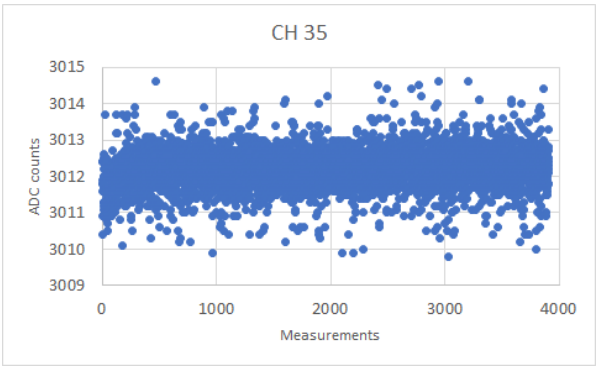
5.3 Stability Tests at LIP

Following the Test Beam tests, a series of stability tests in a lab environment was performed. There are only two pairs of HV Remote and HV Supplies boards from the current prototyping phase and one pair was left at CERN for continuous testing. The other pair is also being continuously tested in the LIP laboratory. When performing these tests, one of the HV DC/DC converters had not yet been mounted on the HV Supplies board. Given that each of the two DC/DC converters supplied one-half of the HV Remote, the stability and load tests described below were performed with only half of the HV Remote channels operational.

For three days, the HV Remote ADC monitored all of its channels, which had an applied voltage of 750 V. A voltmeter was also used to monitor two of the channels, 35 and 36. Since the ADC sampling time is shorter than the HV Remote regulation loops' stabilisation time, each registered data point (in ADC counts) corresponds to an average of 5 data points sampled by the ADC. The graphs in figures 5.7 and 5.8 show the output voltage variation of each channel during the test's duration.

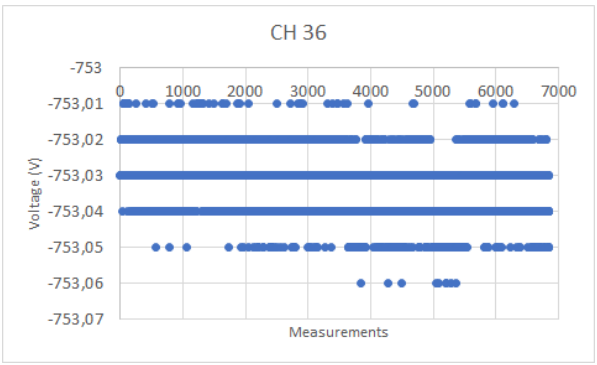


(a) Measurements taken with the voltmeter.

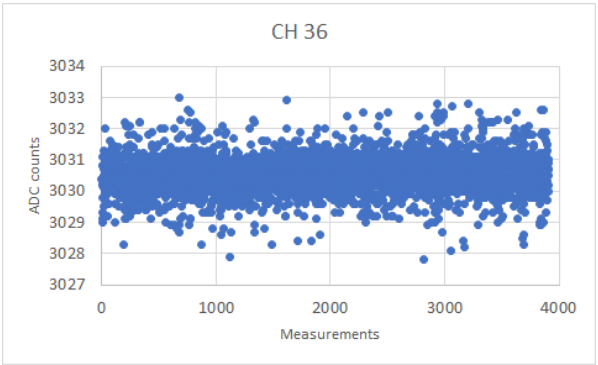


(b) Measurements taken with the ADC.

Figure 5.7: Voltage measurements acquired in a 3 day period of channel 35.



(a) Measurements taken with the voltmeter.



(b) Measurements taken with the ADC.

Figure 5.8: Voltage measurements acquired in a 3 day period of channel 36.

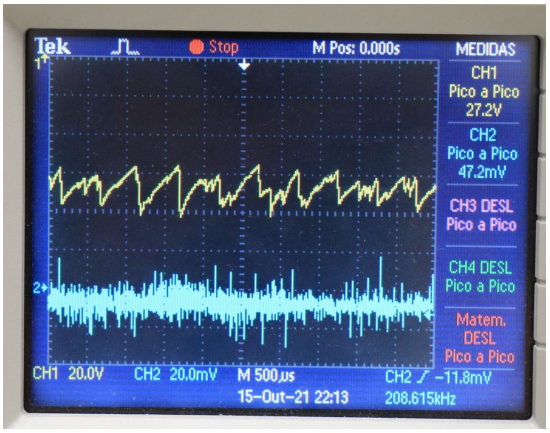
The data obtained using the voltmeter showed a maximum output voltage variation of 0.08 V and a standard deviation of 0.013 V. These results indicate that the HV Remote control loops' noise level is within the specifications set before hand, corresponding to 0.17% of the nominal voltage value. The graphs of the ADC measurements performed with the remainder channels are featured in appendix E.

5.4 Load Tests

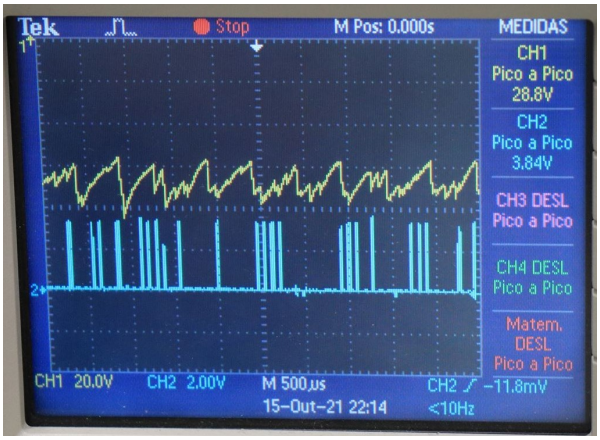
In order to characterise the board's response to different types of loads, each of its channel outputs was connected to different loads while monitored by an oscilloscope. For all tests performed and described below, the voltage applied to the HV channels of the HV Remote was 750 V.

5.4.1 Without Load

To monitor the board's behaviour while its outputs have no load, the oscilloscope was connected directly to the HV Remote channel's output pin through a HV probe with $\times 100$ scale factor. The Raspberry Pi's SCLK signal was used as the trigger level for the oscilloscope and the channel measuring the HV's input coupling was set to Alternating Current (AC).



(a) Without any data acquisition taking place.



(b) With a stability test running.

Figure 5.9: Pictures of channel 44 fluctuations detected with an oscilloscope, in response to a float load. The blue channel in the oscilloscope is the SCLK signal and the yellow one is the HV Remote channel output voltage AC measurement.

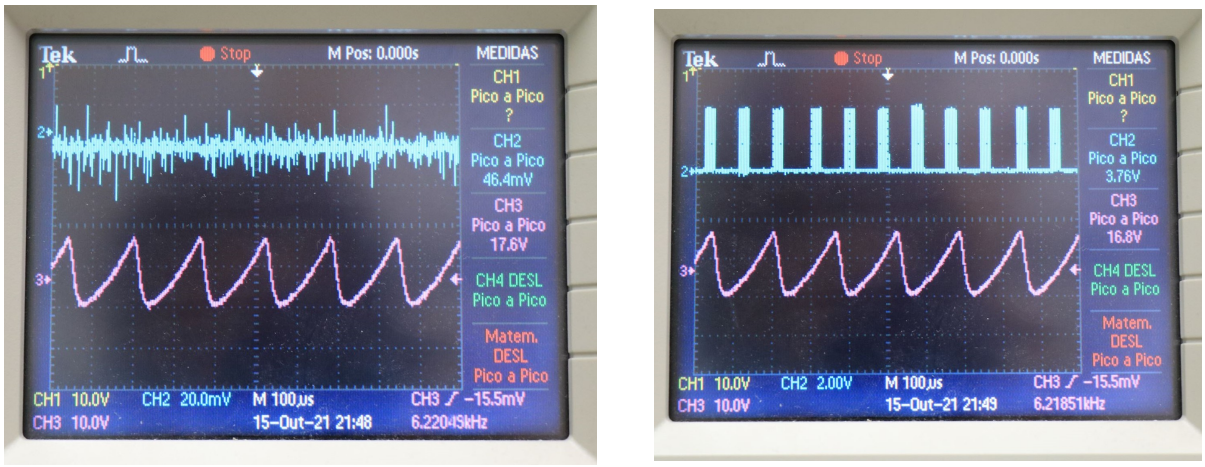
All channels displayed the same behaviour depicted in the pictures of figure 5.9, with fluctuations between 27 V and 34 V. The fluctuations detected by the oscilloscope are greater than the ones detected for the disconnected PMTs during the Test Beam, which was unexpected. However, it is possible that these results might have been affected by the HV oscilloscope probe, which had an attenuation factor of $\times 100$. The HV outputs were then connected to a voltmeter for further inspection, but these fluctuations were not detected.

It is possible to conclude from this data that fluctuation amplitude increases by establishing SPI communication with the board, as shown in the aforementioned pictures. In either case, the fluctuations detected during this test and at the Test Beam are not out of specification as they are not representative of HV Remote's operating conditions.

5.4.2 Resistive Load

As a resistive load, a voltage divider composed of a set of 5 resistors of 1 MΩ was used. A $\times 10$ gain oscilloscope probe was placed at the middle of the voltage divider, measuring a 2 MΩ load and assuring the measured voltage is always under the probe's voltage limit (300 V). Figure 5.10 shows pictures of the observed events for channel 44 with and without SPI communication.

All channels displayed fluctuations between 14 V and 18 V for this resistance value. The fluctuation amplitude has decreased when compared to the results obtained with a float load, which is expected.



(a) Without any data acquisition taking place.

(b) With a stability test running.

Figure 5.10: Pictures of channel 44 fluctuations detected with an oscilloscope, in response to a passive load. In both pictures, the blue channel in the oscilloscope is the SCLK signal and the pink channel is the AC measurement of load voltage.

Increasing the resistance will result in a decrease in the fluctuations but also the nominal output voltage.

5.4.3 Active Load

A HV divider is required to apply HV to the PMTs. The HV dividers currently used at TileCal (figure 5.11) are passive dividers with a Resistor-Capacitor (RC) low-pass filter at the HV input. As part of the undergoing upgrade, these voltage dividers will be replaced by active ones, composed of a resistor voltage divider, coupled with capacitors, transistors and diodes. Both voltage dividers have similar behaviours. The new HV divider was coupled as a load to the experimental setup and the observed oscilloscope measurements are displayed in figure 5.12.

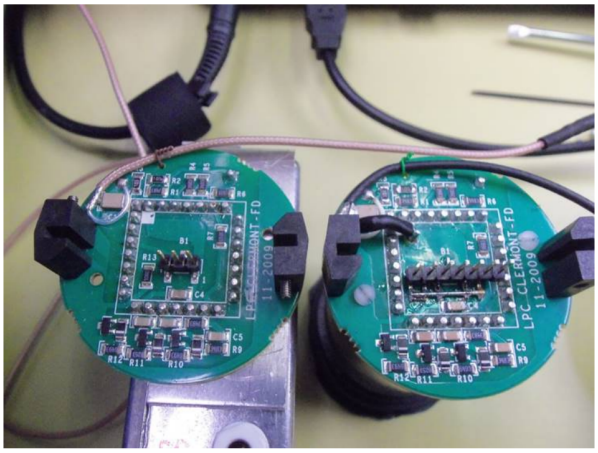
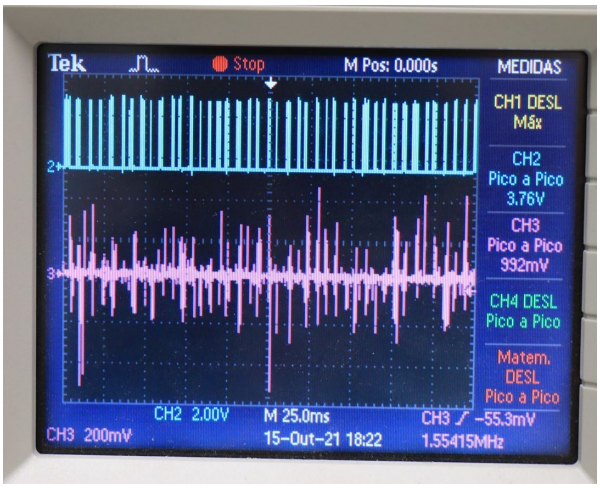


Figure 5.11: Example of one of the high voltage dividers connected to the Tilecal PMTs. [40]

For all channels monitored, fluctuations at the load's output increased whenever SPI communication was established with the board. However, they were significantly inferior when compared to the fluctuations observed with the previous resistive load, suggesting the fluctuations seen before with other loads were high-frequency oscillations.

(a) Without any data acquisition taking place.



(b) With a stability test running.

Figure 5.12: Picture of channel 44 fluctuations detected with an oscilloscope, in response to an active voltage divider. The blue channel in the oscilloscope is the SCLK signal and the pink channel is the AC measurement of load voltage.

5.4.4 PMT Load

In order to observe the effect of the output channels' fluctuations in the response of a PMT, it was necessary to make some changes to the experimental setup. The PMT was connected to the previously monitored active voltage divider and placed in front of a blue LED powered by a signal generator. The two components have been covered so that no other light source is detected by the PMT. The diagram in figure 5.13 illustrates the experimental setup. The set voltage remained 750 V. Figure 5.14 shows how a PMT detection signal affects its HV input.

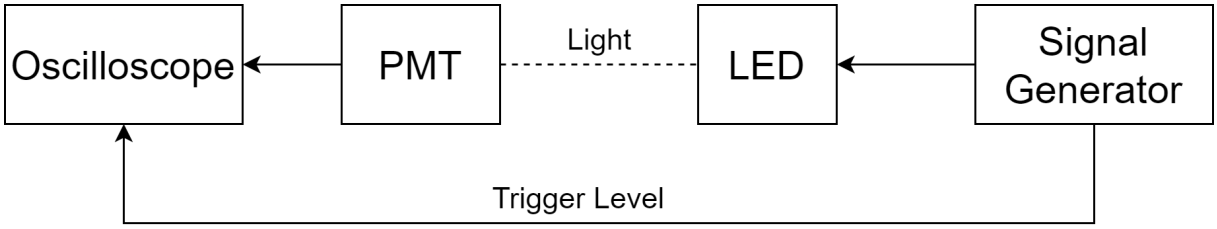


Figure 5.13: Diagram describing the experimental setup used to test the HV Remote response to a PMT load.

When the PMT detects the LED impulse, there is a significant drop of the supplied HV of approximately 2 V and the voltage then stabilises relatively slowly. The PMT response shown in figure 5.14, in blue, is unexpected, as it is supposed to have a single impulse signal and it has multiple. It is believed that the following pulses are reflections along the probe as it might not be correctly terminated with an appropriate resistor. These reflection signals seem to also affect the stabilisation time of the HV Remote output HV.

In order to obtain statistics that characterise the response of a PMT to light when powered by the HV Remote, a digital oscilloscope that allows the acquisition and export of data was used. The experimental setup remained the same, except the oscilloscope was replaced by a digital one and only the output voltage of the HV Remote channel connected to the PMT was monitored. This prevents the theorised signal reflections observed before. The PMT was supplied with 603.58 V. The minimum value fluctuation value was monitored using the oscilloscope's LOW measurement setting. This way, it was possible to quantify the decrease in HV in response to the PMT signal observed in figure 5.14.

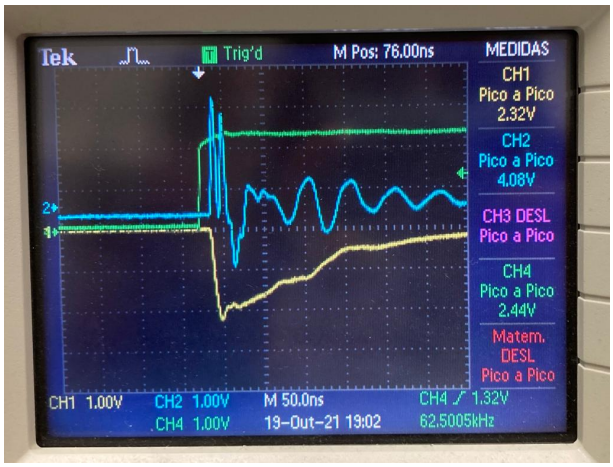


Figure 5.14: Picture of channel 43 fluctuations detected with an oscilloscope, in response to a PMT signal. The blue channel in the oscilloscope is the PMT impulse signal, the green channel is the LED signal, used as the trigger level, and the yellow channel is the HV Remote channel output voltage AC measurement.

Two consecutive measurements were performed for each of the operating modes: with and without simultaneous data acquisition via SPI. All measurements had approximately 100000 events. Figures 5.15 and 5.16 show the PMT signal in response to the LED light (at the top) and the histograms (at the bottom) of the PMT signal peak values of each measurement run without and with SPI communication occurring, respectively.

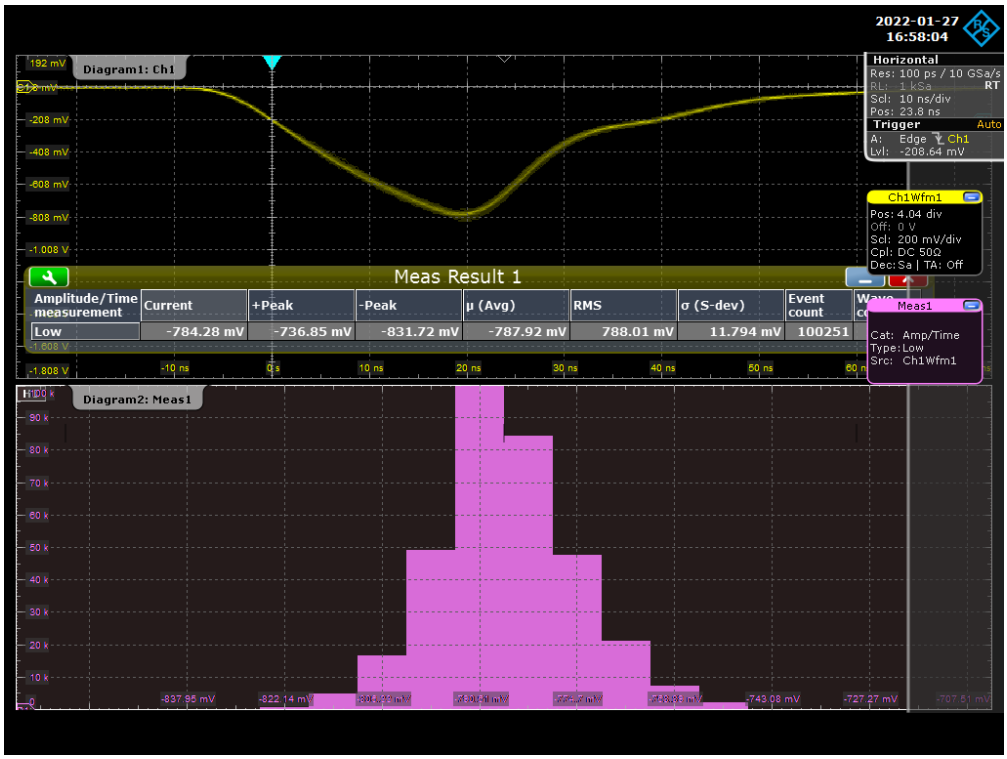


Figure 5.15: Oscilloscope print screen at the end of a measurement run of the output HV in response to a PMT signal (in yellow) without SPI communication occurring. The pink histogram describes the frequency of the detected peak values.

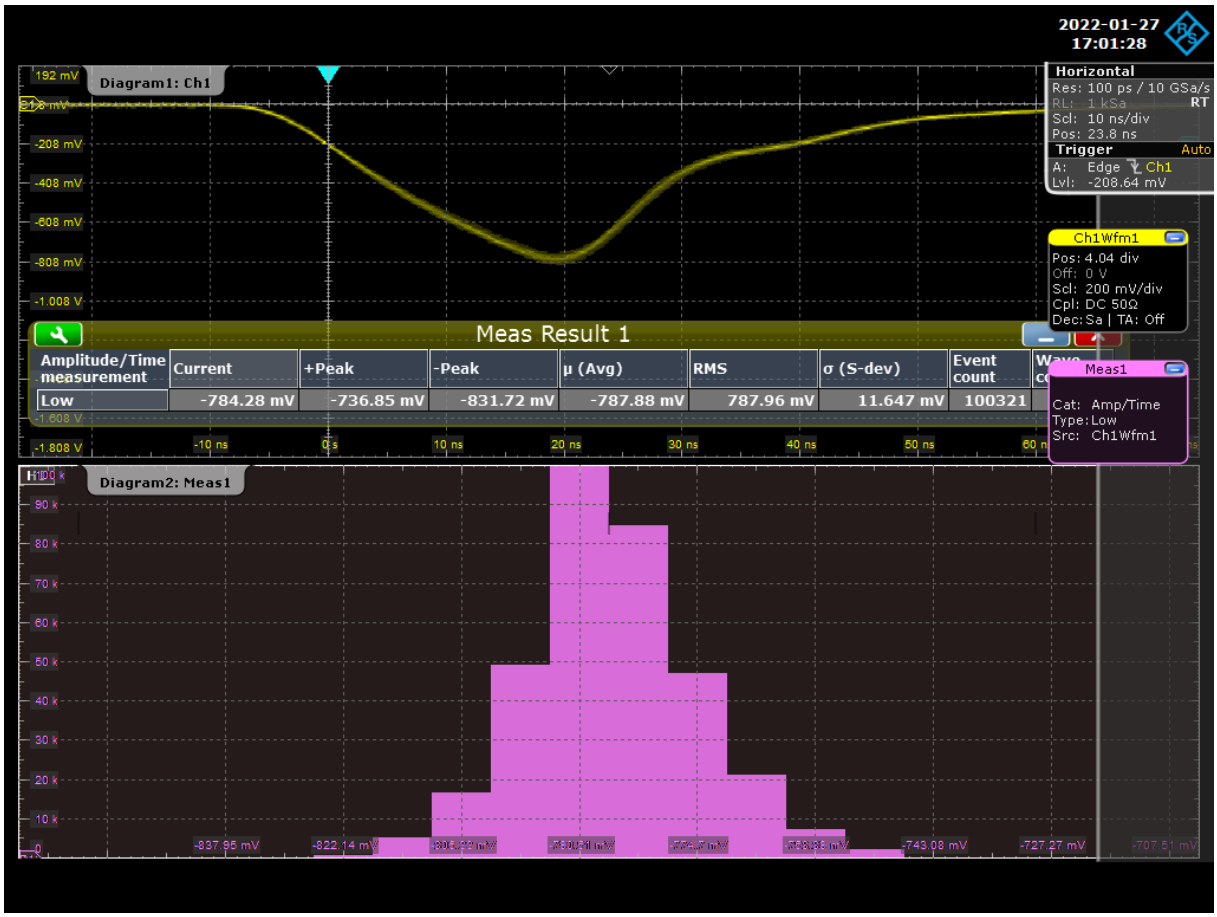


Figure 5.16: Oscilloscope print screen at the end of a measurement run of the output HV in response to a PMT signal (in yellow) while there is SPI communication occurring. The pink histogram describes the frequency of the detected peak values.

In both figures, the most important run parameters are displayed:

- Current: last value detected;
- +Peak: minimum peak value measured;
- -Peak: maximum peak value measured;
- μ (Avg): mean of measured peak values;
- RMS: peak values' root mean square;
- σ (S-dev): peak values' standard deviation;
- Event count: number of PMT signals monitored.

The results displayed above show that while SPI communication does interfere with the measured HV values, its impact is very little when compared to the nominal HV output value. This is shown by the mean peak values with and without SPI communication, which differ from each other by 0.04 mV.

The average HV output variation caused by the PMT response was -787.9 mV, which corresponds to 0.13% of the nominal HV supplied to the PMT. This variation may contribute to the 1 to 2 V variations seen during the Test Beam results' analysis. However, its impact must be further studied by carrying out tests with different nominal HV values.

Chapter 6

Conclusions and Future Work

The work developed in this thesis was greatly affected by the COVID-19 pandemic, which caused a global shortage of electronic components that has yet to be fully resolved at the time of writing this thesis. This shortage caused significant delays in acquiring components and producing the necessary HV Remote, HV Supplies, HV Reading and Cable Test board prototypes.

HV Remote's performance tests were hampered by some lack of automation of the testing process, making the process time-consuming and more complex than anticipated. The HV Reading board, designed to automate this process, is, at the time of writing this thesis, in the manufacturing process. As such, it was only possible to carry out load tests on a few channels, given that switching between the different experimental setups was time-consuming and that only half of the channels were powered due to a missing DC/DC converter of the HV Supplies board.

Despite the difficulties mentioned above, the HV Remote's functional and stability tests were carried out, including at the CERN Test Beam facility, allowing characterisation of its behaviour in the environment for which the setup is intended. These tests also assisted the project team in making minor changes, and a third prototype of the HV Remote is currently in production.

In the future, more tests will have to be performed on the HV Remote. Namely load tests for all the channels and functional testing of the group enable/disable switches, that were rendered inoperable as soon as the current prototypes arrived.

The development and design of the Cable Test board are now complete. It now awaits product availability to be assembled and soldered. Digital control software and a GUI for the board were also developed but have not been completely tested. In the future, a continuous acquisition option must be included in the GUI, as it will become useful to perform data readings over large periods of time.

Much work remained to be done, including the functional testing of the Cable Test board's components and software, and its integration with the HV Reading hardware and software, as well as the performance of continuity and insulation tests on the cables that will connect the HV Remote to the TileCal PMTs.

References

[1] P. Rato. “Test system for the remote high voltage distribution system of the Tilecal”. MA thesis. Faculdade de Ciências da Universidade de Lisboa, Sept. 2021.

[2] R. Marques. “Testes Funcionais e Calibração dos Protótipos do Sistema de Distribuição da Alta Tensão do Tilecal”. MA thesis. Faculdade de Ciências da Universidade de Lisboa, Dec. 2021.

[3] A. Airapetian et al. *ATLAS detector and physics performance: Technical Design Report, 1*. Tech. rep. Geneva, 1999. URL: https://cds.cern.ch/record/391176.

[4] CERN. *ATLAS*. URL: https://home.cern/science/accelerators/large-hadron-collider.

[5] F. Scuri. “Upgrade of the ATLAS Tile Calorimeter for the High Luminosity LHC”. In: *Journal of Physics: Conference Series* 1162 (Jan. 2019), p. 012017. URL: https://doi.org/10.1088/1742-6596/1162/1/012017.

[6] CERN. *The Large Hadron Collider*. URL: https://home.cern/science/experiments/atlas.

[7] ATLAS Collaboration. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *Journal of Instrumentation* 3.08 (Aug. 2008). URL: https://doi.org/10.1088/1748-0221/3/08/s08003.

[8] ATLAS Experiment. *The Magnet System*. URL: https://atlas.cern/discover/detector/inner-detector.

[9] ATLAS Experiment. *Muon Spectrometer*. URL: https://atlas.cern/discover/detector/muon-spectrometer.

[10] J. Abdallah et al. “The optical instrumentation of the ATLAS Tile Calorimeter”. In: *Journal of Instrumentation* 8.01 (Jan. 2013), P01005–P01005. URL: https://doi.org/10.1088/1748-0221/8/01/p01005.

[11] ATLAS Experiment. *Trigger and Data Acquisition*. URL: https://atlas.cern/discover/detector/trigger-daq.

[12] D. Calvet et al. “The High Voltage distribution system of the ATLAS Tile Calorimeter and its performance during data taking”. In: *Journal of Instrumentation* 13.08 (Aug. 2018). URL: https://doi.org/10.1088/1748-0221/13/08/p08006.

[13] R. Chadelas et al. *High voltage distributor system for the Tile hadron calorimeter of the ATLAS detector*. Tech. rep. ATL-TILECAL-2000-003. PCCF-RI-2000-04. Aubière: Clermont-Ferrand 2. Lab. Phys. Corpusc. Cosmol., Feb. 2000. URL: https://cds.cern.ch/record/436230.

[14] I. Dawson and V. Hedberg. “Radiation in the USA15 cavern in ATLAS”. In: (2004).

[15] G. Evans for the LIP team. *HV Remote Board*. Tech. rep. Lisboa, Portugal, Nov. 2020.

[16] F. Cuim. "Functional Tester for High Voltage Boards of the TILECAL Calorimeter". MA thesis. Faculdade de Ciências da Universidade de Lisboa, 2019. URL: http://hdl.handle.net/10451/41488.

[17] P. Dhaker. "Introduction to the SPI Interface". In: (Sept. 2018). URL: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html.

[18] MIKROE. *SPI Bus*. URL: https://www.mikroe.com/blog/spi-bus.

[19] Microchip. *16-Bit I/O Expander with Serial Interface*. MCP23017/MCP23S17 datasheet. [Revised Jul. 2016]. June 2005. URL: https://ww1.microchip.com/downloads/en/devicedoc/20001952c.pdf.

[20] Texas Instruments. *DAC7568, DAC8168, DAC8568 12-/14-/16-Bit, Octal-Channel, Ultralow Glitch, Voltage Output, Digital-to-Analog Converters with 2.5-V 2-ppm/°C Internal Reference*. DAC7568 datasheet. [Revised Apr. 2018]. Jan. 2009. URL: https://www.ti.com/lit/ds/symlink/dac7568.pdf?ts=1643368985768&ref_url=https%5C%253A%5C%252F%5C%252Fwww.google.com%5C%252F.

[21] ON Semiconductor. *5 Watt Surmetic 40 Silicon Zener Diodes*. [Rev. 7]. May 2006. URL: https://www.datasheetq.com/datasheet-download/798872/1/ONSEMI/1N5386B.

[22] Maxim Integrated. *+2.7V, Low-Power, 12-Bit Serial ADCs in 8-Pin SO*. MAX17531 datasheet. [Revised Oct. 2010]. Mar. 2010. URL: https://datasheets.maximintegrated.com/en/ds/MAX1240-MAX1241.pdf.

[23] En.wikipedia.org. *DC-to-DC converter*. URL: https://en.wikipedia.org/wiki/DC-to-DC_converter.

[24] R. Fernandez for the LIP team. *HVSupplies 2020*. Tech. rep. Lisboa, Portugal, Oct. 2020.

[25] En.wikipedia.org. *Relay*. URL: https://en.wikipedia.org/wiki/Relay.

[26] Advanced Energy. *Interfacing with the UltraVolt M and V Series of Microsize High Voltage Power Supplies*. [Rev. B]. URL: https://www.hvproducts.de/wp-content/uploads/2021/04/interfacing-with-high-voltage-m-and-v-series-tn-m-v-1-1.pdf.

[27] Advanced Energy. *Miniature, Micro-Sized, High Voltage Biasing Supplies*. UltraVolt M Series datasheet. Aug. 2019. URL: https://pt.mouser.com/datasheet/2/863/en_hv_m_series_data_sheet-1775618.pdf.

[28] Maxim Integrated. *4V to 42V, 50mA, Ultra-Small, High-Efficiency, Synchronous Step-Down DC-DC Converter with 22µA No-Load Supply Current*. MAX17531 datasheet. [Revised Aug. 2017]. Nov. 2014. URL: https://datasheets.maximintegrated.com/en/ds/MAX17531.pdf.

[29] Len Sherman. *Making a Voltage Inverter From A Buck (Step-Down) DC-DC Converter*. Application Note 3844. Maxim Integrated. URL: https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3844.html.

[30] Texas Instruments. *INA828 50-µV Offset, 7-nV/$\sqrt{Hz}$ Noise, Low-Power, Precision Instrumentation Amplifier*. INA828 datasheet. [Revised Jan. 2018]. Aug. 2017. URL: https://www.ti.com/lit/ds/symlink/ina828.pdf?ts=1635241044691&ref_url=https%5C%253A%5C%252F%5C%252Fwww.ti.com%5C%252Fproduct%5C%252FINA828.

[31] Z. Peterson. *Top PCB Layout Guidelines Every PCB Designer Needs to Know*. [Updated Oct. 2021]. Altium Limited. Feb. 2017. URL: https://resources.altium.com/p/top-5-pcb-design-guidelines-every-pcb-designer-needs-know.

[32] Altium Limited. *High Voltage PCB Design: Creepage and Clearance Distance*. [Updated Dec. 2020]. Nov. 2017. URL: https://resources.altium.com/p/high-voltage-pcb-design-creepage-and-clearance-distance.

[33] S. Sattel. *PCB Layout Basics Part 1: How to Place Your Components*. Autodesk Inc. URL: https://www.autodesk.com/products/eagle/blog/pcb-layout-basics-component-placement/.

[34] EMS Solutions. *Guidelines for Designing a High Voltage PCB*. Nov. 2020. URL: https://www.myemssolutions.com/guidelines-for-designing-a-high-voltage-pcb/.

[35] Cadence PCB Solutions. *High Voltage Circuit Design Guidelines and Materials*. URL: https://resources.pcb.cadence.com/blog/2020-high-voltage-circuit-design-guidelines-and-materials.

[36] Michel Deslierres. *SpiDev Documentation (draft version)*. July 2020. URL: https://www.sigmdel.ca/michel/ha/rpi/dnld/draft_spidev_doc.pdf.

[37] Texas Instruments. *DACx311 2-V to 5.5-V, 80-µA, 8-, 10-, and 12-Bit, Low-Power, Single-Channel, Digital-to-Analog Converters in SC70 Package*. DAC6311 datasheet. [Revised Jul. 2015]. Aug. 2008. URL: https://www.ti.com/lit/ds/symlink/dac6311.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1642073727824&ref_url=https%5C%253A%5C%252F%5C%252Fru.mouser.com%5C%252F.

[38] Texas Instruments. *MUX50x 36-V, Low-Capacitance, Low-Leakage-Current, Precision Analog Multiplexers*. MUX506, MUX507 datasheet. [Revised Nov. 2017)]. Nov. 2016. URL: https://www.ti.com/lit/ds/symlink/mux506.pdf?ts=1635152738514&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMUX506.

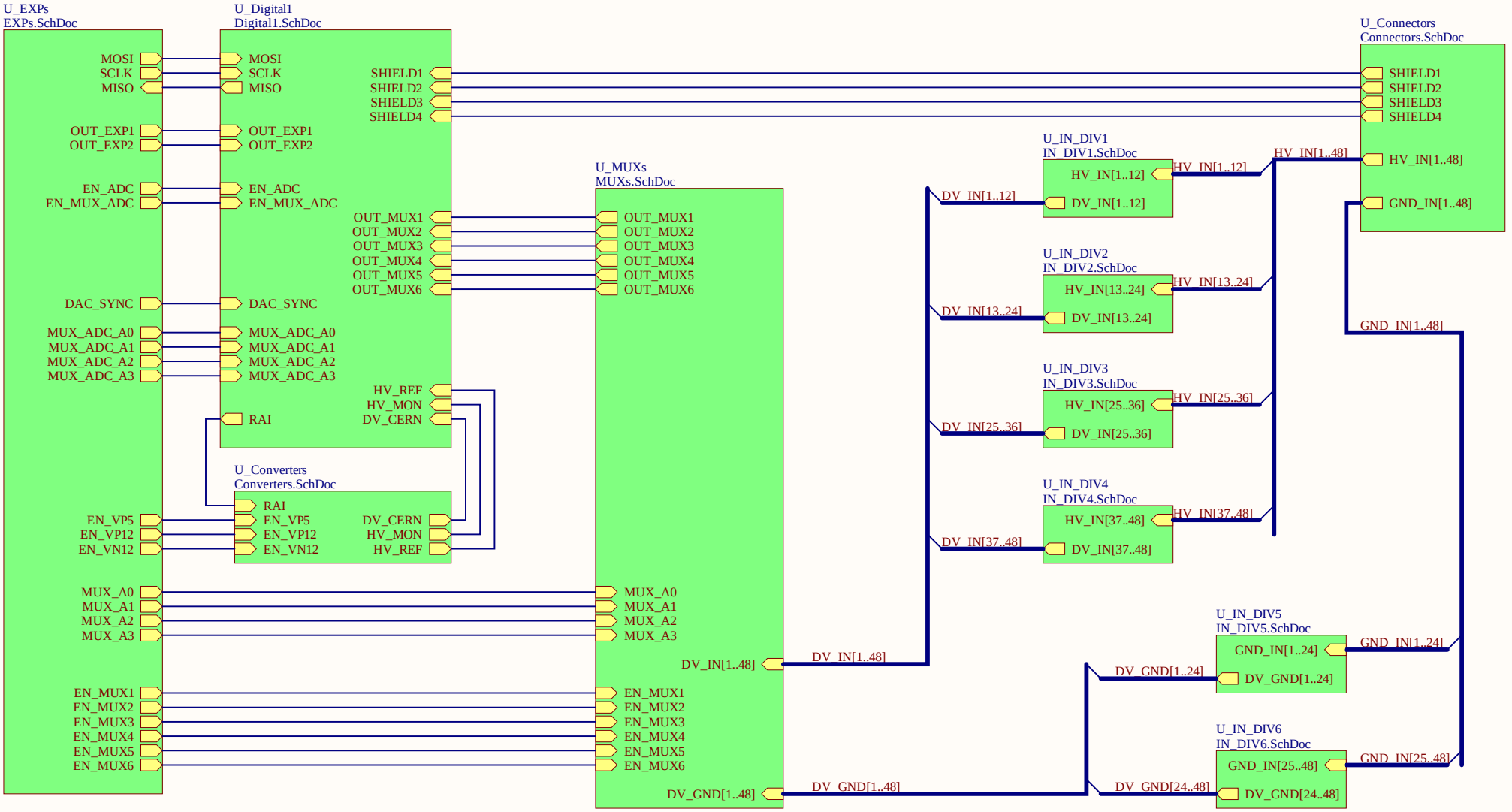
[39] The Qt Company Ltd. *Qt Designer Manual*. URL: https://doc.qt.io/qt-5/qtdesigner-manual.html.

[40] ATLAS Collaboration. *Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter*. Tech. rep. Geneva: CERN, Sept. 2017. URL: https://cds.cern.ch/record/2285583.

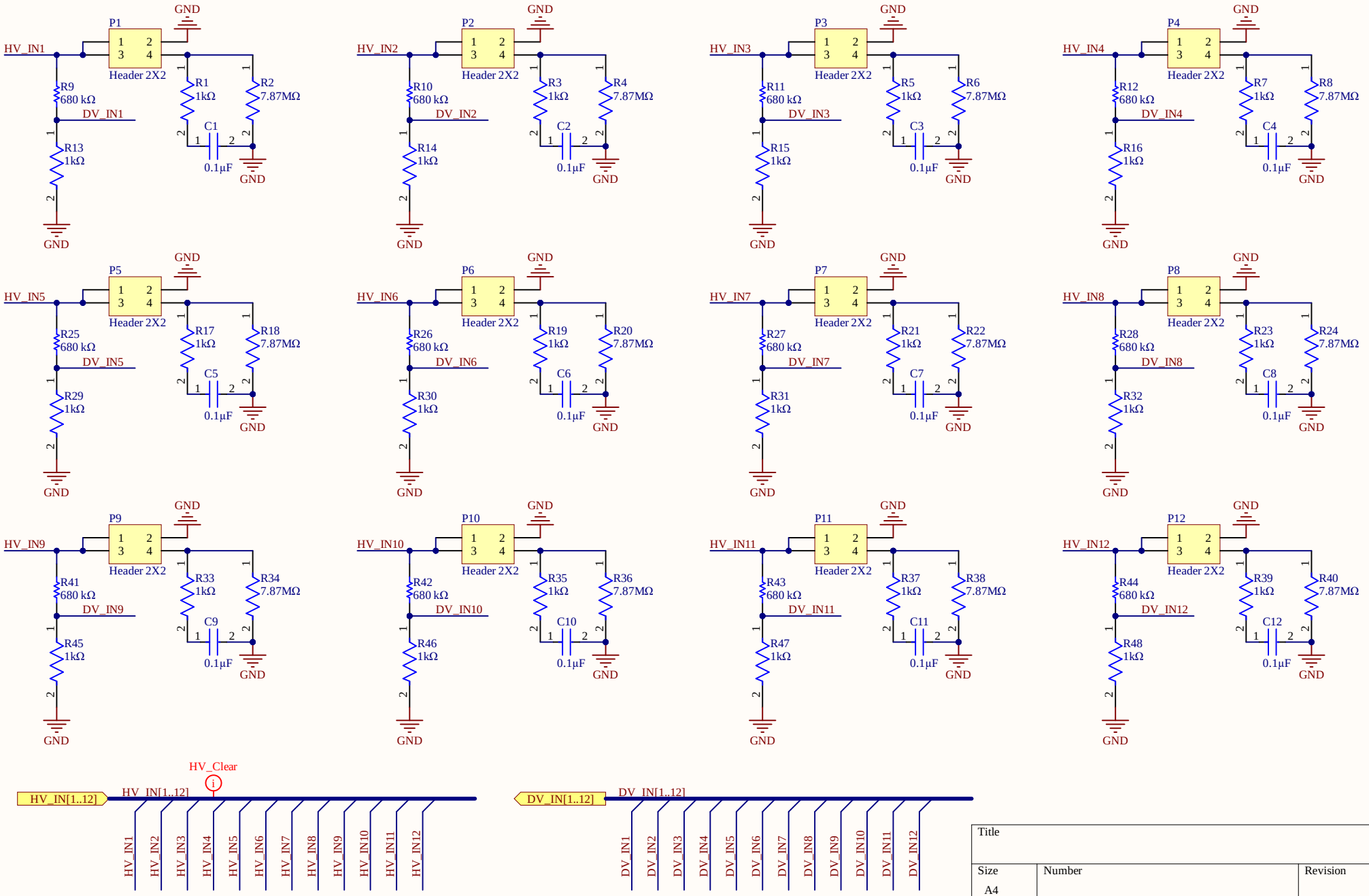
Appendices

Appendix A

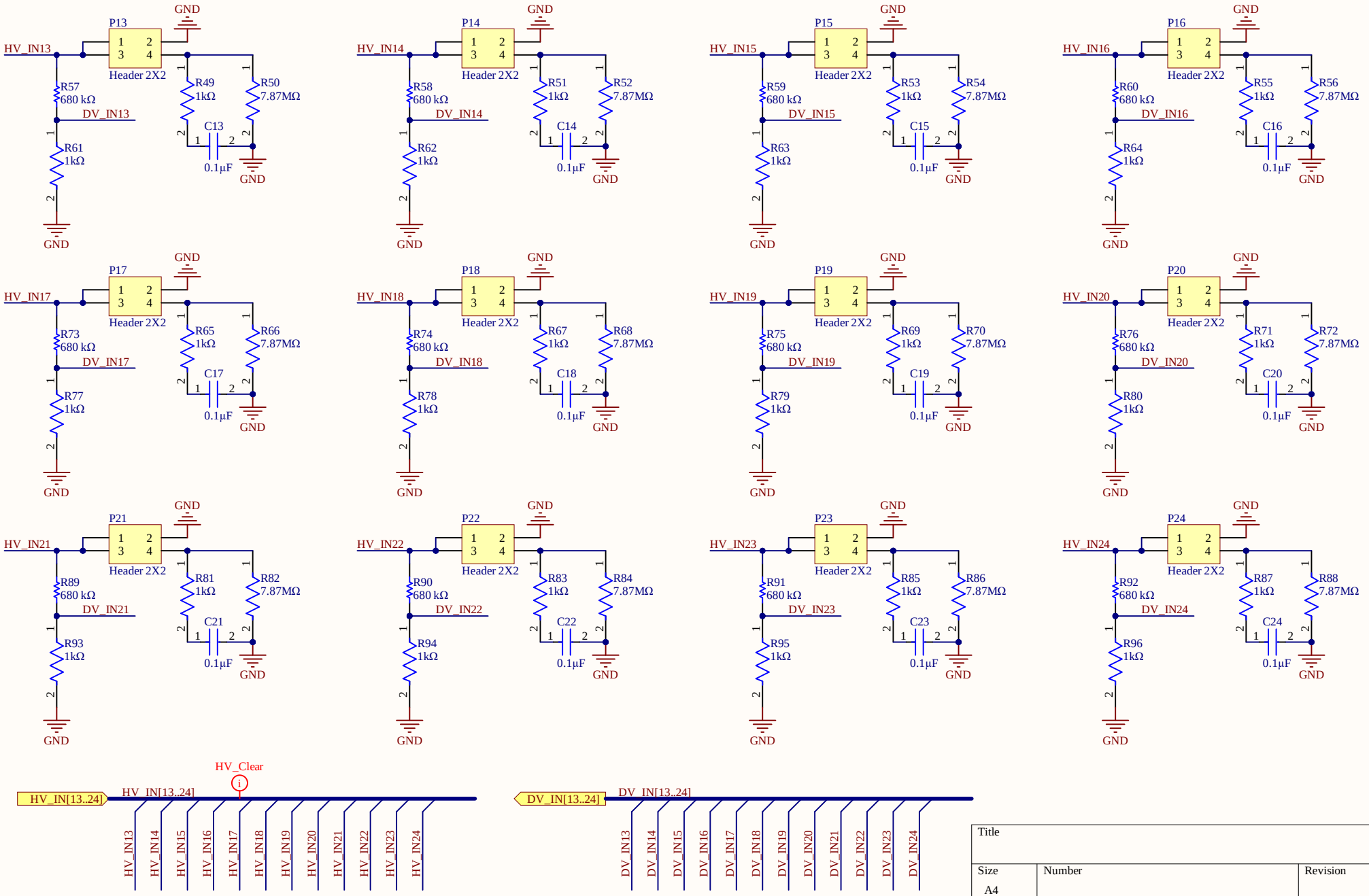
Schematic Designs of the Cable Test Board



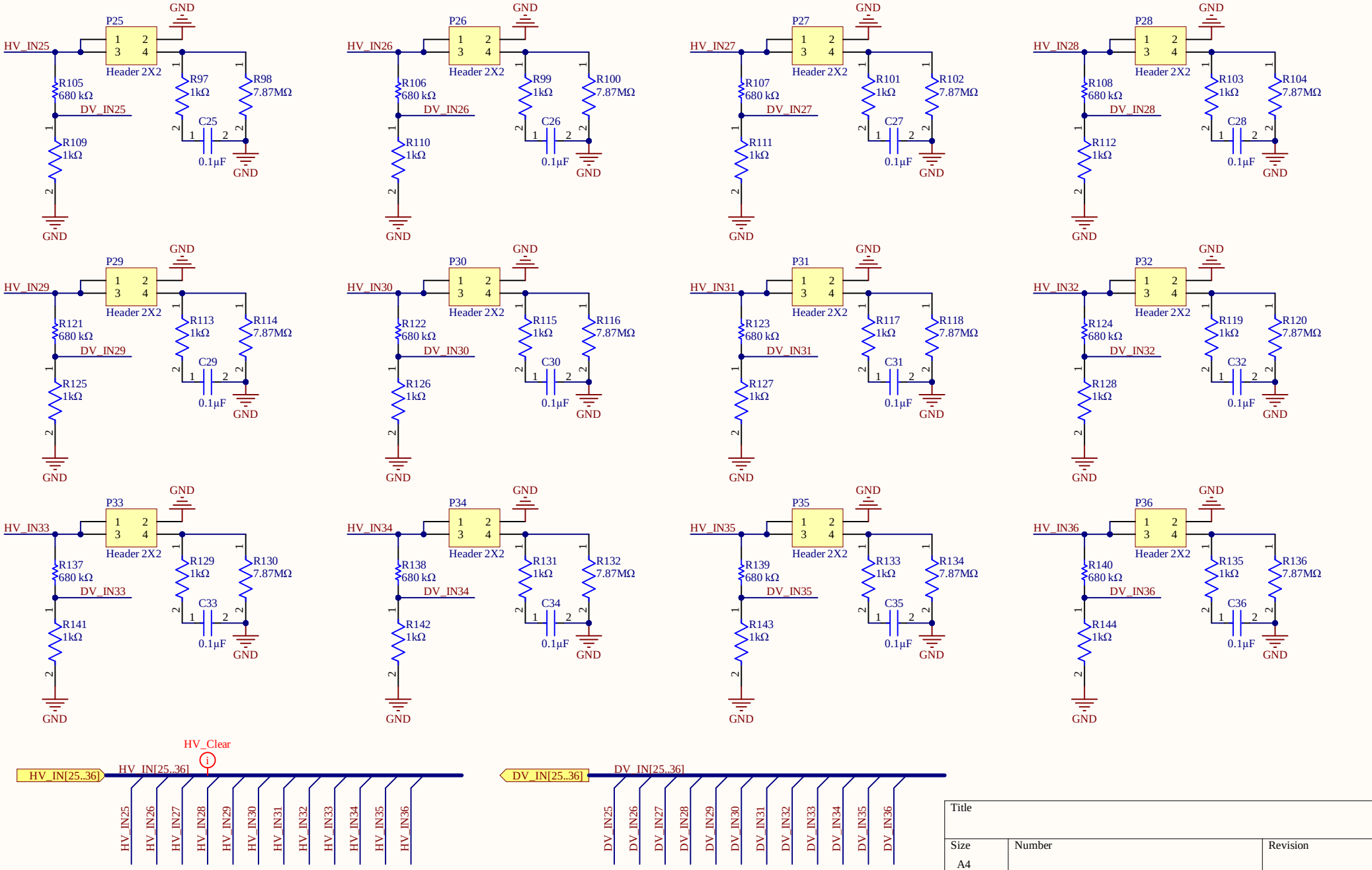
Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\Main_CTB.SchDoc		Drawn By:



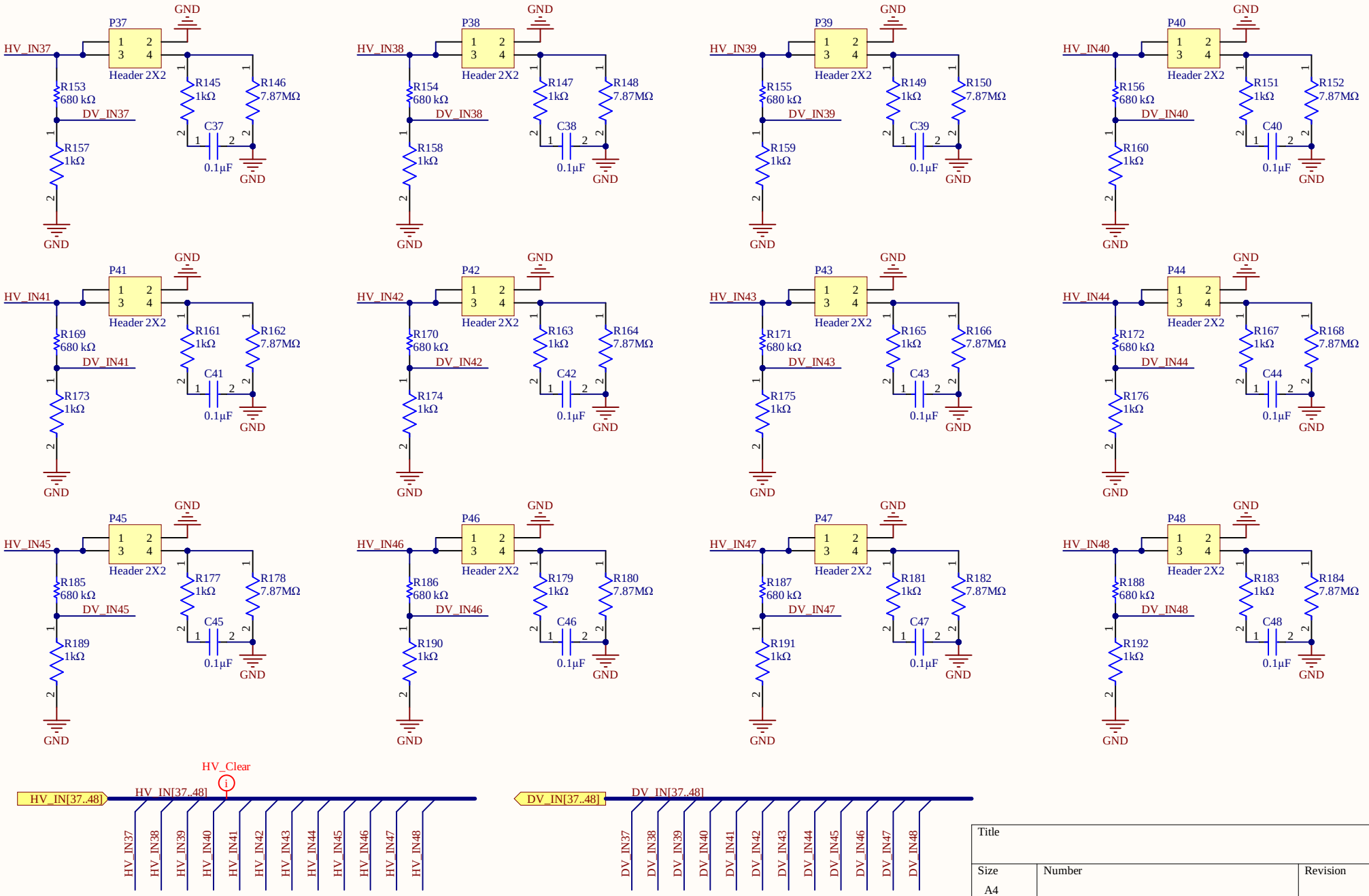
Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\IN_DIV1.SchDoc		Drawn By:



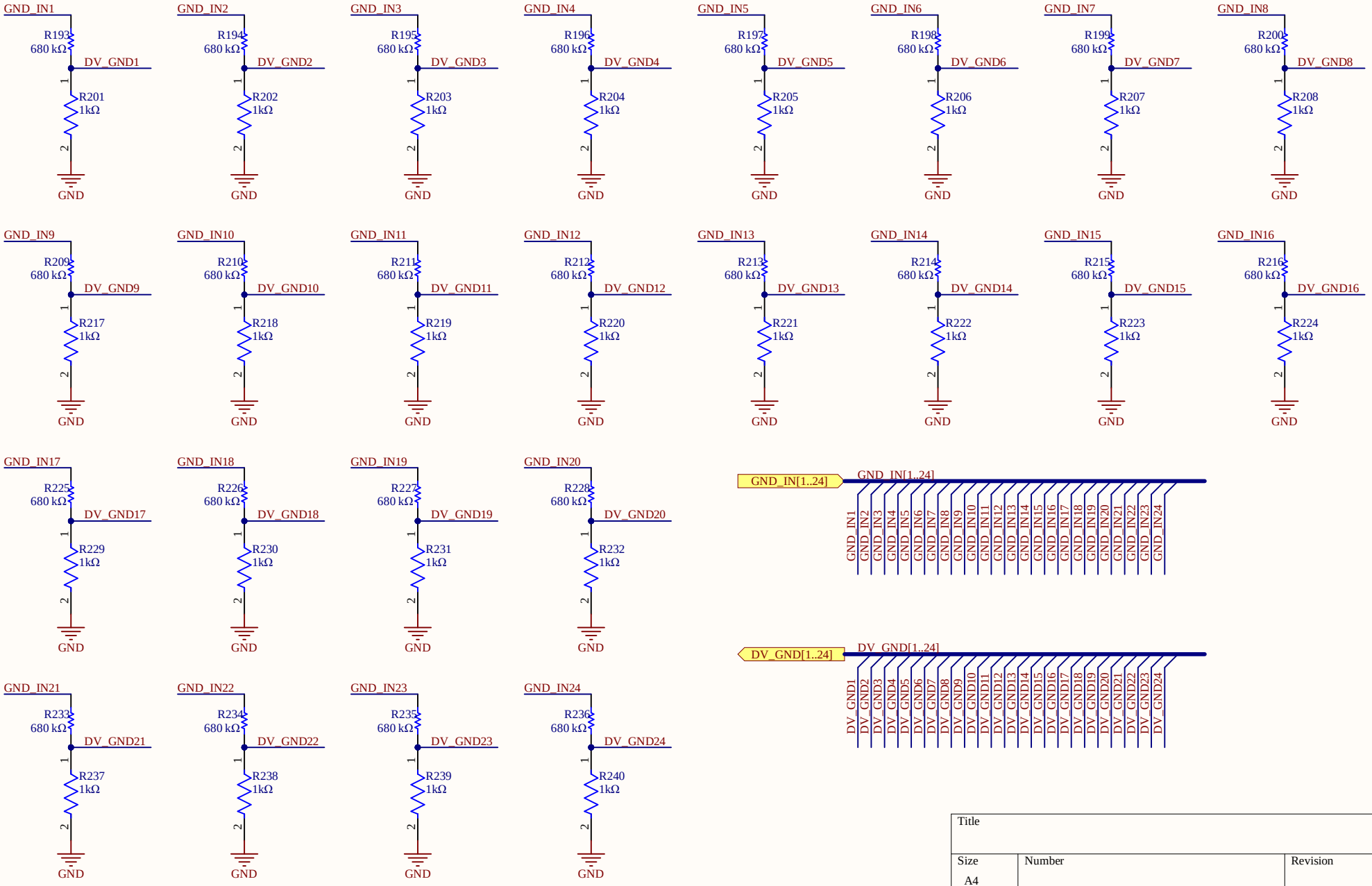
Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\IN_DIV2.SchDoc		Drawn By:

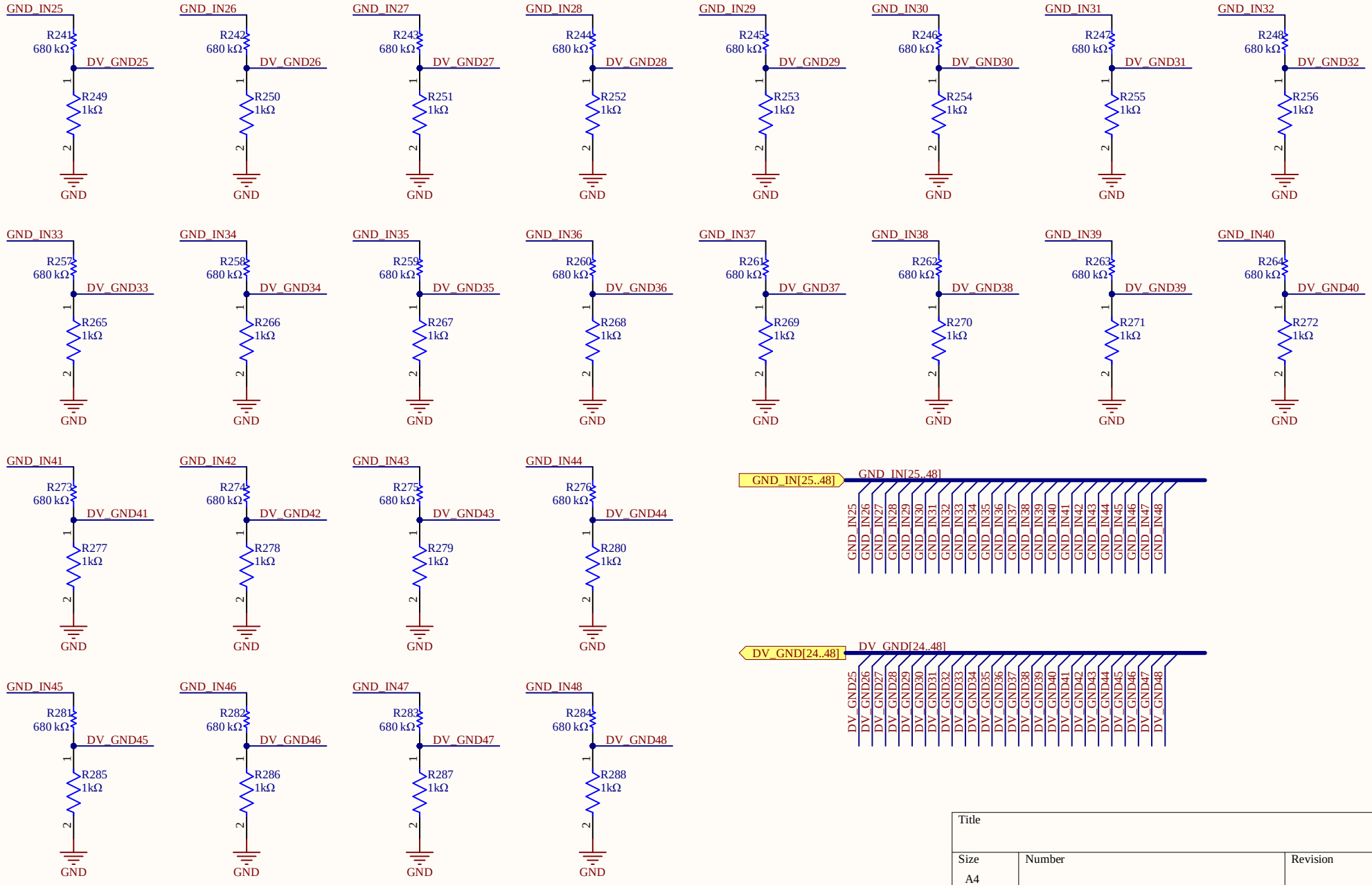


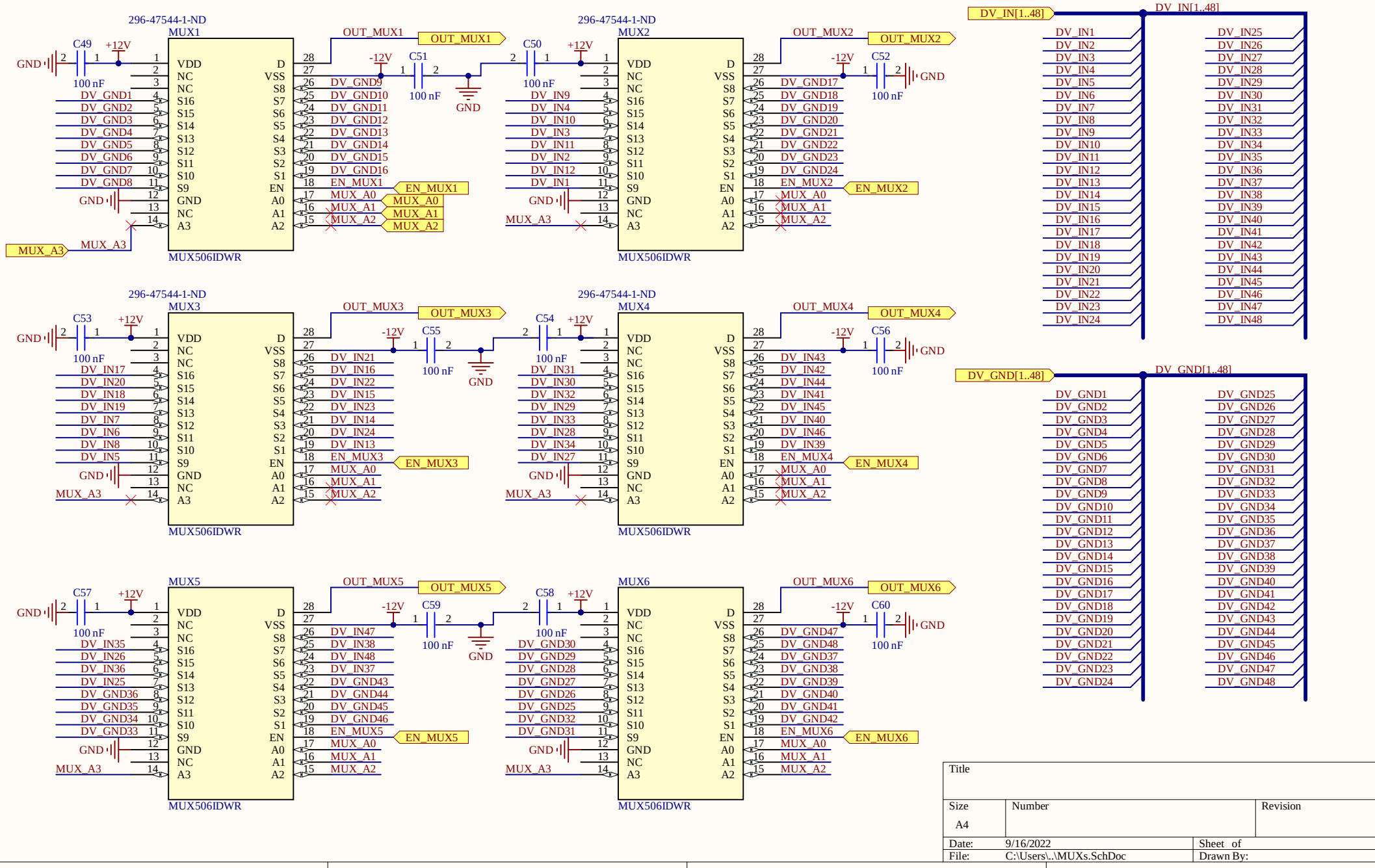
Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\IN_DIV3.SchDoc		Drawn By:

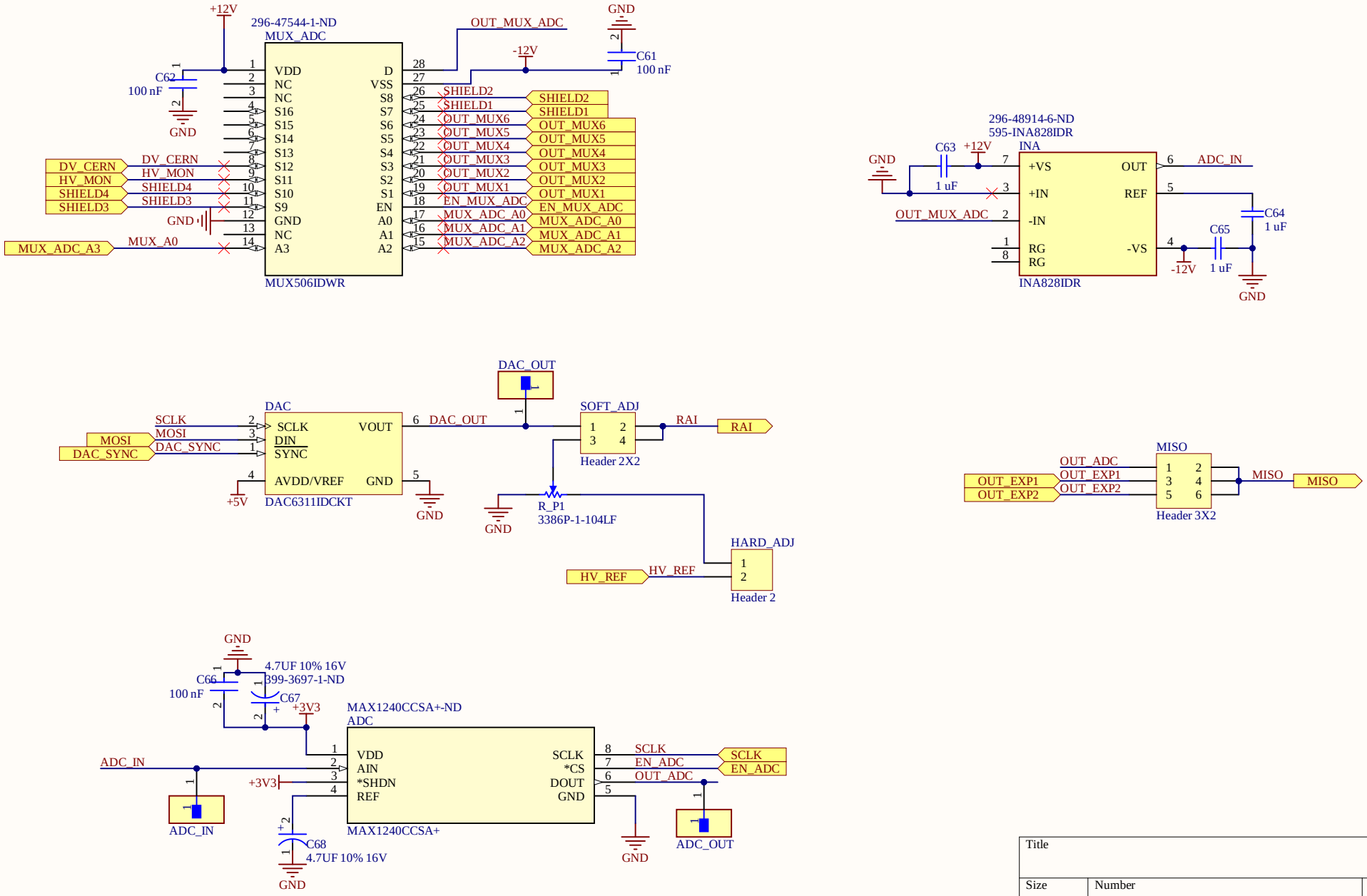


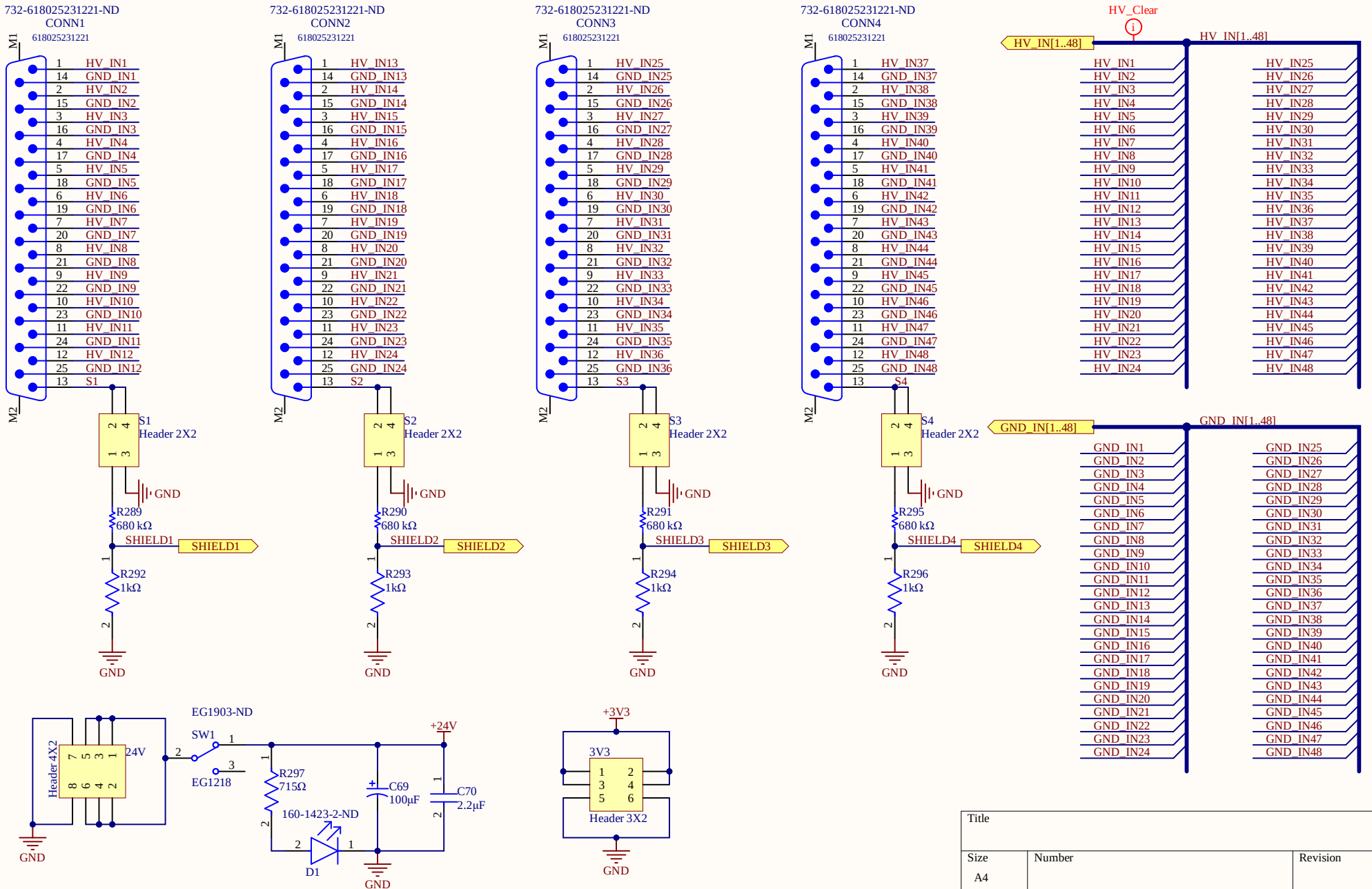
Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\IN_DIV4.SchDoc		Drawn By:

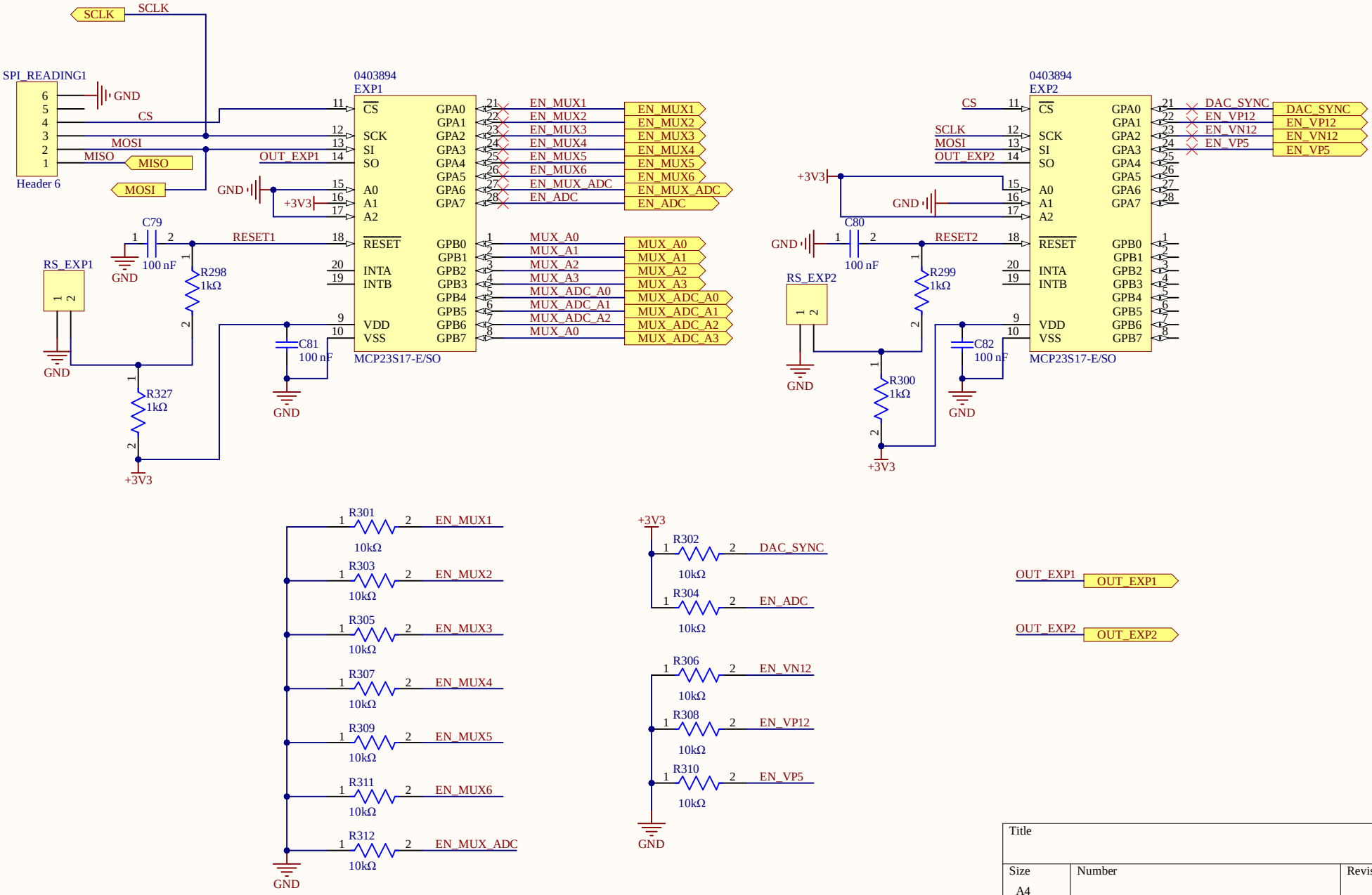

GND_IN1
R193
680 kΩ
DV_GND1
R201
1kΩ
GND_IN2
R194
680 kΩ
DV_GND2
R202
1kΩ
GND_IN3
R195
680 kΩ
DV_GND3
R203
1kΩ
GND_IN4
R196
680 kΩ
DV_GND4
R204
1kΩ
GND_IN5
R197
680 kΩ
DV_GND5
R205
1kΩ
GND_IN6
R198
680 kΩ
DV_GND6
R206
1kΩ
GND_IN7
R199
680 kΩ
DV_GND7
R207
1kΩ
GND_IN8
R200
680 kΩ
DV_GND8
R208
1kΩ
GND_IN9
R209
680 kΩ
DV_GND9
R217
1kΩ
GND_IN10
R210
680 kΩ
DV_GND10
R218
1kΩ
GND_IN11
R211
680 kΩ
DV_GND11
R219
1kΩ
GND_IN12
R212
680 kΩ
DV_GND12
R220
1kΩ
GND_IN13
R213
680 kΩ
DV_GND13
R221
1kΩ
GND_IN14
R214
680 kΩ
DV_GND14
R222
1kΩ
GND_IN15
R215
680 kΩ
DV_GND15
R223
1kΩ
GND_IN16
R216
680 kΩ
DV_GND16
R224
1kΩ
GND_IN17
R225
680 kΩ
DV_GND17
R229
1kΩ
GND_IN18
R226
680 kΩ
DV_GND18
R230
1kΩ
GND_IN19
R227
680 kΩ
DV_GND19
R231
1kΩ
GND_IN20
R228
680 kΩ
DV_GND20
R232
1kΩ
GND_IN21
R233
680 kΩ
DV_GND21
R237
1kΩ
GND_IN22
R234
680 kΩ
DV_GND22
R238
1kΩ
GND_IN23
R235
680 kΩ
DV_GND23
R239
1kΩ
GND_IN24
R236
680 kΩ
DV_GND24
R240
1kΩ
GND
GND_IN[1..24]
GND_IN1
GND_IN2
GND_IN3
GND_IN4
GND_IN5
GND_IN6
GND_IN7
GND_IN8
GND_IN9
GND_IN10
GND_IN11
GND_IN12
GND_IN13
GND_IN14
GND_IN15
GND_IN16
GND_IN17
GND_IN18
GND_IN19
GND_IN20
GND_IN21
GND_IN22
GND_IN23
GND_IN24
DV_GND[1..24]
DV_GND1
DV_GND2
DV_GND3
DV_GND4
DV_GND5
DV_GND6
DV_GND7
DV_GND8
DV_GND9
DV_GND10
DV_GND11
DV_GND12
DV_GND13
DV_GND14
DV_GND15
DV_GND16
DV_GND17
DV_GND18
DV_GND19
DV_GND20
DV_GND21
DV_GND22
DV_GND23
DV_GND24
Title
Size
A4
Number
Revision
Date: 9/16/2022
Sheet of
File: C:\Users\..\IN_DIV5.SchDoc
Drawn By:


GND_IN25
R241
680 kΩ
DV_GND25
R249
1kΩ
GND
GND_IN26
R242
680 kΩ
DV_GND26
R250
1kΩ
GND
GND_IN27
R243
680 kΩ
DV_GND27
R251
1kΩ
GND
GND_IN28
R244
680 kΩ
DV_GND28
R252
1kΩ
GND
GND_IN29
R245
680 kΩ
DV_GND29
R253
1kΩ
GND
GND_IN30
R246
680 kΩ
DV_GND30
R254
1kΩ
GND
GND_IN31
R247
680 kΩ
DV_GND31
R255
1kΩ
GND
GND_IN32
R248
680 kΩ
DV_GND32
R256
1kΩ
GND
GND_IN33
R257
680 kΩ
DV_GND33
R265
1kΩ
GND
GND_IN34
R258
680 kΩ
DV_GND34
R266
1kΩ
GND
GND_IN35
R259
680 kΩ
DV_GND35
R267
1kΩ
GND
GND_IN36
R260
680 kΩ
DV_GND36
R268
1kΩ
GND
GND_IN37
R261
680 kΩ
DV_GND37
R269
1kΩ
GND
GND_IN38
R262
680 kΩ
DV_GND38
R270
1kΩ
GND
GND_IN39
R263
680 kΩ
DV_GND39
R271
1kΩ
GND
GND_IN40
R264
680 kΩ
DV_GND40
R272
1kΩ
GND
GND_IN41
R273
680 kΩ
DV_GND41
R277
1kΩ
GND
GND_IN42
R274
680 kΩ
DV_GND42
R278
1kΩ
GND
GND_IN43
R275
680 kΩ
DV_GND43
R279
1kΩ
GND
GND_IN44
R276
680 kΩ
DV_GND44
R280
1kΩ
GND
GND_IN45
R281
680 kΩ
DV_GND45
R285
1kΩ
GND
GND_IN46
R282
680 kΩ
DV_GND46
R286
1kΩ
GND
GND_IN47
R283
680 kΩ
DV_GND47
R287
1kΩ
GND
GND_IN48
R284
680 kΩ
DV_GND48
R288
1kΩ
GND
GND_IN[25..48]
GND_IN[25..48]
GND_IN25
GND_IN26
GND_IN27
GND_IN28
GND_IN29
GND_IN30
GND_IN31
GND_IN32
GND_IN33
GND_IN34
GND_IN35
GND_IN36
GND_IN37
GND_IN38
GND_IN39
GND_IN40
GND_IN41
GND_IN42
GND_IN43
GND_IN44
GND_IN45
GND_IN46
GND_IN47
GND_IN48
DV_GND[24..48]
DV_GND[24..48]
DV_GND25
DV_GND26
DV_GND27
DV_GND28
DV_GND29
DV_GND30
DV_GND31
DV_GND32
DV_GND33
DV_GND34
DV_GND35
DV_GND36
DV_GND37
DV_GND38
DV_GND39
DV_GND40
DV_GND41
DV_GND42
DV_GND43
DV_GND44
DV_GND45
DV_GND46
DV_GND47
DV_GND48
Title
Size
A4
Number
Revision
Date:
9/16/2022
Sheet of
File:
C:\Users\..\IN_DIV6.SchDoc
Drawn By:

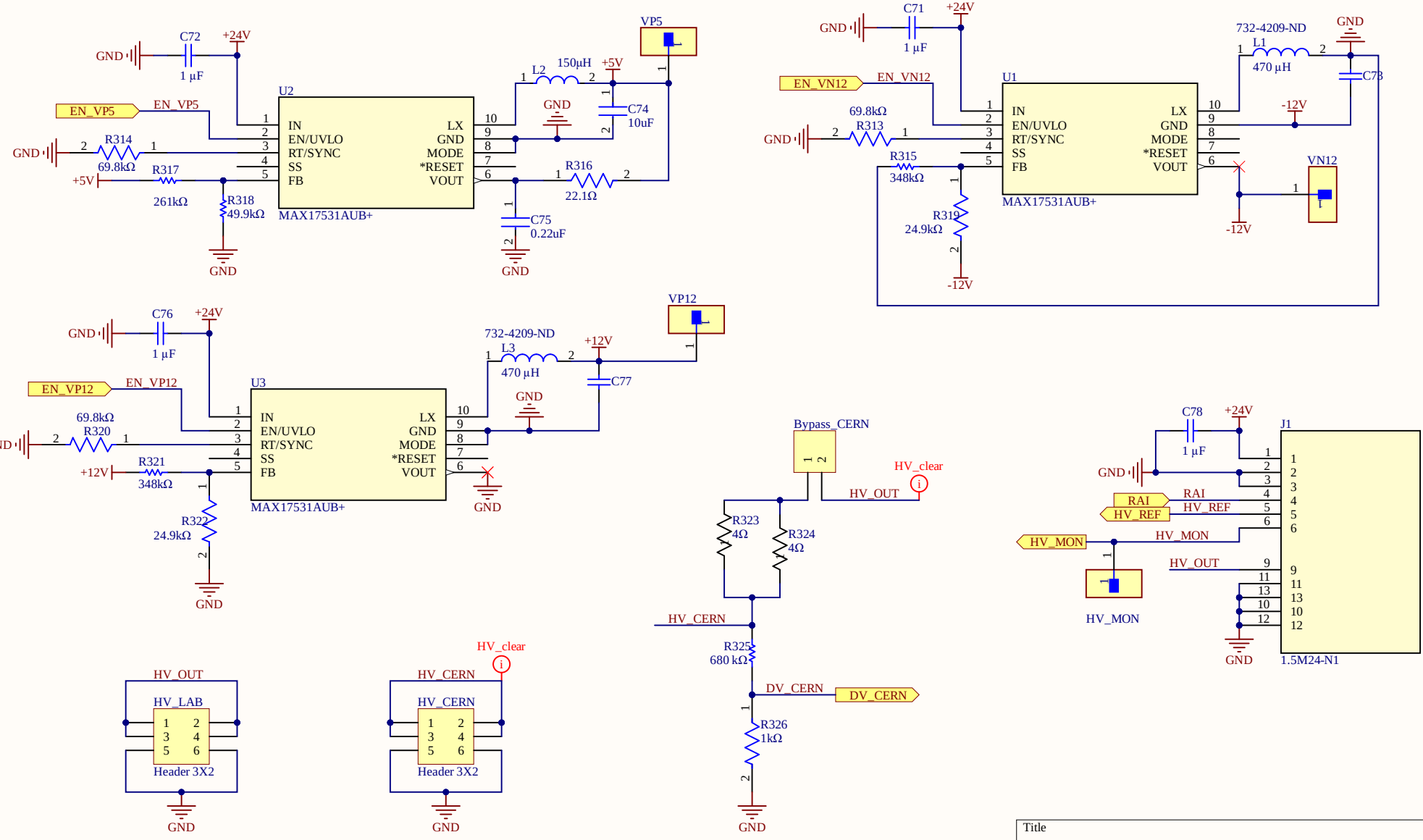

296-47544-1-ND
MUX1
MUX506IDWR
C49
C51
+12V
-12V
100 nF
GND
OUT_MUX1
EN_MUX1
MUX_A0
MUX_A1
MUX_A2
MUX_A3
DV_GND1
DV_GND2
DV_GND3
DV_GND4
DV_GND5
DV_GND6
DV_GND7
DV_GND8
DV_GND9
DV_GND10
DV_GND11
DV_GND12
DV_GND13
DV_GND14
DV_GND15
DV_GND16
296-47544-1-ND
MUX2
MUX506IDWR
C50
C52
OUT_MUX2
EN_MUX2
DV_IN9
DV_IN4
DV_IN10
DV_IN3
DV_IN11
DV_IN2
DV_IN12
DV_IN1
DV_GND17
DV_GND18
DV_GND19
DV_GND20
DV_GND21
DV_GND22
DV_GND23
DV_GND24
296-47544-1-ND
MUX3
MUX506IDWR
C53
C55
OUT_MUX3
EN_MUX3
DV_IN17
DV_IN20
DV_IN18
DV_IN19
DV_IN7
DV_IN6
DV_IN8
DV_IN5
DV_IN21
DV_IN16
DV_IN22
DV_IN15
DV_IN23
DV_IN14
DV_IN24
DV_IN13
296-47544-1-ND
MUX4
MUX506IDWR
C54
C56
OUT_MUX4
EN_MUX4
DV_IN31
DV_IN30
DV_IN32
DV_IN29
DV_IN33
DV_IN28
DV_IN34
DV_IN27
DV_IN43
DV_IN42
DV_IN44
DV_IN41
DV_IN45
DV_IN40
DV_IN46
DV_IN39
MUX5
MUX506IDWR
C57
C59
OUT_MUX5
EN_MUX5
DV_IN35
DV_IN26
DV_IN36
DV_IN25
DV_GND36
DV_GND35
DV_GND34
DV_GND33
DV_IN47
DV_IN38
DV_IN48
DV_IN37
DV_GND43
DV_GND44
DV_GND45
DV_GND46
MUX6
MUX506IDWR
C58
C60
OUT_MUX6
EN_MUX6
DV_GND30
DV_GND29
DV_GND28
DV_GND27
DV_GND26
DV_GND25
DV_GND32
DV_GND31
DV_GND47
DV_GND48
DV_GND37
DV_GND38
DV_GND39
DV_GND40
DV_GND41
DV_GND42
DV_IN[1..48]
DV_IN1
DV_IN2
DV_IN3
DV_IN4
DV_IN5
DV_IN6
DV_IN7
DV_IN8
DV_IN9
DV_IN10
DV_IN11
DV_IN12
DV_IN13
DV_IN14
DV_IN15
DV_IN16
DV_IN17
DV_IN18
DV_IN19
DV_IN20
DV_IN21
DV_IN22
DV_IN23
DV_IN24
DV_IN25
DV_IN26
DV_IN27
DV_IN28
DV_IN29
DV_IN30
DV_IN31
DV_IN32
DV_IN33
DV_IN34
DV_IN35
DV_IN36
DV_IN37
DV_IN38
DV_IN39
DV_IN40
DV_IN41
DV_IN42
DV_IN43
DV_IN44
DV_IN45
DV_IN46
DV_IN47
DV_IN48
DV_GND[1..48]
DV_GND1
DV_GND2
DV_GND3
DV_GND4
DV_GND5
DV_GND6
DV_GND7
DV_GND8
DV_GND9
DV_GND10
DV_GND11
DV_GND12
DV_GND13
DV_GND14
DV_GND15
DV_GND16
DV_GND17
DV_GND18
DV_GND19
DV_GND20
DV_GND21
DV_GND22
DV_GND23
DV_GND24
DV_GND25
DV_GND26
DV_GND27
DV_GND28
DV_GND29
DV_GND30
DV_GND31
DV_GND32
DV_GND33
DV_GND34
DV_GND35
DV_GND36
DV_GND37
DV_GND38
DV_GND39
DV_GND40
DV_GND41
DV_GND42
DV_GND43
DV_GND44
DV_GND45
DV_GND46
DV_GND47
DV_GND48
Title
Size
A4
Number
Revision
Date: 9/16/2022
File: C:\Users\..\MUXs.SchDoc
Sheet of
Drawn By:


+12V
296-47544-1-ND
MUX_ADC
OUT_MUX_ADC
GND
-12V
C61
100 nF
C62
100 nF
GND
VDD
NC
NC
S16
S15
S14
S13
S12
S11
S10
S9
GND
NC
A3
D
VSS
S8
S7
S6
S5
S4
S3
S2
S1
EN
A0
A1
A2
SHIELD2
SHIELD1
OUT_MUX6
OUT_MUX5
OUT_MUX4
OUT_MUX3
OUT_MUX2
OUT_MUX1
EN_MUX_ADC
MUX_ADC_A0
MUX_ADC_A1
MUX_ADC_A2
DV_CERN
HV_MON
SHIELD4
SHIELD3
GND
MUX_ADC_A3
MUX_A0
MUX506IDWR
296-48914-6-ND
595-INA828IDR
INA
C63
+12V
GND
1 uF
OUT_MUX_ADC
+VS
+IN
-IN
RG
RG
OUT
REF
-VS
ADC_IN
C64
1 uF
C65
1 uF
-12V
GND
INA828IDR
DAC_OUT
DAC
SCLK
MOSI
DAC_SYNC
SCLK
DIN
SYNC
VOUT
AVDD/VREF
GND
DAC_OUT
+5V
GND
SOFT_ADJ
Header 2X2
RAI
R_P1
3386P-1-104LF
GND
HARD_ADJ
HV_REF
HV_REF
Header 2
MISO
OUT_ADC
OUT_EXP1
OUT_EXP2
MISO
Header 3X2
GND
C66
100 nF
4.7UF 10% 16V
399-3697-1-ND
C67
+3V3
MAX1240CCSA+-ND
ADC
ADC_IN
+3V3
VDD
AIN
*SHDN
REF
SCLK
*CS
DOUT
GND
SCLK
EN_ADC
OUT_ADC
ADC_IN
C68
4.7UF 10% 16V
GND
MAX1240CCSA+
GND
ADC_OUT
Title
Size
A4
Number
Revision
Date:
9/16/2022
Sheet of
File:
C:\Users\..\Digital1.SchDoc
Drawn By:


732-618025231221-ND
CONN1
618025231221
M1
1 HV_IN1
14 GND_IN1
2 HV_IN2
15 GND_IN2
3 HV_IN3
16 GND_IN3
4 HV_IN4
17 GND_IN4
5 HV_IN5
18 GND_IN5
6 HV_IN6
19 GND_IN6
7 HV_IN7
20 GND_IN7
8 HV_IN8
21 GND_IN8
9 HV_IN9
22 GND_IN9
10 HV_IN10
23 GND_IN10
11 HV_IN11
24 GND_IN11
12 HV_IN12
25 GND_IN12
13 S1
M2
S1
Header 2X2
GND
R289
680 kΩ
SHIELD1
SHIELD1
R292
1kΩ
GND
732-618025231221-ND
CONN2
618025231221
M1
1 HV_IN13
14 GND_IN13
2 HV_IN14
15 GND_IN14
3 HV_IN15
16 GND_IN15
4 HV_IN16
17 GND_IN16
5 HV_IN17
18 GND_IN17
6 HV_IN18
19 GND_IN18
7 HV_IN19
20 GND_IN19
8 HV_IN20
21 GND_IN20
9 HV_IN21
22 GND_IN21
10 HV_IN22
23 GND_IN22
11 HV_IN23
24 GND_IN23
12 HV_IN24
25 GND_IN24
13 S2
M2
S2
Header 2X2
GND
R290
680 kΩ
SHIELD2
SHIELD2
R293
1kΩ
GND
732-618025231221-ND
CONN3
618025231221
M1
1 HV_IN25
14 GND_IN25
2 HV_IN26
15 GND_IN26
3 HV_IN27
16 GND_IN27
4 HV_IN28
17 GND_IN28
5 HV_IN29
18 GND_IN29
6 HV_IN30
19 GND_IN30
7 HV_IN31
20 GND_IN31
8 HV_IN32
21 GND_IN32
9 HV_IN33
22 GND_IN33
10 HV_IN34
23 GND_IN34
11 HV_IN35
24 GND_IN35
12 HV_IN36
25 GND_IN36
13 S3
M2
S3
Header 2X2
GND
R291
680 kΩ
SHIELD3
SHIELD3
R294
1kΩ
GND
732-618025231221-ND
CONN4
618025231221
M1
1 HV_IN37
14 GND_IN37
2 HV_IN38
15 GND_IN38
3 HV_IN39
16 GND_IN39
4 HV_IN40
17 GND_IN40
5 HV_IN41
18 GND_IN41
6 HV_IN42
19 GND_IN42
7 HV_IN43
20 GND_IN43
8 HV_IN44
21 GND_IN44
9 HV_IN45
22 GND_IN45
10 HV_IN46
23 GND_IN46
11 HV_IN47
24 GND_IN47
12 HV_IN48
25 GND_IN48
13 S4
M2
S4
Header 2X2
GND
R295
680 kΩ
SHIELD4
SHIELD4
R296
1kΩ
GND
HV_Clear
HV_IN[1..48]
HV_IN[1..48]
HV_IN1
HV_IN2
HV_IN3
HV_IN4
HV_IN5
HV_IN6
HV_IN7
HV_IN8
HV_IN9
HV_IN10
HV_IN11
HV_IN12
HV_IN13
HV_IN14
HV_IN15
HV_IN16
HV_IN17
HV_IN18
HV_IN19
HV_IN20
HV_IN21
HV_IN22
HV_IN23
HV_IN24
HV_IN25
HV_IN26
HV_IN27
HV_IN28
HV_IN29
HV_IN30
HV_IN31
HV_IN32
HV_IN33
HV_IN34
HV_IN35
HV_IN36
HV_IN37
HV_IN38
HV_IN39
HV_IN40
HV_IN41
HV_IN42
HV_IN43
HV_IN44
HV_IN45
HV_IN46
HV_IN47
HV_IN48
GND_IN[1..48]
GND_IN[1..48]
GND_IN1
GND_IN2
GND_IN3
GND_IN4
GND_IN5
GND_IN6
GND_IN7
GND_IN8
GND_IN9
GND_IN10
GND_IN11
GND_IN12
GND_IN13
GND_IN14
GND_IN15
GND_IN16
GND_IN17
GND_IN18
GND_IN19
GND_IN20
GND_IN21
GND_IN22
GND_IN23
GND_IN24
GND_IN25
GND_IN26
GND_IN27
GND_IN28
GND_IN29
GND_IN30
GND_IN31
GND_IN32
GND_IN33
GND_IN34
GND_IN35
GND_IN36
GND_IN37
GND_IN38
GND_IN39
GND_IN40
GND_IN41
GND_IN42
GND_IN43
GND_IN44
GND_IN45
GND_IN46
GND_IN47
GND_IN48
Header 4X2
24V
GND
EG1903-ND
SW1
EG1218
+24V
R297
715Ω
160-1423-2-ND
D1
C69
100µF
C70
2.2µF
GND
+3V3
3V3
Header 3X2
GND
Title
Size
A4
Number
Revision
Date: 9/16/2022
File: C:\Users\..\Connectors.SchDoc
Sheet of
Drawn By:



Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\EXPs.SchDoc		Drawn By:



Title		
Size: A4	Number	Revision
Date: 9/16/2022		Sheet of
File: C:\Users\..\Converters.SchDoc		Drawn By:

Appendix B

Cable Test Board List of Components

Name	Description	Footprint	Quantity
ERJ-2RKF3483X		RESC1005X40X25LL05T05	2
GRM21BR61C105KA01L		CAPC2013X135X45ML10T25	3
HV732BTTD3303F	Thick Film Resistor SMD 0.25W 680K 1% 500 VOLTS	RESC3216X70X40LL20T20	101
RK73H2ATTD4992F		RESC2013X60X30ML20T20	1
VJ0603Y104KXQCW1BC		CAPC1608X95X40ML15T15	2
MUX506IDWR	1-pA on-state leakage current, 36-V, 16: 1, 1-channel precision analog multiplexer 28-SOIC -40 to 125	DW28_TEX	5
DAC6311IDCKT	10-Bit, Single Channel, 80 uA, 1.8 to 5.5 V DAC, -40 to 125 degC, 6-pin SOT23 (DCK6), Green (RoHS & no Sb/Br)	TI-DCK6_V	1
MCP23S17-E/SO	16-Bit I/O Expander with Serial Interface, 28-Pin SOIC, Extended Temperature	SOIC-SO28_L	2
MAX17531AUB+	42V, Low IQ Synchronous Buck Regulator Enables High Efficiency	21-0061L_MXM	3
261K 1% 2010(5025)	261K 0.75W 1% 2010 (5025 Metric) SMD	RESC2010(5025)_N	1
MAX1240CCSA+	ADC Single SAR 73ksps 12-bit Serial 8-Pin SOIC N	21-0041B_8_MXM	1
UCL1V101MNL1GS	Aluminum Electrolytic Capacitor, 100 uF, 35 V, 20%, -55 to 105 degC, 2-Pin SMD, RoHS, Tape and Reel	NIC-UCL1V101MNL1GS_L	1
CAP 4.7uF 6.3V 0603(1608)	CAP 4.7uF 6.3V -20% to +80% 0603 (1608 Metric) Thickness 1mm SMD	CAPC0603(1608)100_M	2
T491A475K016AT	CAP TANT 4.7UF 10% 16V 1206	FP-T491A475K016AT-MFG	2
GRM155R71C224KA12D	Chip Multilayer Ceramic Capacitors for General Purpose, 0402, 0.22uF, X7R, 15%, 10%, 16V	FP-GRM155-0_05-IPC_C	1
GRM21BR70J106KA73K	Chip Multilayer Ceramic Capacitors for General Purpose, 0805, 10uF, X7R, 15%, 10%, 6.3V	FP-GRM21B-0_15-IPC_C	1
PBC01SAAN	CONN HEADER VERT 1POS	FP-PBC01SAAN-MFG	7
Header 2	Header, 2-Pin	HDR1X2	4
Header 2X2	Header, 2-Pin, Dual row	HDR2X2	53
Header 3X2	Header, 3-Pin, Dual row	HDR2X3	4
Header 4X2	Header, 4-Pin, Dual row	HDR2X4	1
Header 6	Header, 6-Pin	HDR1X6	1
INA828IDR	IC INST AMP 1 CIRCUIT 8SOIC	FP-D0008A-MFG	1
LTST-C171GKT	LED GREEN CLEAR CHIP SMD	FP-LTST-C171GKT-MFG	1
618025231221	Male PCB Connector with Hex Screw WR-DSUB, 8.08 mm, Angled, 25 pins	618025231221	4
C1608X5R1E225K080AB	Multilayer Ceramic Capacitors 2.2μF ±10% 25V X5R SMD 0603	FP-C1608-080-0_1-MFG	1
1.5M24-N1	No Description Available	CONN9_M24-N1_ULT	1
MUX506IDWR	No Description Available	DW28_TEX	2
HMK325B7104MF-T	None	FP-1210-L_3_2_0_3-W_2_5_0-MFG	65
UMK212B7105KG-T	None	CAPC2013X135X50ML10T25	4
RT0402DRE0769K8L	RES SMD 69.8K OHM 0.5% 1/16W	FP-RT0402-MFG	3
CRCW06037M87FKEB	RES Thick Film, 7.87MΩ, 1%, 0.1W, 100ppm/°C, 0603	FP-CRCW0603-e3-IPC_C	48
CRCW040210K0JNED	RES Thick Film, 10kΩ, 5%, 0.063W, 200ppm/°C, 0402	FP-CRCW0402-e3-IPC_B	12
CRCW060324K9FKEB	RES Thick Film, 24.9kΩ, 1%, 0.1W, 100ppm/°C, 0603	FP-CRCW0603-e3-IPC_C	2
7447471471	Shielded Radial Leaded Wire Wound Inductor WE-TIS, L=470 μH	WE-TIS 1111	2
EG1218	Slide Switch, 500 V, -20 to 70 degC, 3-Pin THD, RoHS	ESWI-EG1218_V	1
RN73C2A22R1BTDF	SMD Chip Resistor, Thin Film, 22.1Ω, 100 V, 0805 [2012 Metric], 100 mW, 0.1%, RN73 Series	FP-RN732A-MFG	1
3386P-1-104LF	Square Trimpot(R) Trimming Potentiometer, 100 KOhm, +/- 10%, 0.5 W, -55 to 125 degC, 3-Pin THD, RoHS, Tube	BOUR-3386P	1
CR0805-FX-7150ELF	Thick Film Chip Resistors 0805 715Ω 0.125W 1% 100ppm/°C	FP-CR0805-IPC_B	1
CR1206-FX-1001ELF	Thick Film Chip Resistors 1206 1kΩ 0.25W 1% 100ppm/°C	FP-CR1206-IPC_A	153
4Ω	Through Hole Resistor, 40 Series, 4 Ohm, 1 W, 1%, 150 V, Axial Leaded Rohs Compliant: Yes	RES_41_OHM	2
LQH43MN151J03L	Wire Wound Ferrite Inductor for General Circuits 150μH ±5% 3.7Ω 130mA 1812	FP-LQH43CN_03-MFG	1

Appendix C

Cable Test Board Software Interface Code

C.1 DAC6311 Class

```
import time
import spidev
from MCP23S17 import MCP23S17

class DAC6311(object):
    """
    This class is an abstraction of the DAC6311, which is a 10-Bit Single Channel DAC,
        for the
    Raspberry Pi. It requires the Python modules spidev and RPi.GPIO.
    """
    OP_MODES = {
        '0' : 0b00,
        '1' : 0b01,
        '2' : 0b10,
        '3' : 0b11
    }

    def __init__(self, spiDevice = spidev.SpiDev(), SYNC = -1, MCP = MCP23S17)
        """
        Constructor

        Keyword Arguments:
        SYNC -- pin of the SYNC -- pin of the MCP23S17 that will be used as SYNC.
                The write sequence starts by bringing SYNC low
        MCP -- the port expander that will control the SYNC pin of the DAC
        """

        self.__spi = spiDevice
        self.__spiMode = 0b01
        self.__SYNC = SYNC
        self.__MCP = MCP

    def writeRegister(self, op_mode, data): #alterar
        """
```

```
        Write into the input shift register

        Keyword Arguments:
        data -- 10-bit data to be written in the input register.
                Must be an integer less than 1024
        op_mode -- integer between 0 and 3 that determines the DAC's mode
                   of operation. 0 is normal operation, 1-3 are power-down modes.
        """

        assert type(data) is int
        assert type(op_mode) is int
        assert op_mode in range(0,4)
        assert data < 1024


        prefix = DAC6311.OP_MODES[op_mode]
        feature = 0b0000

        self.__sendWord(prefix, data, feature)

    def __sendWord(self, prefix, data, feature):
        """
        Enables communication with the DAC by lowering the SYNC line,
        sends the word to the DAC via spidev xfer2 function which takes a list
        of bytes.

        Keyword Arguments:
        prefix -- 2-bit word that determines de mode of operation.
        data -- 10-bit data word to be sent
        feature -- 4-feature bits which are to be ignored
        """
        data1 = data >> 4
        first_word = (prefix << 6) | data1
        second_word = ((data & 0b00001111) << 4) | feature

        self.__MCP.digitalWrite(self.__SYNC, self.__MCP.LEVEL_LOW)
        self.__spi.mode = self.__spiMode
        self.__spi.xfer2([first_word, second_word])
        self.__MCP.digitalWrite(self.__SYNC, self.__MCP.LEVEL_HIGH)
```

C.2 CableTest Class

```
from MCP23S17 import MCP23S17
from DAC6311 import DAC6311
from MAX1240 import MAX1240
import spidev
import time
import RPi.GPIO as GPIO
import CTBMap as map
```

```
from Reading_code import Reading_code
from Reading_code2 import Reading_code2


class CABLETEST(object):

    def __init__(self, spiDevice = spidev.SpiDev(), iGPIO = GPIO, CS_PIN = 40, board =
        1):

        self.__spi = spiDevice
        self.__GPIO = iGPIO
        self.__CS_PIN = CS_PIN
        self.__GPIO.setup(self.__CS_PIN, self.__GPIO.OUT)
        self.__HVRBoard = board
        self.__MCP = dict()

    def initBoard(self):
        """
        Port Expander, ADC, DAC and HV Reading Board configuration.
        """

        for n in map.MCP_CONFIG_MAP:
            address = map.MCP_CONFIG_MAP[n][0] ##port expander hardware address
            direction = map.MCP_CONFIG_MAP[n][1] #port expander GPIOs direction
            cs = self.__CS_PIN
            self.__MCP[n] = MCP23S17(self.__spi, address, -1, cs)
            self.__MCP[n].configureMCP()
            self.__MCP[n].setDirectionForAll(direction)

        self.enableCommunication(True)

        ###Setting GPIOs deafult start values
        self.__MCP1_default = map.MCP_CONFIG_MAP[1][2]
        self.__MCP2_default = map.MCP_CONFIG_MAP[2][2]
        self.__HVR1_MCP1_default = map.MCP_CONFIG_MAP[3][2]
        self.__HVR1_MCP2_default = map.MCP_CONFIG_MAP[4][2]
        self.__HVR2_MCP_default = map.MCP_CONFIG_MAP[5][2]

        self.__MCP[0].writeGPIO(self.__MCP1_default)
        self.__MCP[1].writeGPIO(self.__MCP2_default)

        self.__ADC_MCP = map.ADC_MCP
        self.__HVR1_MCP1 = map.HVR1_MCP1
        self.__HVR1_MCP2 = map.HVR1_MCP2
        self.__HVR2_MCP = map.HVR2_MCP

        mcp_SYNC = map.DAC_SYNC[1]
        mcp_SYNC_PIN = map.DAC_SYNC[0]

        self.__DAC = DAC6311(self.__spi, self.__mcp_SYNC_PIN, self.__MCP[mcp_SYNC])
        self.__ADC = MAX1240(self.__spi, map.EN_ADC, self.__MCP[self.__ADC_MCP])
```

```
        if self.__HVRBoard == 1:
            self.__HVReading = Reading_code(self.__spi, self.__MCP[self.__HVR1_MCP1],
                self.__MCP[self.__HVR1_MCP2], self.__GPIO)
            self.__MCP[2].writeGPIO(self.__HVR1_MCP1_default)
            self.__MCP[3].writeGPIO(self.__HVR1_MCP2_default)

        else:
            self.__HVReading = Reading_code2(self.__spi, self.__MCP[self.__HVR2_MCP],
                self.__GPIO)
            self.__MCP[4].writeGPIO(self.__HVR2_MCP_default)

        self.enableCommunication(False)


    def enableCommunication(self, onOff):
        """
        Enables or disables communication with this board.
        Keyword Arguments:
            onOff (bool): if true communication is enabled, if false
            communication is disabled
        """
        assert type(onOff) is bool

        if onOff == True:
            self.__GPIO.output(self.__CS_PIN,self.__GPIO.LOW)
        else:
            self.__GPIO.output(self.__CS_PIN,self.__GPIO.HIGH)


    def enableDisableSupplies(self, supply, onOff):
        """
        Enables or disables one power supply.

        Keyword Arguments:
        supply -- string with the name of the power supply, as it addressed in
                  CTBMap.
        onOff -- Boolean representing the desired state (on or off)
        """

        assert supply == 'VP12' or supply == 'VN12' or supply == 'V5'
        assert onOff == True or onOff == False

        self.__GPIO.output(self.__CS_PIN, self.__GPIO.LOW)

        expander = map.SUPPLY_LV_MAP[supply]
        self.__MCP[expander[0]].digitalWrite(expander[1], onOff)

        self.__GPIO.output(self.__CS_PIN, self.__GPIO.HIGH)
```

```
def readHVChannel(self, channel):
    """
    Performs one reading of a specific HV channel.
    Keyword Arguments:
    channel (int) -- from 1 to 48

    Returns:
    value (int) -- The reading from the ADC. 12-bit integer.
    """

    assert channel in range(1,49)

    self.enableCommunication(True)

    hv_Selection = map.HVCHANNEL2CHIP_MAP[channel]
    muxEnable = hv_Selection[0]
    muxAddress = hv_Selection[1]
    muxFinalAddress = hv_Selection[2]

    #Building word command to transfer to the Port Expander containing the MUXs
    #and the ADC
    GPIOA = 0b11000000 | muxEnable ##last digits enable ADC and final MUX
    GPIOB = (muxFinalAddress << 4) | muxAddress
    GPIO_word = (GPIOB << 8) | GPIOA
    self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

    reading = self.__ADC.readVoltage()

    self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

    self.enableCommunication(False)

    return reading


def readGNDChannel(self, channel):
    """
    Performs one reading of a specific GND channel.
    Keyword Arguments:
    channel (int) -- from 1 to 48

    Returns:
    value (int) -- The reading from the ADC. 12-bit integer.
    """

    assert channel in range(1,49)

    self.enableCommunication(True)

    gnd_Selection = map.GNDCHANNEL2CHIP_MAP[channel]
    muxEnable = gnd_Selection[0]
```

```
        muxAddress = gnd_Selection[1]
        muxFinalAddress = gnd_Selection[2]

        GPIOA = 0b11000000 | muxEnable
        GPIOB = (muxFinalAddress << 4) | muxAddress
        GPIO_word = (GPIOB << 8) | GPIOA
        self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

        reading = self.__ADC.readVoltage()

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return reading


    def setHV(self, hv):
        """
        Adjusts the high voltage output of the DC/DC converter.

        Keyword Arguments:
        hv (int) -- Value between 0 to 1500 corresponding to the value in Volts
        of the desired voltage (max. 1500 V).
        """
        assert hv in range(0, 1500)

        self.enableCommunication(True)

        hvOrder = (hv*1024)//1500

        self.__DAC.writeRegister(0, hvOrder)

        self.enableCommunication(False)


    def readSupplyHV(self):
        """
        Performs one reading of the HV supplied.

        Returns:
        value (int) -- The reading from the ADC. 12-bit integer.
        """

        self.enableCommunication(True)

        hv_supply = map.SUPPLY_HV_MAP

        muxEnable = hv_supply[0]
        muxAddress = hv_supply[1]
        muxFinalAddress = hv_supply[2]
```

```
        GPIOA = 0b11000000 | muxEnable
        GPIOB = (muxFinalAddress << 4) | muxAddress
        GPIO_word = (GPIOB << 8) | GPIOA
        self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

        reading = self.__ADC.readVoltage()

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return reading


    def setHVCable(self, cable):
        """
        Selects the cable to which HV will be connected.
        Keyword Arguments:
        cable (int) -- cable channel number
        """

        self.enableCommunication(True)

        self.__HVReading.reading_func(1, cable, -1)

        self.enableCommunication(False)

    def readAllHVs(self):
        """
        Performs a reading to all HV channels of this board.

        Returns:
            values (list(int)): a list with all the readings of all channels.
            All readings are returned as values of 12 bits.
    """

        values = list()

        self.enableCommunication(True)

        for ch in range(1,49):
            hv_Selection = map.HVCHANNEL2CHIP_MAP[ch]
            muxEnable = hv_Selection[0]
            muxAddress = hv_Selection[1]
            muxFinalAddress = hv_Selection[2]

            GPIOA = 0b11000000 | muxEnable
            GPIOB = (muxFinalAddress << 4) | muxAddress
            GPIO_word = (GPIOB << 8) | GPIOA
            self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)
```

```
            values.append(self.ADC.readVoltage())

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return values


    def readAllGNDs(self):
        """
        Performs a reading to all GND channels of this board.

        Returns:
            values (list(int)): a list with all the readings of all channels.
            All readings are returned as values of 12 bits.
    """

        values = list()

        self.enableCommunication(True)

        for ch in range(1,49):
            gnd_Selection = map.GNDCHANNEL2CHIP_MAP[ch]
            muxEnable = gnd_Selection[0]
            muxAddress = gnd_Selection[1]
            muxFinalAddress = gnd_Selection[2]

            GPIOA = 0b11000000 | muxEnable
            GPIOB = (muxFinalAddress << 4) | muxAddress
            GPIO_word = (GPIOB << 8) | GPIOA
            self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

            values.append(self.ADC.readVoltage())

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return values


    def readShield(self, shield_num):
        """
        Performs a reading of the shield voltage.

        Keyword Arguments:
        shield_num (int) -- shield cable number (1 to 4)
        """
```

```
        assert shield_num in range(1,5)

        self.enableCommunication(True)

        shield_Selection = map.SHIELDSIGNAL2CHIP_MAP[shield_num]
        muxEnable = shield_Selection[0]
        muxAddress = shield_Selection[1]
        muxFinalAddress = shield_Selection[2]

        GPIOA = 0b11000000 | muxEnable
        GPIOB = (muxFinalAddress << 4) | muxAddress
        GPIO_word = (GPIOB << 8) | GPIOA
        self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

        reading = self.__ADC.readVoltage()

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return reading


    def readCERN(self):

        self.enableCommunication(True)

        cern_Selection = map.CERNSIGNAL2CHIP_MAP
        muxEnable = cern_Selection[0]
        muxAddress = cern_Selection[1]
        muxFinalAddress = cern_Selection[2]

        GPIOA = 0b11000000 | muxEnable
        GPIOB = (muxFinalAddress << 4) | muxAddress
        GPIO_word = (GPIOB << 8) | GPIOA
        self.__MCP[self.__ADC_MCP].writeGPIO(GPIO_word)

        reading = self.__ADC.readVoltage()

        self.__MCP[self.__ADC_MCP].writeGPIO(self.__MCP1_default)

        self.enableCommunication(False)

        return reading


    def readPortExpander(self, expanderNumber):
        """
        Performs a reading of the port expander's output register

        Keyword Arguments:
```

```
        expanderNumber (int) -- port expander number (0 or 1)
        """

        assert expanderNumber == 0 or expanderNumber == 1

        self.enableCommunication(True)

        read = self.__MCP[expanderNumber].readGPIO()

        self.enableCommunication(False)

        print("Port Expander ", expanderNumber, ": ", '{0:0{1}b}'.format(read, 16))

        return read


    def set_spiSpeed(self, speed):
        self.__spi.max_speed_hz = speed


    def closeBoard(self):
        """
        Sets the DAC value to 0 and disables all low voltages
        """

        self.enableCommunication(True)

        self.setHV(0)

        self.enableDisableSupplies('VP12', False)
        self.enableDisableSupplies('VN12', False)
        self.enableDisableSupplies('V5', False)

        self.enableCommunication(False)
```

C.3 CTBMap

```
from MCP23S17 import MCP23S17
import RPi.GPIO as GPIO

DAC_SYNC = [1,22]
ADC_MCP = 0
HVR1_MCP1 = 2
HVR1_MCP2 = 3
HVR2_MCP = 4
EN_ADC = 8

# MCP Number: MCP address | MCP GPIOs direction | MCP CS pin | MCP default GPIO word
MCP_CONFIG_MAP = {
```

```
1: [0b010, MCP23S17.DIR_OUTPUT, 0b1111111000000001],
2: [0b101, MCP23S17.DIR_OUTPUT, 0b1100000000000000],
3: [0b000, MCP23S17.DIR_OUTPUT, 0b0000000000000000],
4: [0b111, MCP23S17.DIR_OUTPUT, 0b0000000000000000],
5: [0b001, MCP23S17.DIR_OUTPUT, 0b0000000000000000]
}

# Signal Name : MUX Enable word 6 bits | MUX address word 4 bits | Final MUX output
    selector word 4 bits
HVCHANNEL2CHIP_MAP = {
"1": [2,8,1],
"2": [2,10,1],
"3": [2,12,1],
"4": [2,14,1],
"5": [4,8,2],
"6": [4,10,2],
"7": [4,11,2],
"8": [4,9,2],

"9": [2,15,1],
"10": [2,13,1],
"11": [2,11,1],
"12": [2,9,1],
"13": [4,0,2],
"14": [4,2,2],
"15": [4,4,2],
"16": [4,6,2],

"17": [4,15,2],
"18": [4,13,2],
"19": [4,12,2],
"20": [4,14,2],
"21": [4,7,2],
"22": [4,5,2],
"23": [4,3,2],
"24": [4,1,2],

"25": [16,12,4],
"26": [16,14,4],
"27": [8,8,3],
"28": [8,10,3],
"29": [8,12,3],
"30": [8,14,3],
"31": [8,15,3],
"32": [8,13,3],

"33": [8,11,3],
"34": [8,9,3],
"35": [16,15,4],
"36": [16,13,4],
"37": [16,4,4],
```

```
"38": [16,6,4],
"39": [8,0,3],
"40": [8,2,3],

"41": [8,4,3],
"42": [8,6,3],
"43": [8,7,3],
"44": [8,5,3],
"45": [8,4,3],
"46": [8,1,3],
"47": [16,7,4],
"48": [16,5,4]
}


GNDCHANNEL2CHIP_MAP = {
"1": [1,15,0],
"2": [1,14,0],
"3": [1,13,0],
"4": [1,12,0],
"5": [1,11,0],
"6": [1,10,0],
"7": [1,9,0],
"8": [1,8,0],

"9": [1,7,0],
"10": [1,6,0],
"11": [1,5,0],
"12": [1,4,0],
"13": [1,3,0],
"14": [1,2,0],
"15": [1,1,0],
"16": [1,0,0],

"17": [2,7,1],
"18": [2,6,1],
"19": [2,5,1],
"20": [2,4,1],
"21": [2,3,1],
"22": [2,2,1],
"23": [2,1,1],
"24": [2,0,1],

"25": [16,7,4],
"26": [16,6,4],
"27": [16,5,4],
"28": [16,4,4],
"29": [16,3,4],
"30": [16,2,4],
"31": [16,1,4],
"32": [32,9,5],
```

```
"33": [16,8,4],
"34": [16,9,4],
"35": [16,10,4],
"36": [16,11,4],
"37": [32,5,5],
"38": [32,4,5],
"39": [32,3,5],
"40": [32,2,5],

"41": [32,1,5],
"42": [32,0,5],
"43": [16,3,4],
"44": [16,2,4],
"45": [16,1,4],
"46": [16,0,4],
"47": [32,7,5],
"48": [32,6,5]
}


SHIELDSIGNAL2CHIP_MAP = {
'1': [0,0,6],
'2': [0,0,7],
'3': [0,0,8],
'4': [0,0,9]
}

CERNSIGNAL2CHIP_MAP = [0,0,11]

SUPPLY_LV_MAP = {
'VP12': [1,22]
'VN12': [1,23]
'V5': [1,24]
}

SUPPLY_HV_MAP = [0,0,10]
```

Appendix D

Cable Test Board Graphical User Interface Code

```
# -*- coding: utf-8 -*-

from CTB import CABLETEST
import spidev
import time
import RPi.GPIO as GPIO
from datetime import date

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtCore import pyqtSignal


class Ui_MainWindow(QtWidgets.QWidget):

    finished = pyqtSignal()

    def __init__(self, *args, **kwargs):

        QtWidgets.QWidget.__init__(self)

        self.__spi = spidev.SpiDev()
        self.__spi.open(0,0)
        self.__spi.max_speed_hz = 976000
        self.__spi.no_cs = True
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BOARD)

        self.__CS_PINS = ["18", "22", "29", "31", "36", "37"]
        self.__CS_PIN = None
        self.__HVR_BOARDS = ["1", "2"]
        self.__HVR_BOARD = None
        self.__ctbBoard = None
        self.__lcdDisplays = []
```

```
        self.__shieldDisplays = []
        self.__continueRun = False


    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1307, 976)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setMinimumSize(QtCore.QSize(1200, 830))
        self.centralwidget.setObjectName("centralwidget")
        self.scrollArea = QtWidgets.QScrollArea(self.centralwidget)
        self.scrollArea.setGeometry(QtCore.QRect(40, 10, 1241, 921))
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 1239, 919))
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
        self.gridLayout_5 = QtWidgets.QGridLayout(self.scrollAreaWidgetContents)
        self.gridLayout_5.setObjectName("gridLayout_5")
        self.groupBox_readHV = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
        self.groupBox_readHV.setMinimumSize(QtCore.QSize(350, 800))
        self.groupBox_readHV.setMaximumSize(QtCore.QSize(350, 16777215))
        font = QtGui.QFont()
        font.setPointSize(9)
        self.groupBox_readHV.setFont(font)
        self.groupBox_readHV.setObjectName("groupBox_readHV")
        self.scrollArea_readHV = QtWidgets.QScrollArea(self.groupBox_readHV)
        self.scrollArea_readHV.setEnabled(True)
        self.scrollArea_readHV.setGeometry(QtCore.QRect(30, 70, 300, 800))
        self.scrollArea_readHV.setMinimumSize(QtCore.QSize(300, 0))
        self.scrollArea_readHV.setMaximumSize(QtCore.QSize(300, 800))
        self.scrollArea_readHV.setWidgetResizable(True)
        self.scrollArea_readHV.setObjectName("scrollArea_readHV")
        self.scrollContents = QtWidgets.QWidget()
        self.scrollContents.setGeometry(QtCore.QRect(0, 0, 277, 1695))
        self.scrollContents.setObjectName("scrollContents")
        self.gridLayout = QtWidgets.QGridLayout(self.scrollContents)
        self.gridLayout.setObjectName("gridLayout")
        self.label_44 = QtWidgets.QLabel(self.scrollContents)
        self.label_44.setObjectName("label_44")
        self.gridLayout.addWidget(self.label_44, 33, 0, 1, 1)
        self.lcdNumber_16 = QtWidgets.QLCDNumber(self.scrollContents)
        self.lcdNumber_16.setObjectName("lcdNumber_16")
        self.gridLayout.addWidget(self.lcdNumber_16, 15, 1, 1, 1)
        self.label_19 = QtWidgets.QLabel(self.scrollContents)
        self.label_19.setObjectName("label_19")
        self.gridLayout.addWidget(self.label_19, 12, 0, 1, 1)
        self.lcdNumber_13 = QtWidgets.QLCDNumber(self.scrollContents)
        self.lcdNumber_13.setObjectName("lcdNumber_13")
        self.gridLayout.addWidget(self.lcdNumber_13, 12, 1, 1, 1)
        self.read_1 = QtWidgets.QPushButton(self.scrollContents)
```

```
self.read_1.setObjectName("read_1")
self.gridLayout.addWidget(self.read_1, 0, 2, 1, 1)
self.label_HV2 = QtWidgets.QLabel(self.scrollContents)
self.label_HV2.setObjectName("label_HV2")
self.gridLayout.addWidget(self.label_HV2, 1, 0, 1, 1)
self.lcdNumber_14 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_14.setObjectName("lcdNumber_14")
self.gridLayout.addWidget(self.lcdNumber_14, 13, 1, 1, 1)
self.lcdNumber_1 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_1.setObjectName("lcdNumber_1")
self.gridLayout.addWidget(self.lcdNumber_1, 0, 1, 1, 1)
self.lcdNumber_2 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_2.setObjectName("lcdNumber_2")
self.gridLayout.addWidget(self.lcdNumber_2, 1, 1, 1, 1)
self.label_HV1 = QtWidgets.QLabel(self.scrollContents)
self.label_HV1.setObjectName("label_HV1")
self.gridLayout.addWidget(self.label_HV1, 0, 0, 1, 1)
self.read_2 = QtWidgets.QPushButton(self.scrollContents)
self.read_2.setObjectName("read_2")
self.gridLayout.addWidget(self.read_2, 1, 2, 1, 1)
self.label_HV3 = QtWidgets.QLabel(self.scrollContents)
self.label_HV3.setObjectName("label_HV3")
self.gridLayout.addWidget(self.label_HV3, 2, 0, 1, 1)
self.lcdNumber_3 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_3.setObjectName("lcdNumber_3")
self.gridLayout.addWidget(self.lcdNumber_3, 2, 1, 1, 1)
self.read_3 = QtWidgets.QPushButton(self.scrollContents)
self.read_3.setObjectName("read_3")
self.gridLayout.addWidget(self.read_3, 2, 2, 1, 1)
self.label_16 = QtWidgets.QLabel(self.scrollContents)
self.label_16.setObjectName("label_16")
self.gridLayout.addWidget(self.label_16, 9, 0, 1, 1)
self.lcdNumber_12 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_12.setObjectName("lcdNumber_12")
self.gridLayout.addWidget(self.lcdNumber_12, 11, 1, 1, 1)
self.label_43 = QtWidgets.QLabel(self.scrollContents)
self.label_43.setObjectName("label_43")
self.gridLayout.addWidget(self.label_43, 32, 0, 1, 1)
self.lcdNumber_8 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_8.setObjectName("lcdNumber_8")
self.gridLayout.addWidget(self.lcdNumber_8, 7, 1, 1, 1)
self.label_42 = QtWidgets.QLabel(self.scrollContents)
self.label_42.setObjectName("label_42")
self.gridLayout.addWidget(self.label_42, 31, 0, 1, 1)
self.lcdNumber_15 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_15.setObjectName("lcdNumber_15")
self.gridLayout.addWidget(self.lcdNumber_15, 14, 1, 1, 1)
self.label_HV8 = QtWidgets.QLabel(self.scrollContents)
self.label_HV8.setObjectName("label_HV8")
self.gridLayout.addWidget(self.label_HV8, 7, 0, 1, 1)
self.lcdNumber_9 = QtWidgets.QLCDNumber(self.scrollContents)
```

```
self.lcdNumber_9.setObjectName("lcdNumber_9")
self.gridLayout.addWidget(self.lcdNumber_9, 8, 1, 1, 1)
self.label_18 = QtWidgets.QLabel(self.scrollContents)
self.label_18.setObjectName("label_18")
self.gridLayout.addWidget(self.label_18, 11, 0, 1, 1)
self.lcdNumber_10 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_10.setObjectName("lcdNumber_10")
self.gridLayout.addWidget(self.lcdNumber_10, 9, 1, 1, 1)
self.label_15 = QtWidgets.QLabel(self.scrollContents)
self.label_15.setObjectName("label_15")
self.gridLayout.addWidget(self.label_15, 8, 0, 1, 1)
self.label_22 = QtWidgets.QLabel(self.scrollContents)
self.label_22.setObjectName("label_22")
self.gridLayout.addWidget(self.label_22, 15, 0, 1, 1)
self.lcdNumber_7 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_7.setObjectName("lcdNumber_7")
self.gridLayout.addWidget(self.lcdNumber_7, 6, 1, 1, 1)
self.label_HV5 = QtWidgets.QLabel(self.scrollContents)
self.label_HV5.setObjectName("label_HV5")
self.gridLayout.addWidget(self.label_HV5, 4, 0, 1, 1)
self.lcdNumber_11 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_11.setObjectName("lcdNumber_11")
self.gridLayout.addWidget(self.lcdNumber_11, 10, 1, 1, 1)
self.label_20 = QtWidgets.QLabel(self.scrollContents)
self.label_20.setObjectName("label_20")
self.gridLayout.addWidget(self.label_20, 13, 0, 1, 1)
self.read_4 = QtWidgets.QPushButton(self.scrollContents)
self.read_4.setObjectName("read_4")
self.gridLayout.addWidget(self.read_4, 3, 2, 1, 1)
self.label_HV6 = QtWidgets.QLabel(self.scrollContents)
self.label_HV6.setObjectName("label_HV6")
self.gridLayout.addWidget(self.label_HV6, 5, 0, 1, 1)
self.label_HV7 = QtWidgets.QLabel(self.scrollContents)
self.label_HV7.setObjectName("label_HV7")
self.gridLayout.addWidget(self.label_HV7, 6, 0, 1, 1)
self.label_17 = QtWidgets.QLabel(self.scrollContents)
self.label_17.setObjectName("label_17")
self.gridLayout.addWidget(self.label_17, 10, 0, 1, 1)
self.lcdNumber_4 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_4.setObjectName("lcdNumber_4")
self.gridLayout.addWidget(self.lcdNumber_4, 3, 1, 1, 1)
self.label_21 = QtWidgets.QLabel(self.scrollContents)
self.label_21.setObjectName("label_21")
self.gridLayout.addWidget(self.label_21, 14, 0, 1, 1)
self.label_HV4 = QtWidgets.QLabel(self.scrollContents)
self.label_HV4.setObjectName("label_HV4")
self.gridLayout.addWidget(self.label_HV4, 3, 0, 1, 1)
self.lcdNumber_5 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_5.setObjectName("lcdNumber_5")
self.gridLayout.addWidget(self.lcdNumber_5, 4, 1, 1, 1)
self.lcdNumber_6 = QtWidgets.QLCDNumber(self.scrollContents)
```

```
self.lcdNumber_6.setObjectName("lcdNumber_6")
self.gridLayout.addWidget(self.lcdNumber_6, 5, 1, 1, 1)
self.lcdNumber_43 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_43.setObjectName("lcdNumber_43")
self.gridLayout.addWidget(self.lcdNumber_43, 42, 1, 1, 1)
self.lcdNumber_44 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_44.setObjectName("lcdNumber_44")
self.gridLayout.addWidget(self.lcdNumber_44, 43, 1, 1, 1)
self.label_93 = QtWidgets.QLabel(self.scrollContents)
self.label_93.setObjectName("label_93")
self.gridLayout.addWidget(self.label_93, 43, 0, 1, 1)
self.lcdNumber_45 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_45.setObjectName("lcdNumber_45")
self.gridLayout.addWidget(self.lcdNumber_45, 44, 1, 1, 1)
self.lcdNumber_47 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_47.setObjectName("lcdNumber_47")
self.gridLayout.addWidget(self.lcdNumber_47, 46, 1, 1, 1)
self.lcdNumber_46 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_46.setObjectName("lcdNumber_46")
self.gridLayout.addWidget(self.lcdNumber_46, 45, 1, 1, 1)
self.lcdNumber_48 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_48.setObjectName("lcdNumber_48")
self.gridLayout.addWidget(self.lcdNumber_48, 47, 1, 1, 1)
self.label_95 = QtWidgets.QLabel(self.scrollContents)
self.label_95.setObjectName("label_95")
self.gridLayout.addWidget(self.label_95, 45, 0, 1, 1)
self.label_94 = QtWidgets.QLabel(self.scrollContents)
self.label_94.setObjectName("label_94")
self.gridLayout.addWidget(self.label_94, 44, 0, 1, 1)
self.label_97 = QtWidgets.QLabel(self.scrollContents)
self.label_97.setObjectName("label_97")
self.gridLayout.addWidget(self.label_97, 47, 0, 1, 1)
self.label_96 = QtWidgets.QLabel(self.scrollContents)
self.label_96.setObjectName("label_96")
self.gridLayout.addWidget(self.label_96, 46, 0, 1, 1)
self.lcdNumber_42 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_42.setObjectName("lcdNumber_42")
self.gridLayout.addWidget(self.lcdNumber_42, 41, 1, 1, 1)
self.label_91 = QtWidgets.QLabel(self.scrollContents)
self.label_91.setObjectName("label_91")
self.gridLayout.addWidget(self.label_91, 41, 0, 1, 1)
self.label_92 = QtWidgets.QLabel(self.scrollContents)
self.label_92.setObjectName("label_92")
self.gridLayout.addWidget(self.label_92, 42, 0, 1, 1)
self.lcdNumber_17 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_17.setObjectName("lcdNumber_17")
self.gridLayout.addWidget(self.lcdNumber_17, 16, 1, 1, 1)
self.lcdNumber_26 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_26.setObjectName("lcdNumber_26")
self.gridLayout.addWidget(self.lcdNumber_26, 25, 1, 1, 1)
self.label_23 = QtWidgets.QLabel(self.scrollContents)
```

```
self.label_23.setObjectName("label_23")
self.gridLayout.addWidget(self.label_23, 16, 0, 1, 1)
self.lcdNumber_20 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_20.setObjectName("lcdNumber_20")
self.gridLayout.addWidget(self.lcdNumber_20, 19, 1, 1, 1)
self.lcdNumber_18 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_18.setObjectName("lcdNumber_18")
self.gridLayout.addWidget(self.lcdNumber_18, 17, 1, 1, 1)
self.lcdNumber_19 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_19.setObjectName("lcdNumber_19")
self.gridLayout.addWidget(self.lcdNumber_19, 18, 1, 1, 1)
self.label_25 = QtWidgets.QLabel(self.scrollContents)
self.label_25.setObjectName("label_25")
self.gridLayout.addWidget(self.label_25, 18, 0, 1, 1)
self.lcdNumber_21 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_21.setObjectName("lcdNumber_21")
self.gridLayout.addWidget(self.lcdNumber_21, 20, 1, 1, 1)
self.label_24 = QtWidgets.QLabel(self.scrollContents)
self.label_24.setObjectName("label_24")
self.gridLayout.addWidget(self.label_24, 17, 0, 1, 1)
self.lcdNumber_22 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_22.setObjectName("lcdNumber_22")
self.gridLayout.addWidget(self.lcdNumber_22, 21, 1, 1, 1)
self.label_26 = QtWidgets.QLabel(self.scrollContents)
self.label_26.setObjectName("label_26")
self.gridLayout.addWidget(self.label_26, 19, 0, 1, 1)
self.lcdNumber_23 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_23.setObjectName("lcdNumber_23")
self.gridLayout.addWidget(self.lcdNumber_23, 22, 1, 1, 1)
self.lcdNumber_24 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_24.setObjectName("lcdNumber_24")
self.gridLayout.addWidget(self.lcdNumber_24, 23, 1, 1, 1)
self.label_27 = QtWidgets.QLabel(self.scrollContents)
self.label_27.setObjectName("label_27")
self.gridLayout.addWidget(self.label_27, 20, 0, 1, 1)
self.label_28 = QtWidgets.QLabel(self.scrollContents)
self.label_28.setObjectName("label_28")
self.gridLayout.addWidget(self.label_28, 21, 0, 1, 1)
self.lcdNumber_25 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_25.setObjectName("lcdNumber_25")
self.gridLayout.addWidget(self.lcdNumber_25, 24, 1, 1, 1)
self.label_30 = QtWidgets.QLabel(self.scrollContents)
self.label_30.setObjectName("label_30")
self.gridLayout.addWidget(self.label_30, 23, 0, 1, 1)
self.label_31 = QtWidgets.QLabel(self.scrollContents)
self.label_31.setObjectName("label_31")
self.gridLayout.addWidget(self.label_31, 24, 0, 1, 1)
self.label_32 = QtWidgets.QLabel(self.scrollContents)
self.label_32.setObjectName("label_32")
self.gridLayout.addWidget(self.label_32, 25, 0, 1, 1)
self.lcdNumber_28 = QtWidgets.QLCDNumber(self.scrollContents)
```

```
self.lcdNumber_28.setObjectName("lcdNumber_28")
self.gridLayout.addWidget(self.lcdNumber_28, 27, 1, 1, 1)
self.lcdNumber_29 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_29.setObjectName("lcdNumber_29")
self.gridLayout.addWidget(self.lcdNumber_29, 28, 1, 1, 1)
self.lcdNumber_27 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_27.setObjectName("lcdNumber_27")
self.gridLayout.addWidget(self.lcdNumber_27, 26, 1, 1, 1)
self.label_33 = QtWidgets.QLabel(self.scrollContents)
self.label_33.setObjectName("label_33")
self.gridLayout.addWidget(self.label_33, 26, 0, 1, 1)
self.lcdNumber_34 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_34.setObjectName("lcdNumber_34")
self.gridLayout.addWidget(self.lcdNumber_34, 33, 1, 1, 1)
self.lcdNumber_31 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_31.setObjectName("lcdNumber_31")
self.gridLayout.addWidget(self.lcdNumber_31, 30, 1, 1, 1)
self.label_35 = QtWidgets.QLabel(self.scrollContents)
self.label_35.setObjectName("label_35")
self.gridLayout.addWidget(self.label_35, 28, 0, 1, 1)
self.label_34 = QtWidgets.QLabel(self.scrollContents)
self.label_34.setObjectName("label_34")
self.gridLayout.addWidget(self.label_34, 27, 0, 1, 1)
self.label_37 = QtWidgets.QLabel(self.scrollContents)
self.label_37.setObjectName("label_37")
self.gridLayout.addWidget(self.label_37, 30, 0, 1, 1)
self.lcdNumber_32 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_32.setObjectName("lcdNumber_32")
self.gridLayout.addWidget(self.lcdNumber_32, 31, 1, 1, 1)
self.lcdNumber_33 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_33.setObjectName("lcdNumber_33")
self.gridLayout.addWidget(self.lcdNumber_33, 32, 1, 1, 1)
self.label_36 = QtWidgets.QLabel(self.scrollContents)
self.label_36.setObjectName("label_36")
self.gridLayout.addWidget(self.label_36, 29, 0, 1, 1)
self.lcdNumber_30 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_30.setObjectName("lcdNumber_30")
self.gridLayout.addWidget(self.lcdNumber_30, 29, 1, 1, 1)
self.label_46 = QtWidgets.QLabel(self.scrollContents)
self.label_46.setObjectName("label_46")
self.gridLayout.addWidget(self.label_46, 35, 0, 1, 1)
self.lcdNumber_35 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_35.setObjectName("lcdNumber_35")
self.gridLayout.addWidget(self.lcdNumber_35, 34, 1, 1, 1)
self.lcdNumber_36 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_36.setObjectName("lcdNumber_36")
self.gridLayout.addWidget(self.lcdNumber_36, 35, 1, 1, 1)
self.label_45 = QtWidgets.QLabel(self.scrollContents)
self.label_45.setObjectName("label_45")
self.gridLayout.addWidget(self.label_45, 34, 0, 1, 1)
self.label_29 = QtWidgets.QLabel(self.scrollContents)
```

```
self.label_29.setObjectName("label_29")
self.gridLayout.addWidget(self.label_29, 22, 0, 1, 1)
self.label_48 = QtWidgets.QLabel(self.scrollContents)
self.label_48.setObjectName("label_48")
self.gridLayout.addWidget(self.label_48, 37, 0, 1, 1)
self.lcdNumber_37 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_37.setObjectName("lcdNumber_37")
self.gridLayout.addWidget(self.lcdNumber_37, 36, 1, 1, 1)
self.label_47 = QtWidgets.QLabel(self.scrollContents)
self.label_47.setObjectName("label_47")
self.gridLayout.addWidget(self.label_47, 36, 0, 1, 1)
self.lcdNumber_39 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_39.setObjectName("lcdNumber_39")
self.gridLayout.addWidget(self.lcdNumber_39, 38, 1, 1, 1)
self.label_49 = QtWidgets.QLabel(self.scrollContents)
self.label_49.setObjectName("label_49")
self.gridLayout.addWidget(self.label_49, 38, 0, 1, 1)
self.lcdNumber_38 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_38.setObjectName("lcdNumber_38")
self.gridLayout.addWidget(self.lcdNumber_38, 37, 1, 1, 1)
self.label_50 = QtWidgets.QLabel(self.scrollContents)
self.label_50.setObjectName("label_50")
self.gridLayout.addWidget(self.label_50, 39, 0, 1, 1)
self.lcdNumber_40 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_40.setObjectName("lcdNumber_40")
self.gridLayout.addWidget(self.lcdNumber_40, 39, 1, 1, 1)
self.lcdNumber_41 = QtWidgets.QLCDNumber(self.scrollContents)
self.lcdNumber_41.setObjectName("lcdNumber_41")
self.gridLayout.addWidget(self.lcdNumber_41, 40, 1, 1, 1)
self.label_51 = QtWidgets.QLabel(self.scrollContents)
self.label_51.setObjectName("label_51")
self.gridLayout.addWidget(self.label_51, 40, 0, 1, 1)
self.read_5 = QtWidgets.QPushButton(self.scrollContents)
self.read_5.setObjectName("read_5")
self.gridLayout.addWidget(self.read_5, 4, 2, 1, 1)
self.read_7 = QtWidgets.QPushButton(self.scrollContents)
self.read_7.setObjectName("read_7")
self.gridLayout.addWidget(self.read_7, 6, 2, 1, 1)
self.read_8 = QtWidgets.QPushButton(self.scrollContents)
self.read_8.setObjectName("read_8")
self.gridLayout.addWidget(self.read_8, 7, 2, 1, 1)
self.read_6 = QtWidgets.QPushButton(self.scrollContents)
self.read_6.setObjectName("read_6")
self.gridLayout.addWidget(self.read_6, 5, 2, 1, 1)
self.read_10 = QtWidgets.QPushButton(self.scrollContents)
self.read_10.setObjectName("read_10")
self.gridLayout.addWidget(self.read_10, 9, 2, 1, 1)
self.read_11 = QtWidgets.QPushButton(self.scrollContents)
self.read_11.setObjectName("read_11")
self.gridLayout.addWidget(self.read_11, 10, 2, 1, 1)
self.read_9 = QtWidgets.QPushButton(self.scrollContents)
```

```
self.read_9.setObjectName("read_9")
self.gridLayout.addWidget(self.read_9, 8, 2, 1, 1)
self.read_12 = QtWidgets.QPushButton(self.scrollContents)
self.read_12.setObjectName("read_12")
self.gridLayout.addWidget(self.read_12, 11, 2, 1, 1)
self.read_18 = QtWidgets.QPushButton(self.scrollContents)
self.read_18.setObjectName("read_18")
self.gridLayout.addWidget(self.read_18, 17, 2, 1, 1)
self.read_15 = QtWidgets.QPushButton(self.scrollContents)
self.read_15.setObjectName("read_15")
self.gridLayout.addWidget(self.read_15, 14, 2, 1, 1)
self.read_13 = QtWidgets.QPushButton(self.scrollContents)
self.read_13.setObjectName("read_13")
self.gridLayout.addWidget(self.read_13, 12, 2, 1, 1)
self.read_14 = QtWidgets.QPushButton(self.scrollContents)
self.read_14.setObjectName("read_14")
self.gridLayout.addWidget(self.read_14, 13, 2, 1, 1)
self.read_16 = QtWidgets.QPushButton(self.scrollContents)
self.read_16.setObjectName("read_16")
self.gridLayout.addWidget(self.read_16, 15, 2, 1, 1)
self.read_17 = QtWidgets.QPushButton(self.scrollContents)
self.read_17.setObjectName("read_17")
self.gridLayout.addWidget(self.read_17, 16, 2, 1, 1)
self.read_20 = QtWidgets.QPushButton(self.scrollContents)
self.read_20.setObjectName("read_20")
self.gridLayout.addWidget(self.read_20, 19, 2, 1, 1)
self.read_19 = QtWidgets.QPushButton(self.scrollContents)
self.read_19.setObjectName("read_19")
self.gridLayout.addWidget(self.read_19, 18, 2, 1, 1)
self.read_21 = QtWidgets.QPushButton(self.scrollContents)
self.read_21.setObjectName("read_21")
self.gridLayout.addWidget(self.read_21, 20, 2, 1, 1)
self.read_24 = QtWidgets.QPushButton(self.scrollContents)
self.read_24.setObjectName("read_24")
self.gridLayout.addWidget(self.read_24, 23, 2, 1, 1)
self.read_22 = QtWidgets.QPushButton(self.scrollContents)
self.read_22.setObjectName("read_22")
self.gridLayout.addWidget(self.read_22, 21, 2, 1, 1)
self.read_23 = QtWidgets.QPushButton(self.scrollContents)
self.read_23.setObjectName("read_23")
self.gridLayout.addWidget(self.read_23, 22, 2, 1, 1)
self.read_25 = QtWidgets.QPushButton(self.scrollContents)
self.read_25.setObjectName("read_25")
self.gridLayout.addWidget(self.read_25, 24, 2, 1, 1)
self.read_29 = QtWidgets.QPushButton(self.scrollContents)
self.read_29.setObjectName("read_29")
self.gridLayout.addWidget(self.read_29, 28, 2, 1, 1)
self.read_28 = QtWidgets.QPushButton(self.scrollContents)
self.read_28.setObjectName("read_28")
self.gridLayout.addWidget(self.read_28, 27, 2, 1, 1)
self.read_27 = QtWidgets.QPushButton(self.scrollContents)
```

```
self.read_27.setObjectName("read_27")
self.gridLayout.addWidget(self.read_27, 26, 2, 1, 1)
self.read_26 = QtWidgets.QPushButton(self.scrollContents)
self.read_26.setObjectName("read_26")
self.gridLayout.addWidget(self.read_26, 25, 2, 1, 1)
self.read_30 = QtWidgets.QPushButton(self.scrollContents)
self.read_30.setObjectName("read_30")
self.gridLayout.addWidget(self.read_30, 29, 2, 1, 1)
self.read_31 = QtWidgets.QPushButton(self.scrollContents)
self.read_31.setObjectName("read_31")
self.gridLayout.addWidget(self.read_31, 30, 2, 1, 1)
self.read_32 = QtWidgets.QPushButton(self.scrollContents)
self.read_32.setObjectName("read_32")
self.gridLayout.addWidget(self.read_32, 31, 2, 1, 1)
self.read_33 = QtWidgets.QPushButton(self.scrollContents)
self.read_33.setObjectName("read_33")
self.gridLayout.addWidget(self.read_33, 32, 2, 1, 1)
self.read_34 = QtWidgets.QPushButton(self.scrollContents)
self.read_34.setObjectName("read_34")
self.gridLayout.addWidget(self.read_34, 33, 2, 1, 1)
self.read_35 = QtWidgets.QPushButton(self.scrollContents)
self.read_35.setObjectName("read_35")
self.gridLayout.addWidget(self.read_35, 34, 2, 1, 1)
self.read_36 = QtWidgets.QPushButton(self.scrollContents)
self.read_36.setObjectName("read_36")
self.gridLayout.addWidget(self.read_36, 35, 2, 1, 1)
self.read_37 = QtWidgets.QPushButton(self.scrollContents)
self.read_37.setObjectName("read_37")
self.gridLayout.addWidget(self.read_37, 36, 2, 1, 1)
self.read_38 = QtWidgets.QPushButton(self.scrollContents)
self.read_38.setObjectName("read_38")
self.gridLayout.addWidget(self.read_38, 37, 2, 1, 1)
self.read_39 = QtWidgets.QPushButton(self.scrollContents)
self.read_39.setObjectName("read_39")
self.gridLayout.addWidget(self.read_39, 38, 2, 1, 1)
self.read_40 = QtWidgets.QPushButton(self.scrollContents)
self.read_40.setObjectName("read_40")
self.gridLayout.addWidget(self.read_40, 39, 2, 1, 1)
self.read_41 = QtWidgets.QPushButton(self.scrollContents)
self.read_41.setObjectName("read_41")
self.gridLayout.addWidget(self.read_41, 40, 2, 1, 1)
self.read_42 = QtWidgets.QPushButton(self.scrollContents)
self.read_42.setObjectName("read_42")
self.gridLayout.addWidget(self.read_42, 41, 2, 1, 1)
self.read_43 = QtWidgets.QPushButton(self.scrollContents)
self.read_43.setObjectName("read_43")
self.gridLayout.addWidget(self.read_43, 42, 2, 1, 1)
self.read_44 = QtWidgets.QPushButton(self.scrollContents)
self.read_44.setObjectName("read_44")
self.gridLayout.addWidget(self.read_44, 43, 2, 1, 1)
self.read_45 = QtWidgets.QPushButton(self.scrollContents)
```

```
self.read_45.setObjectName("read_45")
self.gridLayout.addWidget(self.read_45, 44, 2, 1, 1)
self.read_46 = QtWidgets.QPushButton(self.scrollContents)
self.read_46.setObjectName("read_46")
self.gridLayout.addWidget(self.read_46, 45, 2, 1, 1)
self.read_47 = QtWidgets.QPushButton(self.scrollContents)
self.read_47.setObjectName("read_47")
self.gridLayout.addWidget(self.read_47, 46, 2, 1, 1)
self.read_48 = QtWidgets.QPushButton(self.scrollContents)
self.read_48.setObjectName("read_48")
self.gridLayout.addWidget(self.read_48, 47, 2, 1, 1)
self.scrollArea_readHV.setWidget(self.scrollContents)
self.label_89 = QtWidgets.QLabel(self.groupBox_readHV)
self.label_89.setGeometry(QtCore.QRect(30, 30, 161, 21))
self.label_89.setObjectName("label_89")
self.read_all_hv = QtWidgets.QPushButton(self.groupBox_readHV)
self.read_all_hv.setGeometry(QtCore.QRect(205, 27, 93, 28))
self.read_all_hv.setObjectName("read_all_hv")
self.gridLayout_5.addWidget(self.groupBox_readHV, 0, 1, 1, 1)
self.verticalLayout = QtWidgets.QVBoxLayout()
self.verticalLayout.setSizeConstraint(QtWidgets.QLayout.SetDefaultConstraint)
self.verticalLayout.setSpacing(7)
self.verticalLayout.setObjectName("verticalLayout")
self.groupBox_LV = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox_LV.setMinimumSize(QtCore.QSize(400, 300))
self.groupBox_LV.setMaximumSize(QtCore.QSize(400, 300))
font = QtGui.QFont()
font.setFamily("MS Shell Dlg 2")
font.setPointSize(9)
self.groupBox_LV.setFont(font)
self.groupBox_LV.setObjectName("groupBox_LV")
self.button_enable_12 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_enable_12.setGeometry(QtCore.QRect(70, 170, 91, 41))
self.button_enable_12.setObjectName("button_enable_12")
self.status_N12 = QtWidgets.QLabel(self.groupBox_LV)
self.status_N12.setGeometry(QtCore.QRect(290, 240, 71, 21))
font = QtGui.QFont()
font.setBold(True)
font.setWeight(75)
self.status_N12.setFont(font)
self.status_N12.setObjectName("status_N12")
self.status_5 = QtWidgets.QLabel(self.groupBox_LV)
self.status_5.setGeometry(QtCore.QRect(290, 120, 71, 21))
font = QtGui.QFont()
font.setBold(True)
font.setWeight(75)
self.status_5.setFont(font)
self.status_5.setObjectName("status_5")
self.status_P12 = QtWidgets.QLabel(self.groupBox_LV)
self.status_P12.setGeometry(QtCore.QRect(290, 180, 71, 21))
font = QtGui.QFont()
```

```
font.setBold(True)
font.setWeight(75)
self.status_P12.setFont(font)
self.status_P12.setObjectName("status_P12")
self.button_enable_5 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_enable_5.setGeometry(QtCore.QRect(70, 110, 91, 41))
self.button_enable_5.setObjectName("button_enable_5")
self.button_enable_N12 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_enable_N12.setGeometry(QtCore.QRect(70, 230, 91, 41))
self.button_enable_N12.setObjectName("button_enable_N12")
self.button_disable_5 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_disable_5.setGeometry(QtCore.QRect(180, 110, 91, 41))
self.button_disable_5.setObjectName("button_disable_5")
self.button_disable_12 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_disable_12.setGeometry(QtCore.QRect(180, 170, 91, 41))
self.button_disable_12.setObjectName("button_disable_12")
self.button_disable_N12 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_disable_N12.setGeometry(QtCore.QRect(180, 230, 91, 41))
self.button_disable_N12.setObjectName("button_disable_N12")
self.button_disable_3V3 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_disable_3V3.setGeometry(QtCore.QRect(180, 50, 91, 41))
self.button_disable_3V3.setObjectName("button_disable_3V3")
self.status_3V3 = QtWidgets.QLabel(self.groupBox_LV)
self.status_3V3.setGeometry(QtCore.QRect(290, 60, 71, 21))
font = QtGui.QFont()
font.setBold(True)
font.setWeight(75)
self.status_3V3.setFont(font)
self.status_3V3.setObjectName("status_3V3")
self.button_enable_3V3 = QtWidgets.QPushButton(self.groupBox_LV)
self.button_enable_3V3.setGeometry(QtCore.QRect(70, 50, 91, 41))
self.button_enable_3V3.setObjectName("button_enable_3V3")
self.label_3V3 = QtWidgets.QLabel(self.groupBox_LV)
self.label_3V3.setGeometry(QtCore.QRect(20, 60, 41, 21))
self.label_3V3.setObjectName("label_3V3")
self.label_5V = QtWidgets.QLabel(self.groupBox_LV)
self.label_5V.setGeometry(QtCore.QRect(20, 120, 41, 21))
self.label_5V.setObjectName("label_5V")
self.label_VP12 = QtWidgets.QLabel(self.groupBox_LV)
self.label_VP12.setGeometry(QtCore.QRect(20, 180, 41, 21))
self.label_VP12.setObjectName("label_VP12")
self.label_VN12 = QtWidgets.QLabel(self.groupBox_LV)
self.label_VN12.setGeometry(QtCore.QRect(20, 240, 41, 21))
self.label_VN12.setObjectName("label_VN12")
self.verticalLayout.addWidget(self.groupBox_LV)
self.groupBox = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox.setMinimumSize(QtCore.QSize(400, 125))
self.groupBox.setMaximumSize(QtCore.QSize(400, 100))
font = QtGui.QFont()
font.setPointSize(9)
self.groupBox.setFont(font)
```

```
self.groupBox.setObjectName("groupBox")
self.textEdit_setHV = QtWidgets.QTextEdit(self.groupBox)
self.textEdit_setHV.setGeometry(QtCore.QRect(40, 30, 91, 29))
self.textEdit_setHV.setObjectName("textEdit_setHV")
self.label_unit = QtWidgets.QLabel(self.groupBox)
self.label_unit.setGeometry(QtCore.QRect(150, 40, 31, 16))
font = QtGui.QFont()
font.setPointSize(10)
self.label_unit.setFont(font)
self.label_unit.setObjectName("label_unit")
self.button_setHV = QtWidgets.QPushButton(self.groupBox)
self.button_setHV.setGeometry(QtCore.QRect(240, 30, 91, 31))
font = QtGui.QFont()
font.setPointSize(9)
self.button_setHV.setFont(font)
self.button_setHV.setObjectName("button_setHV")
self.textEdit_setCable = QtWidgets.QTextEdit(self.groupBox)
self.textEdit_setCable.setGeometry(QtCore.QRect(40, 80, 91, 29))
self.textEdit_setCable.setObjectName("textEdit_setCable")
self.label_5 = QtWidgets.QLabel(self.groupBox)
self.label_5.setGeometry(QtCore.QRect(150, 90, 55, 16))
self.label_5.setObjectName("label_5")
self.set_cable = QtWidgets.QPushButton(self.groupBox)
self.set_cable.setGeometry(QtCore.QRect(240, 80, 93, 28))
self.set_cable.setObjectName("set_cable")
self.verticalLayout.addWidget(self.groupBox)
self.groupBox_2 = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox_2.setMinimumSize(QtCore.QSize(400, 100))
self.groupBox_2.setMaximumSize(QtCore.QSize(400, 16777215))
font = QtGui.QFont()
font.setPointSize(9)
self.groupBox_2.setFont(font)
self.groupBox_2.setObjectName("groupBox_2")
self.read_HV = QtWidgets.QPushButton(self.groupBox_2)
self.read_HV.setGeometry(QtCore.QRect(240, 50, 90, 35))
self.read_HV.setObjectName("read_HV")
self.HV_LCD = QtWidgets.QLCDNumber(self.groupBox_2)
self.HV_LCD.setGeometry(QtCore.QRect(30, 50, 101, 35))
self.HV_LCD.setObjectName("HV_LCD")
self.label_4 = QtWidgets.QLabel(self.groupBox_2)
self.label_4.setGeometry(QtCore.QRect(140, 60, 81, 21))
self.label_4.setObjectName("label_4")
self.verticalLayout.addWidget(self.groupBox_2)
self.groupBox_3 = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox_3.setMinimumSize(QtCore.QSize(0, 100))
self.groupBox_3.setMaximumSize(QtCore.QSize(400, 16777215))
font = QtGui.QFont()
font.setPointSize(9)
self.groupBox_3.setFont(font)
self.groupBox_3.setObjectName("groupBox_3")
self.cern_LCD = QtWidgets.QLCDNumber(self.groupBox_3)
```

```
self.cern_LCD.setGeometry(QtCore.QRect(30, 35, 101, 35))
self.cern_LCD.setObjectName("cern_LCD")
self.label_6 = QtWidgets.QLabel(self.groupBox_3)
self.label_6.setGeometry(QtCore.QRect(140, 40, 81, 21))
self.label_6.setObjectName("label_6")
self.read_continuity = QtWidgets.QPushButton(self.groupBox_3)
self.read_continuity.setGeometry(QtCore.QRect(240, 40, 90, 35))
self.read_continuity.setObjectName("read_continuity")
self.verticalLayout.addWidget(self.groupBox_3)
self.groupBox_readSHIELD = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox_readSHIELD.setMinimumSize(QtCore.QSize(400, 200))
self.groupBox_readSHIELD.setMaximumSize(QtCore.QSize(400, 200))
font = QtGui.QFont()
font.setPointSize(9)
self.groupBox_readSHIELD.setFont(font)
self.groupBox_readSHIELD.setObjectName("groupBox_readSHIELD")
self.gridLayout_6 = QtWidgets.QGridLayout(self.groupBox_readSHIELD)
self.gridLayout_6.setObjectName("gridLayout_6")
self.read_shield2 = QtWidgets.QPushButton(self.groupBox_readSHIELD)
self.read_shield2.setObjectName("read_shield2")
self.gridLayout_6.addWidget(self.read_shield2, 1, 3, 1, 1)
self.label_2 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_2.setObjectName("label_2")
self.gridLayout_6.addWidget(self.label_2, 1, 0, 1, 1)
self.shield4_LCD = QtWidgets.QLCDNumber(self.groupBox_readSHIELD)
self.shield4_LCD.setObjectName("shield4_LCD")
self.gridLayout_6.addWidget(self.shield4_LCD, 4, 1, 1, 1)
self.label = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label.setObjectName("label")
self.gridLayout_6.addWidget(self.label, 0, 0, 1, 1)
self.label_3 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_3.setObjectName("label_3")
self.gridLayout_6.addWidget(self.label_3, 2, 0, 1, 1)
self.read_shield1 = QtWidgets.QPushButton(self.groupBox_readSHIELD)
self.read_shield1.setObjectName("read_shield1")
self.gridLayout_6.addWidget(self.read_shield1, 0, 3, 1, 1)
self.shield3_LCD = QtWidgets.QLCDNumber(self.groupBox_readSHIELD)
self.shield3_LCD.setObjectName("shield3_LCD")
self.gridLayout_6.addWidget(self.shield3_LCD, 2, 1, 1, 1)
self.shield2_LCD = QtWidgets.QLCDNumber(self.groupBox_readSHIELD)
self.shield2_LCD.setObjectName("shield2_LCD")
self.gridLayout_6.addWidget(self.shield2_LCD, 1, 1, 1, 1)
self.label_10 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_10.setObjectName("label_10")
self.gridLayout_6.addWidget(self.label_10, 4, 0, 1, 1)
self.read_shield3 = QtWidgets.QPushButton(self.groupBox_readSHIELD)
self.read_shield3.setObjectName("read_shield3")
self.gridLayout_6.addWidget(self.read_shield3, 2, 3, 1, 1)
self.shield1_LCD = QtWidgets.QLCDNumber(self.groupBox_readSHIELD)
self.shield1_LCD.setObjectName("shield1_LCD")
self.gridLayout_6.addWidget(self.shield1_LCD, 0, 1, 1, 1)
```

```
self.read_shield4 = QtWidgets.QPushButton(self.groupBox_readSHIELD)
self.read_shield4.setObjectName("read_shield4")
self.gridLayout_6.addWidget(self.read_shield4, 4, 3, 1, 1)
self.label_11 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_11.setObjectName("label_11")
self.gridLayout_6.addWidget(self.label_11, 0, 2, 1, 1)
self.label_14 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_14.setObjectName("label_14")
self.gridLayout_6.addWidget(self.label_14, 4, 2, 1, 1)
self.label_13 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_13.setObjectName("label_13")
self.gridLayout_6.addWidget(self.label_13, 2, 2, 1, 1)
self.label_12 = QtWidgets.QLabel(self.groupBox_readSHIELD)
self.label_12.setObjectName("label_12")
self.gridLayout_6.addWidget(self.label_12, 1, 2, 1, 1)
self.verticalLayout.addWidget(self.groupBox_readSHIELD)
self.gridLayout_5.addLayout(self.verticalLayout, 0, 0, 1, 1)
self.groupBox_readGND = QtWidgets.QGroupBox(self.scrollAreaWidgetContents)
self.groupBox_readGND.setMinimumSize(QtCore.QSize(350, 0))
self.groupBox_readGND.setMaximumSize(QtCore.QSize(350, 16777215))
font = QtGui.QFont()
font.setPointSize(9)
self.groupBox_readGND.setFont(font)
self.groupBox_readGND.setObjectName("groupBox_readGND")
self.label_90 = QtWidgets.QLabel(self.groupBox_readGND)
self.label_90.setGeometry(QtCore.QRect(30, 30, 161, 21))
self.label_90.setObjectName("label_90")
self.read_all_gnd = QtWidgets.QPushButton(self.groupBox_readGND)
self.read_all_gnd.setGeometry(QtCore.QRect(205, 27, 93, 28))
self.read_all_gnd.setObjectName("read_all_gnd")
self.scrollArea_readHV_2 = QtWidgets.QScrollArea(self.groupBox_readGND)
self.scrollArea_readHV_2.setEnabled(True)
self.scrollArea_readHV_2.setGeometry(QtCore.QRect(30, 70, 300, 800))
self.scrollArea_readHV_2.setMinimumSize(QtCore.QSize(300, 0))
self.scrollArea_readHV_2.setMaximumSize(QtCore.QSize(300, 800))
self.scrollArea_readHV_2.setWidgetResizable(True)
self.scrollArea_readHV_2.setObjectName("scrollArea_readHV_2")
self.scrollContents_6 = QtWidgets.QWidget()
self.scrollContents_6.setGeometry(QtCore.QRect(0, 0, 277, 1695))
self.scrollContents_6.setObjectName("scrollContents_6")
self.gridLayout_8 = QtWidgets.QGridLayout(self.scrollContents_6)
self.gridLayout_8.setObjectName("gridLayout_8")
self.label_81 = QtWidgets.QLabel(self.scrollContents_6)
self.label_81.setObjectName("label_81")
self.gridLayout_8.addWidget(self.label_81, 33, 0, 1, 1)
self.lcdNumber_64 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_64.setObjectName("lcdNumber_64")
self.gridLayout_8.addWidget(self.lcdNumber_64, 15, 1, 1, 1)
self.label_82 = QtWidgets.QLabel(self.scrollContents_6)
self.label_82.setObjectName("label_82")
self.gridLayout_8.addWidget(self.label_82, 12, 0, 1, 1)
```

```
self.lcdNumber_61 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_61.setObjectName("lcdNumber_61")
self.gridLayout_8.addWidget(self.lcdNumber_61, 12, 1, 1, 1)
self.read_49 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_49.setObjectName("read_49")
self.gridLayout_8.addWidget(self.read_49, 0, 2, 1, 1)
self.label_HV2_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV2_4.setObjectName("label_HV2_4")
self.gridLayout_8.addWidget(self.label_HV2_4, 1, 0, 1, 1)
self.lcdNumber_62 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_62.setObjectName("lcdNumber_62")
self.gridLayout_8.addWidget(self.lcdNumber_62, 13, 1, 1, 1)
self.lcdNumber_49 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_49.setObjectName("lcdNumber_49")
self.gridLayout_8.addWidget(self.lcdNumber_49, 0, 1, 1, 1)
self.lcdNumber_50 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_50.setObjectName("lcdNumber_50")
self.gridLayout_8.addWidget(self.lcdNumber_50, 1, 1, 1, 1)
self.label_HV1_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV1_4.setObjectName("label_HV1_4")
self.gridLayout_8.addWidget(self.label_HV1_4, 0, 0, 1, 1)
self.read_50 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_50.setObjectName("read_50")
self.gridLayout_8.addWidget(self.read_50, 1, 2, 1, 1)
self.label_HV3_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV3_4.setObjectName("label_HV3_4")
self.gridLayout_8.addWidget(self.label_HV3_4, 2, 0, 1, 1)
self.lcdNumber_51 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_51.setObjectName("lcdNumber_51")
self.gridLayout_8.addWidget(self.lcdNumber_51, 2, 1, 1, 1)
self.read_51 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_51.setObjectName("read_51")
self.gridLayout_8.addWidget(self.read_51, 2, 2, 1, 1)
self.label_83 = QtWidgets.QLabel(self.scrollContents_6)
self.label_83.setObjectName("label_83")
self.gridLayout_8.addWidget(self.label_83, 9, 0, 1, 1)
self.lcdNumber_60 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_60.setObjectName("lcdNumber_60")
self.gridLayout_8.addWidget(self.lcdNumber_60, 11, 1, 1, 1)
self.label_84 = QtWidgets.QLabel(self.scrollContents_6)
self.label_84.setObjectName("label_84")
self.gridLayout_8.addWidget(self.label_84, 32, 0, 1, 1)
self.lcdNumber_56 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_56.setObjectName("lcdNumber_56")
self.gridLayout_8.addWidget(self.lcdNumber_56, 7, 1, 1, 1)
self.label_85 = QtWidgets.QLabel(self.scrollContents_6)
self.label_85.setObjectName("label_85")
self.gridLayout_8.addWidget(self.label_85, 31, 0, 1, 1)
self.lcdNumber_63 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_63.setObjectName("lcdNumber_63")
self.gridLayout_8.addWidget(self.lcdNumber_63, 14, 1, 1, 1)
```

```
self.label_HV8_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV8_4.setObjectName("label_HV8_4")
self.gridLayout_8.addWidget(self.label_HV8_4, 7, 0, 1, 1)
self.lcdNumber_57 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_57.setObjectName("lcdNumber_57")
self.gridLayout_8.addWidget(self.lcdNumber_57, 8, 1, 1, 1)
self.label_86 = QtWidgets.QLabel(self.scrollContents_6)
self.label_86.setObjectName("label_86")
self.gridLayout_8.addWidget(self.label_86, 11, 0, 1, 1)
self.lcdNumber_58 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_58.setObjectName("lcdNumber_58")
self.gridLayout_8.addWidget(self.lcdNumber_58, 9, 1, 1, 1)
self.label_87 = QtWidgets.QLabel(self.scrollContents_6)
self.label_87.setObjectName("label_87")
self.gridLayout_8.addWidget(self.label_87, 8, 0, 1, 1)
self.label_88 = QtWidgets.QLabel(self.scrollContents_6)
self.label_88.setObjectName("label_88")
self.gridLayout_8.addWidget(self.label_88, 15, 0, 1, 1)
self.lcdNumber_55 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_55.setObjectName("lcdNumber_55")
self.gridLayout_8.addWidget(self.lcdNumber_55, 6, 1, 1, 1)
self.label_HV5_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV5_4.setObjectName("label_HV5_4")
self.gridLayout_8.addWidget(self.label_HV5_4, 4, 0, 1, 1)
self.lcdNumber_59 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_59.setObjectName("lcdNumber_59")
self.gridLayout_8.addWidget(self.lcdNumber_59, 10, 1, 1, 1)
self.label_149 = QtWidgets.QLabel(self.scrollContents_6)
self.label_149.setObjectName("label_149")
self.gridLayout_8.addWidget(self.label_149, 13, 0, 1, 1)
self.read_52 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_52.setObjectName("read_52")
self.gridLayout_8.addWidget(self.read_52, 3, 2, 1, 1)
self.label_HV6_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV6_4.setObjectName("label_HV6_4")
self.gridLayout_8.addWidget(self.label_HV6_4, 5, 0, 1, 1)
self.label_HV7_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV7_4.setObjectName("label_HV7_4")
self.gridLayout_8.addWidget(self.label_HV7_4, 6, 0, 1, 1)
self.label_150 = QtWidgets.QLabel(self.scrollContents_6)
self.label_150.setObjectName("label_150")
self.gridLayout_8.addWidget(self.label_150, 10, 0, 1, 1)
self.lcdNumber_52 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_52.setObjectName("lcdNumber_52")
self.gridLayout_8.addWidget(self.lcdNumber_52, 3, 1, 1, 1)
self.label_151 = QtWidgets.QLabel(self.scrollContents_6)
self.label_151.setObjectName("label_151")
self.gridLayout_8.addWidget(self.label_151, 14, 0, 1, 1)
self.label_HV4_4 = QtWidgets.QLabel(self.scrollContents_6)
self.label_HV4_4.setObjectName("label_HV4_4")
self.gridLayout_8.addWidget(self.label_HV4_4, 3, 0, 1, 1)
```

```
self.lcdNumber_53 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_53.setObjectName("lcdNumber_53")
self.gridLayout_8.addWidget(self.lcdNumber_53, 4, 1, 1, 1)
self.lcdNumber_54 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_54.setObjectName("lcdNumber_54")
self.gridLayout_8.addWidget(self.lcdNumber_54, 5, 1, 1, 1)
self.lcdNumber_91 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_91.setObjectName("lcdNumber_91")
self.gridLayout_8.addWidget(self.lcdNumber_91, 42, 1, 1, 1)
self.lcdNumber_92 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_92.setObjectName("lcdNumber_92")
self.gridLayout_8.addWidget(self.lcdNumber_92, 43, 1, 1, 1)
self.label_152 = QtWidgets.QLabel(self.scrollContents_6)
self.label_152.setObjectName("label_152")
self.gridLayout_8.addWidget(self.label_152, 43, 0, 1, 1)
self.lcdNumber_93 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_93.setObjectName("lcdNumber_93")
self.gridLayout_8.addWidget(self.lcdNumber_93, 44, 1, 1, 1)
self.lcdNumber_95 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_95.setObjectName("lcdNumber_95")
self.gridLayout_8.addWidget(self.lcdNumber_95, 46, 1, 1, 1)
self.lcdNumber_94 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_94.setObjectName("lcdNumber_94")
self.gridLayout_8.addWidget(self.lcdNumber_94, 45, 1, 1, 1)
self.lcdNumber_96 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_96.setObjectName("lcdNumber_96")
self.gridLayout_8.addWidget(self.lcdNumber_96, 47, 1, 1, 1)
self.label_153 = QtWidgets.QLabel(self.scrollContents_6)
self.label_153.setObjectName("label_153")
self.gridLayout_8.addWidget(self.label_153, 45, 0, 1, 1)
self.label_154 = QtWidgets.QLabel(self.scrollContents_6)
self.label_154.setObjectName("label_154")
self.gridLayout_8.addWidget(self.label_154, 44, 0, 1, 1)
self.label_155 = QtWidgets.QLabel(self.scrollContents_6)
self.label_155.setObjectName("label_155")
self.gridLayout_8.addWidget(self.label_155, 47, 0, 1, 1)
self.label_156 = QtWidgets.QLabel(self.scrollContents_6)
self.label_156.setObjectName("label_156")
self.gridLayout_8.addWidget(self.label_156, 46, 0, 1, 1)
self.lcdNumber_90 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_90.setObjectName("lcdNumber_90")
self.gridLayout_8.addWidget(self.lcdNumber_90, 41, 1, 1, 1)
self.label_157 = QtWidgets.QLabel(self.scrollContents_6)
self.label_157.setObjectName("label_157")
self.gridLayout_8.addWidget(self.label_157, 41, 0, 1, 1)
self.label_158 = QtWidgets.QLabel(self.scrollContents_6)
self.label_158.setObjectName("label_158")
self.gridLayout_8.addWidget(self.label_158, 42, 0, 1, 1)
self.lcdNumber_65 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_65.setObjectName("lcdNumber_65")
self.gridLayout_8.addWidget(self.lcdNumber_65, 16, 1, 1, 1)
```

```
self.lcdNumber_74 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_74.setObjectName("lcdNumber_74")
self.gridLayout_8.addWidget(self.lcdNumber_74, 25, 1, 1, 1)
self.label_159 = QtWidgets.QLabel(self.scrollContents_6)
self.label_159.setObjectName("label_159")
self.gridLayout_8.addWidget(self.label_159, 16, 0, 1, 1)
self.lcdNumber_68 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_68.setObjectName("lcdNumber_68")
self.gridLayout_8.addWidget(self.lcdNumber_68, 19, 1, 1, 1)
self.lcdNumber_66 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_66.setObjectName("lcdNumber_66")
self.gridLayout_8.addWidget(self.lcdNumber_66, 17, 1, 1, 1)
self.lcdNumber_67 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_67.setObjectName("lcdNumber_67")
self.gridLayout_8.addWidget(self.lcdNumber_67, 18, 1, 1, 1)
self.label_160 = QtWidgets.QLabel(self.scrollContents_6)
self.label_160.setObjectName("label_160")
self.gridLayout_8.addWidget(self.label_160, 18, 0, 1, 1)
self.lcdNumber_69 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_69.setObjectName("lcdNumber_69")
self.gridLayout_8.addWidget(self.lcdNumber_69, 20, 1, 1, 1)
self.label_161 = QtWidgets.QLabel(self.scrollContents_6)
self.label_161.setObjectName("label_161")
self.gridLayout_8.addWidget(self.label_161, 17, 0, 1, 1)
self.lcdNumber_70 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_70.setObjectName("lcdNumber_70")
self.gridLayout_8.addWidget(self.lcdNumber_70, 21, 1, 1, 1)
self.label_162 = QtWidgets.QLabel(self.scrollContents_6)
self.label_162.setObjectName("label_162")
self.gridLayout_8.addWidget(self.label_162, 19, 0, 1, 1)
self.lcdNumber_71 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_71.setObjectName("lcdNumber_71")
self.gridLayout_8.addWidget(self.lcdNumber_71, 22, 1, 1, 1)
self.lcdNumber_72 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_72.setObjectName("lcdNumber_72")
self.gridLayout_8.addWidget(self.lcdNumber_72, 23, 1, 1, 1)
self.label_163 = QtWidgets.QLabel(self.scrollContents_6)
self.label_163.setObjectName("label_163")
self.gridLayout_8.addWidget(self.label_163, 20, 0, 1, 1)
self.label_164 = QtWidgets.QLabel(self.scrollContents_6)
self.label_164.setObjectName("label_164")
self.gridLayout_8.addWidget(self.label_164, 21, 0, 1, 1)
self.lcdNumber_73 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_73.setObjectName("lcdNumber_73")
self.gridLayout_8.addWidget(self.lcdNumber_73, 24, 1, 1, 1)
self.label_165 = QtWidgets.QLabel(self.scrollContents_6)
self.label_165.setObjectName("label_165")
self.gridLayout_8.addWidget(self.label_165, 23, 0, 1, 1)
self.label_166 = QtWidgets.QLabel(self.scrollContents_6)
self.label_166.setObjectName("label_166")
self.gridLayout_8.addWidget(self.label_166, 24, 0, 1, 1)
```

```
self.label_167 = QtWidgets.QLabel(self.scrollContents_6)
self.label_167.setObjectName("label_167")
self.gridLayout_8.addWidget(self.label_167, 25, 0, 1, 1)
self.lcdNumber_76 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_76.setObjectName("lcdNumber_76")
self.gridLayout_8.addWidget(self.lcdNumber_76, 27, 1, 1, 1)
self.lcdNumber_77 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_77.setObjectName("lcdNumber_77")
self.gridLayout_8.addWidget(self.lcdNumber_77, 28, 1, 1, 1)
self.lcdNumber_75 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_75.setObjectName("lcdNumber_75")
self.gridLayout_8.addWidget(self.lcdNumber_75, 26, 1, 1, 1)
self.label_168 = QtWidgets.QLabel(self.scrollContents_6)
self.label_168.setObjectName("label_168")
self.gridLayout_8.addWidget(self.label_168, 26, 0, 1, 1)
self.lcdNumber_82 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_82.setObjectName("lcdNumber_82")
self.gridLayout_8.addWidget(self.lcdNumber_82, 33, 1, 1, 1)
self.lcdNumber_79 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_79.setObjectName("lcdNumber_79")
self.gridLayout_8.addWidget(self.lcdNumber_79, 30, 1, 1, 1)
self.label_169 = QtWidgets.QLabel(self.scrollContents_6)
self.label_169.setObjectName("label_169")
self.gridLayout_8.addWidget(self.label_169, 28, 0, 1, 1)
self.label_170 = QtWidgets.QLabel(self.scrollContents_6)
self.label_170.setObjectName("label_170")
self.gridLayout_8.addWidget(self.label_170, 27, 0, 1, 1)
self.label_171 = QtWidgets.QLabel(self.scrollContents_6)
self.label_171.setObjectName("label_171")
self.gridLayout_8.addWidget(self.label_171, 30, 0, 1, 1)
self.lcdNumber_80 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_80.setObjectName("lcdNumber_80")
self.gridLayout_8.addWidget(self.lcdNumber_80, 31, 1, 1, 1)
self.lcdNumber_81 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_81.setObjectName("lcdNumber_81")
self.gridLayout_8.addWidget(self.lcdNumber_81, 32, 1, 1, 1)
self.label_172 = QtWidgets.QLabel(self.scrollContents_6)
self.label_172.setObjectName("label_172")
self.gridLayout_8.addWidget(self.label_172, 29, 0, 1, 1)
self.lcdNumber_78 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_78.setObjectName("lcdNumber_78")
self.gridLayout_8.addWidget(self.lcdNumber_78, 29, 1, 1, 1)
self.label_173 = QtWidgets.QLabel(self.scrollContents_6)
self.label_173.setObjectName("label_173")
self.gridLayout_8.addWidget(self.label_173, 35, 0, 1, 1)
self.lcdNumber_83 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_83.setObjectName("lcdNumber_83")
self.gridLayout_8.addWidget(self.lcdNumber_83, 34, 1, 1, 1)
self.lcdNumber_84 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_84.setObjectName("lcdNumber_84")
self.gridLayout_8.addWidget(self.lcdNumber_84, 35, 1, 1, 1)
```

```
self.label_174 = QtWidgets.QLabel(self.scrollContents_6)
self.label_174.setObjectName("label_174")
self.gridLayout_8.addWidget(self.label_174, 34, 0, 1, 1)
self.label_175 = QtWidgets.QLabel(self.scrollContents_6)
self.label_175.setObjectName("label_175")
self.gridLayout_8.addWidget(self.label_175, 22, 0, 1, 1)
self.label_176 = QtWidgets.QLabel(self.scrollContents_6)
self.label_176.setObjectName("label_176")
self.gridLayout_8.addWidget(self.label_176, 37, 0, 1, 1)
self.lcdNumber_85 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_85.setObjectName("lcdNumber_85")
self.gridLayout_8.addWidget(self.lcdNumber_85, 36, 1, 1, 1)
self.label_177 = QtWidgets.QLabel(self.scrollContents_6)
self.label_177.setObjectName("label_177")
self.gridLayout_8.addWidget(self.label_177, 36, 0, 1, 1)
self.lcdNumber_87 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_87.setObjectName("lcdNumber_87")
self.gridLayout_8.addWidget(self.lcdNumber_87, 38, 1, 1, 1)
self.label_178 = QtWidgets.QLabel(self.scrollContents_6)
self.label_178.setObjectName("label_178")
self.gridLayout_8.addWidget(self.label_178, 38, 0, 1, 1)
self.lcdNumber_86 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_86.setObjectName("lcdNumber_86")
self.gridLayout_8.addWidget(self.lcdNumber_86, 37, 1, 1, 1)
self.label_179 = QtWidgets.QLabel(self.scrollContents_6)
self.label_179.setObjectName("label_179")
self.gridLayout_8.addWidget(self.label_179, 39, 0, 1, 1)
self.lcdNumber_88 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_88.setObjectName("lcdNumber_88")
self.gridLayout_8.addWidget(self.lcdNumber_88, 39, 1, 1, 1)
self.lcdNumber_89 = QtWidgets.QLCDNumber(self.scrollContents_6)
self.lcdNumber_89.setObjectName("lcdNumber_89")
self.gridLayout_8.addWidget(self.lcdNumber_89, 40, 1, 1, 1)
self.label_180 = QtWidgets.QLabel(self.scrollContents_6)
self.label_180.setObjectName("label_180")
self.gridLayout_8.addWidget(self.label_180, 40, 0, 1, 1)
self.read_53 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_53.setObjectName("read_53")
self.gridLayout_8.addWidget(self.read_53, 4, 2, 1, 1)
self.read_55 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_55.setObjectName("read_55")
self.gridLayout_8.addWidget(self.read_55, 6, 2, 1, 1)
self.read_56 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_56.setObjectName("read_56")
self.gridLayout_8.addWidget(self.read_56, 7, 2, 1, 1)
self.read_54 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_54.setObjectName("read_54")
self.gridLayout_8.addWidget(self.read_54, 5, 2, 1, 1)
self.read_58 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_58.setObjectName("read_58")
self.gridLayout_8.addWidget(self.read_58, 9, 2, 1, 1)
```

```
self.read_59 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_59.setObjectName("read_59")
self.gridLayout_8.addWidget(self.read_59, 10, 2, 1, 1)
self.read_57 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_57.setObjectName("read_57")
self.gridLayout_8.addWidget(self.read_57, 8, 2, 1, 1)
self.read_60 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_60.setObjectName("read_60")
self.gridLayout_8.addWidget(self.read_60, 11, 2, 1, 1)
self.read_66 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_66.setObjectName("read_66")
self.gridLayout_8.addWidget(self.read_66, 17, 2, 1, 1)
self.read_63 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_63.setObjectName("read_63")
self.gridLayout_8.addWidget(self.read_63, 14, 2, 1, 1)
self.read_61 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_61.setObjectName("read_61")
self.gridLayout_8.addWidget(self.read_61, 12, 2, 1, 1)
self.read_62 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_62.setObjectName("read_62")
self.gridLayout_8.addWidget(self.read_62, 13, 2, 1, 1)
self.read_64 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_64.setObjectName("read_64")
self.gridLayout_8.addWidget(self.read_64, 15, 2, 1, 1)
self.read_65 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_65.setObjectName("read_65")
self.gridLayout_8.addWidget(self.read_65, 16, 2, 1, 1)
self.read_68 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_68.setObjectName("read_68")
self.gridLayout_8.addWidget(self.read_68, 19, 2, 1, 1)
self.read_67 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_67.setObjectName("read_67")
self.gridLayout_8.addWidget(self.read_67, 18, 2, 1, 1)
self.read_69 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_69.setObjectName("read_69")
self.gridLayout_8.addWidget(self.read_69, 20, 2, 1, 1)
self.read_72 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_72.setObjectName("read_72")
self.gridLayout_8.addWidget(self.read_72, 23, 2, 1, 1)
self.read_70 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_70.setObjectName("read_70")
self.gridLayout_8.addWidget(self.read_70, 21, 2, 1, 1)
self.read_71 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_71.setObjectName("read_71")
self.gridLayout_8.addWidget(self.read_71, 22, 2, 1, 1)
self.read_73 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_73.setObjectName("read_73")
self.gridLayout_8.addWidget(self.read_73, 24, 2, 1, 1)
self.read_77 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_77.setObjectName("read_77")
self.gridLayout_8.addWidget(self.read_77, 28, 2, 1, 1)
```

```
self.read_76 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_76.setObjectName("read_76")
self.gridLayout_8.addWidget(self.read_76, 27, 2, 1, 1)
self.read_75 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_75.setObjectName("read_75")
self.gridLayout_8.addWidget(self.read_75, 26, 2, 1, 1)
self.read_74 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_74.setObjectName("read_74")
self.gridLayout_8.addWidget(self.read_74, 25, 2, 1, 1)
self.read_78 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_78.setObjectName("read_78")
self.gridLayout_8.addWidget(self.read_78, 29, 2, 1, 1)
self.read_79 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_79.setObjectName("read_79")
self.gridLayout_8.addWidget(self.read_79, 30, 2, 1, 1)
self.read_80 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_80.setObjectName("read_80")
self.gridLayout_8.addWidget(self.read_80, 31, 2, 1, 1)
self.read_81 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_81.setObjectName("read_81")
self.gridLayout_8.addWidget(self.read_81, 32, 2, 1, 1)
self.read_82 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_82.setObjectName("read_82")
self.gridLayout_8.addWidget(self.read_82, 33, 2, 1, 1)
self.read_83 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_83.setObjectName("read_83")
self.gridLayout_8.addWidget(self.read_83, 34, 2, 1, 1)
self.read_84 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_84.setObjectName("read_84")
self.gridLayout_8.addWidget(self.read_84, 35, 2, 1, 1)
self.read_85 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_85.setObjectName("read_85")
self.gridLayout_8.addWidget(self.read_85, 36, 2, 1, 1)
self.read_86 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_86.setObjectName("read_86")
self.gridLayout_8.addWidget(self.read_86, 37, 2, 1, 1)
self.read_87 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_87.setObjectName("read_87")
self.gridLayout_8.addWidget(self.read_87, 38, 2, 1, 1)
self.read_88 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_88.setObjectName("read_88")
self.gridLayout_8.addWidget(self.read_88, 39, 2, 1, 1)
self.read_89 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_89.setObjectName("read_89")
self.gridLayout_8.addWidget(self.read_89, 40, 2, 1, 1)
self.read_90 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_90.setObjectName("read_90")
self.gridLayout_8.addWidget(self.read_90, 41, 2, 1, 1)
self.read_91 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_91.setObjectName("read_91")
self.gridLayout_8.addWidget(self.read_91, 42, 2, 1, 1)
```

```
self.read_92 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_92.setObjectName("read_92")
self.gridLayout_8.addWidget(self.read_92, 43, 2, 1, 1)
self.read_93 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_93.setObjectName("read_93")
self.gridLayout_8.addWidget(self.read_93, 44, 2, 1, 1)
self.read_94 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_94.setObjectName("read_94")
self.gridLayout_8.addWidget(self.read_94, 45, 2, 1, 1)
self.read_95 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_95.setObjectName("read_95")
self.gridLayout_8.addWidget(self.read_95, 46, 2, 1, 1)
self.read_96 = QtWidgets.QPushButton(self.scrollContents_6)
self.read_96.setObjectName("read_96")
self.gridLayout_8.addWidget(self.read_96, 47, 2, 1, 1)
self.scrollArea_readHV_2.setWidget(self.scrollContents_6)
self.gridLayout_5.addWidget(self.groupBox_readGND, 0, 2, 1, 1)
self.scrollArea.setWidget(self.scrollAreaWidgetContents)
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 1307, 26))
self.menubar.setObjectName("menubar")
self.menuOptions = QtWidgets.QMenu(self.menubar)
self.menuOptions.setObjectName("menuOptions")
MainWindow.setMenuBar(self.menubar)
self.actionSPI_Clock = QtWidgets.QAction(MainWindow)
self.actionSPI_Clock.setObjectName("actionSPI_Clock")
self.actionNew_Board = QtWidgets.QAction(MainWindow)
self.actionNew_Board.setObjectName("actionNew_Board")
self.actionReset_Close_Board = QtWidgets.QAction(MainWindow)
self.actionReset_Close_Board.setObjectName("actionReset_Close_Board")
self.menuOptions.addAction(self.actionNew_Board)
self.menuOptions.addAction(self.actionSPI_Clock)
self.menuOptions.addAction(self.actionReset_Close_Board)
self.menubar.addAction(self.menuOptions.menuAction())

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

self.__lcdDisplays = [self.lcdNumber_1, self.lcdNumber_2, self.lcdNumber_3,
    self.lcdNumber_4, self.lcdNumber_5,
self.lcdNumber_6, self.lcdNumber_7, self.lcdNumber_8, self.lcdNumber_9,
    self.lcdNumber_10, self.lcdNumber_11,
self.lcdNumber_12, self.lcdNumber_13, self.lcdNumber_14, self.lcdNumber_15,
    self.lcdNumber_16, self.lcdNumber_17,
self.lcdNumber_18, self.lcdNumber_19, self.lcdNumber_20, self.lcdNumber_21,
    self.lcdNumber_22, self.lcdNumber_23,
self.lcdNumber_24, self.lcdNumber_25, self.lcdNumber_26, self.lcdNumber_27,
    self.lcdNumber_28, self.lcdNumber_29,
self.lcdNumber_30, self.lcdNumber_31, self.lcdNumber_32, self.lcdNumber_33,
    self.lcdNumber_34, self.lcdNumber_35,
```

```
self.lcdNumber_36, self.lcdNumber_37, self.lcdNumber_38, self.lcdNumber_39,
    self.lcdNumber_40, self.lcdNumber_41,
self.lcdNumber_42, self.lcdNumber_43, self.lcdNumber_44, self.lcdNumber_45,
    self.lcdNumber_46, self.lcdNumber_47,
self.lcdNumber_48, self.lcdNumber_49, self.lcdNumber_50, self.lcdNumber_51,
    self.lcdNumber_52, self.lcdNumber_53,
self.lcdNumber_54, self.lcdNumber_55, self.lcdNumber_56, self.lcdNumber_57,
    self.lcdNumber_58, self.lcdNumber_59,
self.lcdNumber_60, self.lcdNumber_61, self.lcdNumber_62, self.lcdNumber_63,
    self.lcdNumber_64, self.lcdNumber_65,
self.lcdNumber_66, self.lcdNumber_67, self.lcdNumber_68, self.lcdNumber_69,
    self.lcdNumber_70, self.lcdNumber_71,
self.lcdNumber_72, self.lcdNumber_73, self.lcdNumber_74, self.lcdNumber_75,
    self.lcdNumber_76, self.lcdNumber_77,
self.lcdNumber_78, self.lcdNumber_79, self.lcdNumber_80, self.lcdNumber_81,
    self.lcdNumber_82, self.lcdNumber_83,
self.lcdNumber_84, self.lcdNumber_85, self.lcdNumber_86, self.lcdNumber_87,
    self.lcdNumber_88, self.lcdNumber_89,
self.lcdNumber_90, self.lcdNumber_91, self.lcdNumber_92, self.lcdNumber_93,
    self.lcdNumber_94, self.lcdNumber_95,
self.lcdNumber_96]

self.__shieldDisplays = [self.shield1_LCD, self.shield2_LCD, self.shield3_LCD,
    self.shield4_LCD]

#Connections

self.actionNew_Board.triggered.connect(lambda: self.__initializeBoard())
self.actionSPI_Clock.triggered.connect(lambda: self.__setSPISpeed())

#Read HV Buttons connection

self.read_1.clicked.connect(lambda: self.__readHV(1))
self.read_2.clicked.connect(lambda: self.__readHV(2))
self.read_3.clicked.connect(lambda: self.__readHV(3))
self.read_4.clicked.connect(lambda: self.__readHV(4))
self.read_5.clicked.connect(lambda: self.__readHV(5))
self.read_6.clicked.connect(lambda: self.__readHV(6))
self.read_7.clicked.connect(lambda: self.__readHV(7))
self.read_8.clicked.connect(lambda: self.__readHV(8))
self.read_9.clicked.connect(lambda: self.__readHV(9))
self.read_10.clicked.connect(lambda: self.__readHV(10))
self.read_11.clicked.connect(lambda: self.__readHV(11))
self.read_12.clicked.connect(lambda: self.__readHV(12))
self.read_13.clicked.connect(lambda: self.__readHV(13))
self.read_14.clicked.connect(lambda: self.__readHV(14))
self.read_15.clicked.connect(lambda: self.__readHV(15))
self.read_16.clicked.connect(lambda: self.__readHV(16))
self.read_17.clicked.connect(lambda: self.__readHV(17))
self.read_18.clicked.connect(lambda: self.__readHV(18))
self.read_19.clicked.connect(lambda: self.__readHV(19))
```

```
        self.read_20.clicked.connect(lambda: self.__readHV(20))
        self.read_21.clicked.connect(lambda: self.__readHV(21))
        self.read_22.clicked.connect(lambda: self.__readHV(22))
        self.read_23.clicked.connect(lambda: self.__readHV(23))
        self.read_24.clicked.connect(lambda: self.__readHV(24))
        self.read_25.clicked.connect(lambda: self.__readHV(25))
        self.read_26.clicked.connect(lambda: self.__readHV(26))
        self.read_27.clicked.connect(lambda: self.__readHV(27))
        self.read_28.clicked.connect(lambda: self.__readHV(28))
        self.read_29.clicked.connect(lambda: self.__readHV(29))
        self.read_30.clicked.connect(lambda: self.__readHV(30))
        self.read_31.clicked.connect(lambda: self.__readHV(31))
        self.read_32.clicked.connect(lambda: self.__readHV(32))
        self.read_33.clicked.connect(lambda: self.__readHV(33))
        self.read_34.clicked.connect(lambda: self.__readHV(34))
        self.read_35.clicked.connect(lambda: self.__readHV(35))
        self.read_36.clicked.connect(lambda: self.__readHV(36))
        self.read_37.clicked.connect(lambda: self.__readHV(37))
        self.read_38.clicked.connect(lambda: self.__readHV(38))
        self.read_39.clicked.connect(lambda: self.__readHV(39))
        self.read_40.clicked.connect(lambda: self.__readHV(40))
        self.read_41.clicked.connect(lambda: self.__readHV(41))
        self.read_42.clicked.connect(lambda: self.__readHV(42))
        self.read_43.clicked.connect(lambda: self.__readHV(43))
        self.read_44.clicked.connect(lambda: self.__readHV(44))
        self.read_45.clicked.connect(lambda: self.__readHV(45))
        self.read_46.clicked.connect(lambda: self.__readHV(46))
        self.read_47.clicked.connect(lambda: self.__readHV(47))
        self.read_48.clicked.connect(lambda: self.__readHV(48))

        #Read GND Buttons connection

        self.read_49.clicked.connect(lambda: self.__readGND(49))
        self.read_50.clicked.connect(lambda: self.__readGND(50))
        self.read_51.clicked.connect(lambda: self.__readGND(51))
        self.read_52.clicked.connect(lambda: self.__readGND(52))
        self.read_53.clicked.connect(lambda: self.__readGND(53))
        self.read_54.clicked.connect(lambda: self.__readGND(54))
        self.read_55.clicked.connect(lambda: self.__readGND(55))
        self.read_56.clicked.connect(lambda: self.__readGND(56))
        self.read_57.clicked.connect(lambda: self.__readGND(57))
        self.read_58.clicked.connect(lambda: self.__readGND(58))
        self.read_59.clicked.connect(lambda: self.__readGND(59))
        self.read_60.clicked.connect(lambda: self.__readGND(60))
        self.read_61.clicked.connect(lambda: self.__readGND(61))
        self.read_62.clicked.connect(lambda: self.__readGND(62))
        self.read_63.clicked.connect(lambda: self.__readGND(63))
        self.read_64.clicked.connect(lambda: self.__readGND(64))
        self.read_65.clicked.connect(lambda: self.__readGND(65))
        self.read_66.clicked.connect(lambda: self.__readGND(66))
        self.read_67.clicked.connect(lambda: self.__readGND(67))
```

```
self.read_68.clicked.connect(lambda: self.__readGND(68))
self.read_69.clicked.connect(lambda: self.__readGND(69))
self.read_70.clicked.connect(lambda: self.__readGND(70))
self.read_71.clicked.connect(lambda: self.__readGND(71))
self.read_72.clicked.connect(lambda: self.__readGND(72))
self.read_73.clicked.connect(lambda: self.__readGND(73))
self.read_74.clicked.connect(lambda: self.__readGND(74))
self.read_75.clicked.connect(lambda: self.__readGND(75))
self.read_76.clicked.connect(lambda: self.__readGND(76))
self.read_77.clicked.connect(lambda: self.__readGND(77))
self.read_78.clicked.connect(lambda: self.__readGND(78))
self.read_79.clicked.connect(lambda: self.__readGND(79))
self.read_80.clicked.connect(lambda: self.__readGND(80))
self.read_81.clicked.connect(lambda: self.__readGND(81))
self.read_82.clicked.connect(lambda: self.__readGND(82))
self.read_83.clicked.connect(lambda: self.__readGND(83))
self.read_84.clicked.connect(lambda: self.__readGND(84))
self.read_85.clicked.connect(lambda: self.__readGND(85))
self.read_86.clicked.connect(lambda: self.__readGND(86))
self.read_87.clicked.connect(lambda: self.__readGND(87))
self.read_88.clicked.connect(lambda: self.__readGND(88))
self.read_89.clicked.connect(lambda: self.__readGND(89))
self.read_90.clicked.connect(lambda: self.__readGND(90))
self.read_91.clicked.connect(lambda: self.__readGND(91))
self.read_92.clicked.connect(lambda: self.__readGND(92))
self.read_93.clicked.connect(lambda: self.__readGND(93))
self.read_94.clicked.connect(lambda: self.__readGND(94))
self.read_95.clicked.connect(lambda: self.__readGND(95))
self.read_96.clicked.connect(lambda: self.__readGND(96))

self.button_setHV.clicked.connect(lambda:
    self.__setHV(self.textEdit_setHV.toPlainText()))
self.set_cable.clicked.connect(lambda:
    self.__setCable(self.textEdit_setCable.toPlainText()))

self.read_HV.clicked.connect(lambda: self.__readSupplyHV())
self.read_continuity.clicked.connect(lambda: self.__continuityRead())

self.read_shield1.clicked.connect(lambda: self.__readShield(1))
self.read_shield2.clicked.connect(lambda: self.__readShield(2))
self.read_shield3.clicked.connect(lambda: self.__readShield(3))
self.read_shield4.clicked.connect(lambda: self.__readShield(4))

self.read_all_hv.clicked.connect(lambda: self.__readAllHVs())
self.read_all_gnd.clicked.connect(lambda: self.__readAllGNDs())

self.button_enable_3V3.clicked.connect(lambda: self.__enableSupply('V3.3',
    True))
self.button_disable_3V3.clicked.connect(lambda: self.__enableSupply('V3.3',
    False))
self.button_enable_5.clicked.connect(lambda: self.__enableSupply('V5', True))
```

```
        self.button_disable_5.clicked.connect(lambda: self.__enableSupply('V5', False))
        self.button_enable_12.clicked.connect(lambda: self.__enableSupply('VP12',
            True))
        self.button_disable_12.clicked.connect(lambda: self.__enableSupply('VP12',
            False))
        self.button_enable_N12.clicked.connect(lambda: self.__enableSupply('VN12',
            True))
        self.button_disable_N12.clicked.connect(lambda: self.__enableSupply('VN12',
            False))


    def __initializeBoard(self):
        dialog1 = QtWidgets.QInputDialog(self)
        dialog1.adjustSize()
        number1, result1 = dialog1.getItem(self, "Choose a CS pin", "Raspberry Pi pins
            available", self.__CS_PINS, 0, False)
        if result1 == True:
            self.__CS_PIN = int(number1)

        dialog2 = QtWidgets.QInputDialog(self)
        number2, result2 = dialog2.getItem(self, "Choose a HV Reading Board version",
            "Versions available", self.__HVR_BOARDS, 0, False)
        if result2 == True:
            self.__HVR_BOARD = int(number2)

        board = CABLETEST(self.__spi, GPIO, self.__CS_PIN, self.__HVR_BOARD)
        self.__ctbBoard = board
        self.__ctbBoard.initBoard()

        message = QtWidgets.QMessageBox().information(self, "Information Message",
            "The board has been initialized.", QMessageBox.Ok)

    def __setSPISpeed(self):
        dialog = QtWidgets.QInputDialog(self)
        dialog.adjustSize()
        number, result = dialog.getInt(self, "Set SPI clock frequency", "Enter clock
            frequency in kHz",99600,1,125000,1)
        if result == 1:
            speed = dialog.intValue()
            speed = speed * 1000
            self.__ctbBoard.set_spiSpeed(speed)

    def __enableSupply(self, supply, onOff):
        assert supply == 'VP12' or supply == 'VN12' or supply == 'V3.3' or supply ==
            'V5'

        self.__ctbBoard.enableDisableSupplies(supply, onOff)

        if supply == 'VP12':
            if onOff == True:
                self.status_P12.setText("Enabled")
```

```
            else:
                self.status_P12.setText("Disabled")
        elif supply == 'VN12':
            if onOff == True:
                self.status_N12.setText("Enabled")
            else:
                self.status_N12.setText("Disabled")
        elif supply == 'V3.3':
            if onOff == True:
                self.status_3V3.setText("Enabled")
            else:
                self.status_3V3.setText("Disabled")
        else:
            if onOff == True:
                self.status_5.setText("Enabled")
            else:
                self.status_5.setText("Disabled")

    def __setHV(self, HV):
        assert int(HV) < 1500 and int(HV) > 0
        #self.HV_LCD.display(int(HV)) test
        self.__ctbBoard.setHV(int(HV))

    def __readSupplyHV(self):
        HV = self.__ctbBoard.readSupplyHV()
        self.HV_LCD.display(HV)

    def __readHV(self, channel):
        HV = self.__ctbBoard.readHVChannel(channel)
        #test
        #HV =12
        self.__lcdDisplays[channel-1].display(HV)

    def __readGND(self, channel):
        real_channel = channel - 48
        GND = self.__ctbBoard.readGNDChannel(real_channel)
        #TEST
        #GND = 45
        self.__lcdDisplays[channel-1].display(GND)

    def __setCable(self, cable):
        self.__ctbBoard.setHVCable(cable)

    def __continuityRead(self):
        value = self.__ctbBoard.readCERN()
        self.cern_LCD.display(value)

    def __readShield(self, number):
        value = self.__ctbBoard.readShield(number)
        self.shieldDisplays[number-1].display(value)
```

```
    def __readAllHVs(self):
        #test_voltages = [1495, 1191, 121, 1050, 443, 1063, 1130, 1166, 1353, 719,
            1176, 235, 822, 625, 997, 818, 525, 570, 1223, 1029, 248, 485, 933, 427,
            1032, 1126, 974, 200, 143, 1244, 1324, 351, 442, 1111, 366, 220, 16, 920,
            1017, 526, 1463, 1415, 1060, 100, 1393, 441, 379, 123]
        voltages = self.__ctbBoard.readAllHVs()
        for i in range(len(voltages)):
            self.__lcdDisplays[i].display(voltages[i])

    def __readAllGNDs(self):
        voltages = self.__ctbBoard.readAllGNDs()
        #test voltages = [1495, 1191, 121, 1050, 443, 1063, 1130, 1166, 1353, 719,
            1176, 235, 822, 625, 997, 818, 525, 570, 1223, 1029, 248, 485, 933, 427,
            1032, 1126, 974, 200, 143, 1244, 1324, 351, 442, 1111, 366, 220, 16, 920,
            1017, 526, 1463, 1415, 1060, 100, 1393, 441, 379, 123]
        for i in range(len(voltages)):
            self.__lcdDisplays[i+48].display(voltages[i])

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Cable Test Board GUI"))
        self.groupBox_readHV.setTitle(_translate("MainWindow", "HV Cable Readings (Lab
            Setup Only)"))
        self.label_44.setText(_translate("MainWindow", "Cable 34"))
        self.label_19.setText(_translate("MainWindow", "Cable 13"))
        self.read_1.setText(_translate("MainWindow", "Read"))
        self.label_HV2.setText(_translate("MainWindow", "Cable 2"))
        self.label_HV1.setText(_translate("MainWindow", "Cable 1"))
        self.read_2.setText(_translate("MainWindow", "Read"))
        self.label_HV3.setText(_translate("MainWindow", "Cable 3"))
        self.read_3.setText(_translate("MainWindow", "Read"))
        self.label_16.setText(_translate("MainWindow", "Cable 10"))
        self.label_43.setText(_translate("MainWindow", "Cable 33"))
        self.label_42.setText(_translate("MainWindow", "Cable 32"))
        self.label_HV8.setText(_translate("MainWindow", "Cable 8"))
        self.label_18.setText(_translate("MainWindow", "Cable 12"))
        self.label_15.setText(_translate("MainWindow", "Cable 9"))
        self.label_22.setText(_translate("MainWindow", "Cable 16"))
        self.label_HV5.setText(_translate("MainWindow", "Cable 5"))
        self.label_20.setText(_translate("MainWindow", "Cable 14"))
        self.read_4.setText(_translate("MainWindow", "Read"))
        self.label_HV6.setText(_translate("MainWindow", "Cable 6"))
        self.label_HV7.setText(_translate("MainWindow", "Cable 7"))
        self.label_17.setText(_translate("MainWindow", "Cable 11"))
        self.label_21.setText(_translate("MainWindow", "Cable 15"))
        self.label_HV4.setText(_translate("MainWindow", "Cable 4"))
        self.label_93.setText(_translate("MainWindow", "Cable 44"))
        self.label_95.setText(_translate("MainWindow", "Cable 46"))
        self.label_94.setText(_translate("MainWindow", "Cable 45"))
        self.label_97.setText(_translate("MainWindow", "Cable 48"))
        self.label_96.setText(_translate("MainWindow", "Cable 47"))
```

```
self.label_91.setText(_translate("MainWindow", "Cable 42"))
self.label_92.setText(_translate("MainWindow", "Cable 43"))
self.label_23.setText(_translate("MainWindow", "Cable 17"))
self.label_25.setText(_translate("MainWindow", "Cable 19"))
self.label_24.setText(_translate("MainWindow", "Cable 18"))
self.label_26.setText(_translate("MainWindow", "Cable 20"))
self.label_27.setText(_translate("MainWindow", "Cable 21"))
self.label_28.setText(_translate("MainWindow", "Cable 22"))
self.label_30.setText(_translate("MainWindow", "Cable 24"))
self.label_31.setText(_translate("MainWindow", "Cable 25"))
self.label_32.setText(_translate("MainWindow", "Cable 26"))
self.label_33.setText(_translate("MainWindow", "Cable 27"))
self.label_35.setText(_translate("MainWindow", "Cable 29"))
self.label_34.setText(_translate("MainWindow", "Cable 28"))
self.label_37.setText(_translate("MainWindow", "Cable 31"))
self.label_36.setText(_translate("MainWindow", "Cable 30"))
self.label_46.setText(_translate("MainWindow", "Cable 36"))
self.label_45.setText(_translate("MainWindow", "Cable 35"))
self.label_29.setText(_translate("MainWindow", "Cable 23"))
self.label_48.setText(_translate("MainWindow", "Cable 38"))
self.label_47.setText(_translate("MainWindow", "Cable 37"))
self.label_49.setText(_translate("MainWindow", "Cable 39"))
self.label_50.setText(_translate("MainWindow", "Cable 40"))
self.label_51.setText(_translate("MainWindow", "Cable 41"))
self.read_5.setText(_translate("MainWindow", "Read"))
self.read_7.setText(_translate("MainWindow", "Read"))
self.read_8.setText(_translate("MainWindow", "Read"))
self.read_6.setText(_translate("MainWindow", "Read"))
self.read_10.setText(_translate("MainWindow", "Read"))
self.read_11.setText(_translate("MainWindow", "Read"))
self.read_9.setText(_translate("MainWindow", "Read"))
self.read_12.setText(_translate("MainWindow", "Read"))
self.read_18.setText(_translate("MainWindow", "Read"))
self.read_15.setText(_translate("MainWindow", "Read"))
self.read_13.setText(_translate("MainWindow", "Read"))
self.read_14.setText(_translate("MainWindow", "Read"))
self.read_16.setText(_translate("MainWindow", "Read"))
self.read_17.setText(_translate("MainWindow", "Read"))
self.read_20.setText(_translate("MainWindow", "Read"))
self.read_19.setText(_translate("MainWindow", "Read"))
self.read_21.setText(_translate("MainWindow", "Read"))
self.read_24.setText(_translate("MainWindow", "Read"))
self.read_22.setText(_translate("MainWindow", "Read"))
self.read_23.setText(_translate("MainWindow", "Read"))
self.read_25.setText(_translate("MainWindow", "Read"))
self.read_29.setText(_translate("MainWindow", "Read"))
self.read_28.setText(_translate("MainWindow", "Read"))
self.read_27.setText(_translate("MainWindow", "Read"))
self.read_26.setText(_translate("MainWindow", "Read"))
self.read_30.setText(_translate("MainWindow", "Read"))
self.read_31.setText(_translate("MainWindow", "Read"))
```

```
        self.read_32.setText(_translate("MainWindow", "Read"))
        self.read_33.setText(_translate("MainWindow", "Read"))
        self.read_34.setText(_translate("MainWindow", "Read"))
        self.read_35.setText(_translate("MainWindow", "Read"))
        self.read_36.setText(_translate("MainWindow", "Read"))
        self.read_37.setText(_translate("MainWindow", "Read"))
        self.read_38.setText(_translate("MainWindow", "Read"))
        self.read_39.setText(_translate("MainWindow", "Read"))
        self.read_40.setText(_translate("MainWindow", "Read"))
        self.read_41.setText(_translate("MainWindow", "Read"))
        self.read_42.setText(_translate("MainWindow", "Read"))
        self.read_43.setText(_translate("MainWindow", "Read"))
        self.read_44.setText(_translate("MainWindow", "Read"))
        self.read_45.setText(_translate("MainWindow", "Read"))
        self.read_46.setText(_translate("MainWindow", "Read"))
        self.read_47.setText(_translate("MainWindow", "Read"))
        self.read_48.setText(_translate("MainWindow", "Read"))
        self.label_89.setText(_translate("MainWindow", "Readings in ADC Counts"))
        self.read_all_hv.setText(_translate("MainWindow", "Read All"))
        self.groupBox_LV.setTitle(_translate("MainWindow", "Low Voltage Enables"))
        self.button_enable_12.setText(_translate("MainWindow", "Enable"))
        self.status_N12.setText(_translate("MainWindow", "Disabled"))
        self.status_5.setText(_translate("MainWindow", "Disabled"))
        self.status_P12.setText(_translate("MainWindow", "Disabled"))
        self.button_enable_5.setText(_translate("MainWindow", "Enable"))
        self.button_enable_N12.setText(_translate("MainWindow", "Enable"))
        self.button_disable_5.setText(_translate("MainWindow", "Disable"))
        self.button_disable_12.setText(_translate("MainWindow", "Disable"))
        self.button_disable_N12.setText(_translate("MainWindow", "Disable"))
        self.button_disable_3V3.setText(_translate("MainWindow", "Disable"))
        self.status_3V3.setText(_translate("MainWindow", "Disabled"))
        self.button_enable_3V3.setText(_translate("MainWindow", "Enable"))
        self.label_3V3.setText(_translate("MainWindow", "+3V3"))
        self.label_5V.setText(_translate("MainWindow", "+5V"))
        self.label_VP12.setText(_translate("MainWindow", "+12V"))
        self.label_VN12.setText(_translate("MainWindow", "-12V"))
        self.groupBox.setTitle(_translate("MainWindow", "Set HV"))
        self.label_unit.setText(_translate("MainWindow", "V"))
        self.button_setHV.setText(_translate("MainWindow", "Set"))
        self.label_5.setText(_translate("MainWindow", "Cable"))
        self.set_cable.setText(_translate("MainWindow", "Select"))
        self.groupBox_2.setTitle(_translate("MainWindow", "Read HV"))
        self.read_HV.setText(_translate("MainWindow", "Read"))
        self.label_4.setText(_translate("MainWindow", "ADC Counts"))
        self.groupBox_3.setTitle(_translate("MainWindow", "Cable Continuity Reading
            (CERN Setup Only)"))
        self.label_6.setText(_translate("MainWindow", "ADC Counts"))
        self.read_continuity.setText(_translate("MainWindow", "Read"))
        self.groupBox_readSHIELD.setTitle(_translate("MainWindow", "Shield Cable
            Readings"))
        self.read_shield2.setText(_translate("MainWindow", "Read"))
```

```
        self.label_2.setText(_translate("MainWindow", "Cable 2"))
        self.label.setText(_translate("MainWindow", "Cable 1"))
        self.label_3.setText(_translate("MainWindow", "Cable 3"))
        self.read_shield1.setText(_translate("MainWindow", "Read"))
        self.label_10.setText(_translate("MainWindow", "Cable 4"))
        self.read_shield3.setText(_translate("MainWindow", "Read"))
        self.read_shield4.setText(_translate("MainWindow", "Read"))
        self.label_11.setText(_translate("MainWindow", "ADC Counts"))
        self.label_14.setText(_translate("MainWindow", "ADC Counts"))
        self.label_13.setText(_translate("MainWindow", "ADC Counts"))
        self.label_12.setText(_translate("MainWindow", "ADC Counts"))
        self.groupBox_readGND.setTitle(_translate("MainWindow", "GND Cable Readings
            (Lab Setup Only)"))
        self.label_90.setText(_translate("MainWindow", "Readings in ADC Counts"))
        self.read_all_gnd.setText(_translate("MainWindow", "Read All"))
        self.label_81.setText(_translate("MainWindow", "Cable 34"))
        self.label_82.setText(_translate("MainWindow", "Cable 13"))
        self.read_49.setText(_translate("MainWindow", "Read"))
        self.label_HV2_4.setText(_translate("MainWindow", "Cable 2"))
        self.label_HV1_4.setText(_translate("MainWindow", "Cable 1"))
        self.read_50.setText(_translate("MainWindow", "Read"))
        self.label_HV3_4.setText(_translate("MainWindow", "Cable 3"))
        self.read_51.setText(_translate("MainWindow", "Read"))
        self.label_83.setText(_translate("MainWindow", "Cable 10"))
        self.label_84.setText(_translate("MainWindow", "Cable 33"))
        self.label_85.setText(_translate("MainWindow", "Cable 32"))
        self.label_HV8_4.setText(_translate("MainWindow", "Cable 8"))
        self.label_86.setText(_translate("MainWindow", "Cable 12"))
        self.label_87.setText(_translate("MainWindow", "Cable 9"))
        self.label_88.setText(_translate("MainWindow", "Cable 16"))
        self.label_HV5_4.setText(_translate("MainWindow", "Cable 5"))
        self.label_149.setText(_translate("MainWindow", "Cable 14"))
        self.read_52.setText(_translate("MainWindow", "Read"))
        self.label_HV6_4.setText(_translate("MainWindow", "Cable 6"))
        self.label_HV7_4.setText(_translate("MainWindow", "Cable 7"))
        self.label_150.setText(_translate("MainWindow", "Cable 11"))
        self.label_151.setText(_translate("MainWindow", "Cable 15"))
        self.label_HV4_4.setText(_translate("MainWindow", "Cable 4"))
        self.label_152.setText(_translate("MainWindow", "Cable 44"))
        self.label_153.setText(_translate("MainWindow", "Cable 46"))
        self.label_154.setText(_translate("MainWindow", "Cable 45"))
        self.label_155.setText(_translate("MainWindow", "Cable 48"))
        self.label_156.setText(_translate("MainWindow", "Cable 47"))
        self.label_157.setText(_translate("MainWindow", "Cable 42"))
        self.label_158.setText(_translate("MainWindow", "Cable 43"))
        self.label_159.setText(_translate("MainWindow", "Cable 17"))
        self.label_160.setText(_translate("MainWindow", "Cable 19"))
        self.label_161.setText(_translate("MainWindow", "Cable 18"))
        self.label_162.setText(_translate("MainWindow", "Cable 20"))
        self.label_163.setText(_translate("MainWindow", "Cable 21"))
        self.label_164.setText(_translate("MainWindow", "Cable 22"))
```

```
        self.label_165.setText(_translate("MainWindow", "Cable 24"))
        self.label_166.setText(_translate("MainWindow", "Cable 25"))
        self.label_167.setText(_translate("MainWindow", "Cable 26"))
        self.label_168.setText(_translate("MainWindow", "Cable 27"))
        self.label_169.setText(_translate("MainWindow", "Cable 29"))
        self.label_170.setText(_translate("MainWindow", "Cable 28"))
        self.label_171.setText(_translate("MainWindow", "Cable 31"))
        self.label_172.setText(_translate("MainWindow", "Cable 30"))
        self.label_173.setText(_translate("MainWindow", "Cable 36"))
        self.label_174.setText(_translate("MainWindow", "Cable 35"))
        self.label_175.setText(_translate("MainWindow", "Cable 23"))
        self.label_176.setText(_translate("MainWindow", "Cable 38"))
        self.label_177.setText(_translate("MainWindow", "Cable 37"))
        self.label_178.setText(_translate("MainWindow", "Cable 39"))
        self.label_179.setText(_translate("MainWindow", "Cable 40"))
        self.label_180.setText(_translate("MainWindow", "Cable 41"))
        self.read_53.setText(_translate("MainWindow", "Read"))
        self.read_55.setText(_translate("MainWindow", "Read"))
        self.read_56.setText(_translate("MainWindow", "Read"))
        self.read_54.setText(_translate("MainWindow", "Read"))
        self.read_58.setText(_translate("MainWindow", "Read"))
        self.read_59.setText(_translate("MainWindow", "Read"))
        self.read_57.setText(_translate("MainWindow", "Read"))
        self.read_60.setText(_translate("MainWindow", "Read"))
        self.read_66.setText(_translate("MainWindow", "Read"))
        self.read_63.setText(_translate("MainWindow", "Read"))
        self.read_61.setText(_translate("MainWindow", "Read"))
        self.read_62.setText(_translate("MainWindow", "Read"))
        self.read_64.setText(_translate("MainWindow", "Read"))
        self.read_65.setText(_translate("MainWindow", "Read"))
        self.read_68.setText(_translate("MainWindow", "Read"))
        self.read_67.setText(_translate("MainWindow", "Read"))
        self.read_69.setText(_translate("MainWindow", "Read"))
        self.read_72.setText(_translate("MainWindow", "Read"))
        self.read_70.setText(_translate("MainWindow", "Read"))
        self.read_71.setText(_translate("MainWindow", "Read"))
        self.read_73.setText(_translate("MainWindow", "Read"))
        self.read_77.setText(_translate("MainWindow", "Read"))
        self.read_76.setText(_translate("MainWindow", "Read"))
        self.read_75.setText(_translate("MainWindow", "Read"))
        self.read_74.setText(_translate("MainWindow", "Read"))
        self.read_78.setText(_translate("MainWindow", "Read"))
        self.read_79.setText(_translate("MainWindow", "Read"))
        self.read_80.setText(_translate("MainWindow", "Read"))
        self.read_81.setText(_translate("MainWindow", "Read"))
        self.read_82.setText(_translate("MainWindow", "Read"))
        self.read_83.setText(_translate("MainWindow", "Read"))
        self.read_84.setText(_translate("MainWindow", "Read"))
        self.read_85.setText(_translate("MainWindow", "Read"))
        self.read_86.setText(_translate("MainWindow", "Read"))
        self.read_87.setText(_translate("MainWindow", "Read"))
```

```
        self.read_88.setText(_translate("MainWindow", "Read"))
        self.read_89.setText(_translate("MainWindow", "Read"))
        self.read_90.setText(_translate("MainWindow", "Read"))
        self.read_91.setText(_translate("MainWindow", "Read"))
        self.read_92.setText(_translate("MainWindow", "Read"))
        self.read_93.setText(_translate("MainWindow", "Read"))
        self.read_94.setText(_translate("MainWindow", "Read"))
        self.read_95.setText(_translate("MainWindow", "Read"))
        self.read_96.setText(_translate("MainWindow", "Read"))
        self.menuOptions.setTitle(_translate("MainWindow", "Options"))
        self.actionSPI_Clock.setText(_translate("MainWindow", "Set SPI Clock Speed"))
        self.actionNew_Board.setText(_translate("MainWindow", "Initialize Board"))
        self.actionNew_Board.setStatusTip(_translate("MainWindow", "Add new board and
            specify CS pin."))
        self.actionNew_Board.setShortcut(_translate("MainWindow", "Ctrl+N"))
        self.actionReset_Close_Board.setText(_translate("MainWindow", "Reset/Close
            Board"))

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Appendix E

HV Remote Stability Test Results

					V as a function of ADC counts		V as a function of DAC counts	
Ch	ADC		Voltmeter					
	2500	3000	2500	3000	Calibration factor	Offset	Calibration factor	Offset
1	2414,494	2893,805	602,449	722,809	0,251	-3,855	0,241	0,649
2	2428,324	2910,259	605,741	726,776	0,251	-4,119	0,242	0,565
3	2431,088	2913,905	606,987	728,273	0,251	-3,712	0,243	0,558
4	2435,667	2919,160	604,152	724,832	0,250	-3,788	0,241	0,755
5	2429,017	2911,638	603,956	724,711	0,250	-3,801	0,242	0,181
6	2416,113	2896,768	605,233	726,242	0,252	-3,049	0,242	0,184
7	2430,616	2913,551	605,768	726,869	0,251	-3,734	0,242	0,263
8	2428,351	2910,825	604,323	725,130	0,250	-3,713	0,242	0,289
9	2419,663	2900,121	607,090	728,392	0,252	-3,807	0,243	0,579
10	2430,573	2913,479	606,070	727,182	0,251	-3,516	0,242	0,508
11	2411,943	2891,411	601,647	721,967	0,251	-3,620	0,241	0,044
12	2419,175	2899,810	602,483	722,944	0,251	-3,837	0,241	0,174
13	2423,184	2904,496	603,458	724,089	0,251	-3,865	0,241	0,301
14	2436,310	2920,514	605,931	727,068	0,250	-3,578	0,242	0,247
15	2423,880	2905,775	603,951	724,790	0,251	-3,859	0,242	-0,246
16	2423,683	2905,545	603,494	724,194	0,250	-3,603	0,241	-0,002
17	2428,033	2910,688	602,136	723,379	0,251	-7,789	0,242	-4,083
18	2435,460	2919,521	607,057	728,469	0,251	-3,806	0,243	-0,005
19	2416,952	2897,650	602,751	723,487	0,251	-4,308	0,241	-0,926
20	2426,530	2908,434	602,114	722,949	0,251	-6,328	0,242	-2,062
21	2428,646	2911,656	605,853	727,076	0,251	-3,676	0,242	-0,263
22	2421,840	2903,462	602,919	723,484	0,250	-3,347	0,241	0,089
23	2428,342	2911,085	603,113	723,759	0,250	-3,771	0,241	-0,116
24	2433,272	2916,777	606,686	727,940	0,251	-3,535	0,243	0,414
25	2436,590	2920,508	607,259	728,673	0,251	-4,075	0,243	0,190
26	2442,790	2928,005	606,610	728,030	0,250	-4,672	0,243	-0,490
27	2420,683	2901,718	604,045	724,882	0,251	-4,036	0,242	-0,140
28	1699,741	1699,123	420,692	420,880	-0,303	936,191	0,000	419,755
29	2443,942	2929,726	610,504	732,637	0,251	-3,940	0,244	-0,162
30	2442,172	2927,085	609,185	730,951	0,251	-4,067	0,244	0,354
31	2426,524	2908,700	604,391	725,270	0,251	-3,923	0,242	-0,002
32	2431,129	2914,047	605,522	726,599	0,251	-4,011	0,242	0,135
33	2422,544	2904,372	605,110	726,129	0,251	-3,354	0,242	0,013
34	2425,890	2908,514	605,860	727,138	0,251	-3,740	0,243	-0,531
35	2410,479	2889,549	604,241	725,023	0,252	-3,484	0,242	0,331
36	2433,849	2917,279	603,644	724,311	0,250	-3,853	0,241	0,314
37	2435,349	2919,121	603,946	724,734	0,250	-4,110	0,242	0,006
38	2427,695	2909,894	605,348	726,394	0,251	-4,076	0,242	0,117
39	2405,457	2883,795	601,944	722,362	0,252	-3,610	0,241	-0,143
40	2418,773	2899,202	601,583	721,809	0,250	-3,709	0,240	0,452
41	2462,390	2947,101	607,407	728,801	0,250	-9,287	0,243	0,438
42	2441,624	2921,717	604,801	725,682	0,252	-9,967	0,242	0,396
43	2445,912	2924,297	606,088	727,192	0,253	-13,096	0,242	0,571
44	2449,286	2929,340	601,635	720,564	0,248	-5,153	0,238	6,989
45	2427,925	2910,354	606,642	727,857	0,251	-3,401	0,242	0,564
46	2435,456	2919,189	607,120	728,444	0,251	-3,713	0,243	0,498
47	2425,279	2906,864	607,265	728,581	0,252	-3,687	0,243	0,685
48	2440,025	2924,847	607,644	729,071	0,250	-3,480	0,243	0,506

Table E.1: Mean values of ADC and voltmeter measurements for each channel and each DAC order (2500 and 3000 counts). Calibration and offset factors calculated for each channel.