VMM3a/SRS Event Counter Implementation for Synchronization of Data Acquisition Systems

Michael Johannes Paul Vogt

Bachelorarbeit in Physik angefertigt im

Physikalischen Institut

vorgelegt der MathematischNaturwissenschaftlichen Fakultät

der

Rheinischen FriedrichWilhelmsUniversität Bonn

Juli 2025

Contents

Introduction

Micro-Pattern Gaseous Detectors (MPGDs) are particle detectors that combine advantages of gaseous detectors, such as low material budget, with the possibility of high spatial density readout [1, ch. 7.9]. In order to provide a unified method of reading out data from them, the Scalable Readout System (SRS) was developed by the RD51[1] collaboration. It comprises front-end electronics that attach directly to a circuit board on the detector to digitize its signals, and back-end electronics used for configuration and data transfer. Over time, multiple different front-end application-specific integrated circuits (ASICs) were successfully integrated into the SRS [2]. Among these are the APV25 [2], Timepix3 [3] and VMM3a [4], the latter of which was used for the purposes of this thesis.

Experimental setups can make use of multiple detectors to measure the trajectories of particles. The detection of one or multiple particles by the detectors is called an *event*, usually defined by the arrival of a trigger signal at the readouts. This trigger signal can be generated by a central source such as a scintillation detector [1, ch. 13.5.1].

Determining whether data from two different detectors belongs to the same event is a non-trivial task: Synchronization based on time data can only be achieved by providing the same clock signal to the readouts of all detectors. The electronics within one SRS system are synchronized since they receive the same clock signal from a common source [5]. However, this synchronization is not given when using a non-SRS readout. A different method of synchronization is to count the amount of arriving triggers in all readouts and match events with equal trigger counter values. However, if one of the counters misses a trigger, e.g. due to imperfections in electronics, the counters will be unequal for all following events, leading to incorrect matching.

Thus one of the most robust methods of synchronizing events is to produce a trigger counter (also interchangeably called "event counter" in this thesis) in only one readout and send this counter to all readouts involved. The SRS is a natural choice as the generator of this counter due to its configurability and reusability. Trigger counter functionality was previously implemented for the SRS in order to be able to synchronize a beam telescope using the APV25 readout front-end with a detector under test. The latter is read out using a non-SRS readout system that receives the event counter generated by the SRS. The purpose of this thesis is the development of similar event counter functionality for the more modern VMM3a front-end which has higher performance than the APV25 [4].

To handle configuration of front-end electronics and transfer of data between front-end and computer, the SRS contains a component called Front-End Concentrator (FEC). This is a printed circuit board (PCB) that contains, among other parts, a field-programmable gate array (FPGA) [6]. The event counter was implemented in the firmware of this FPGA.

In chapter 1, this thesis will introduce the physical background that enables an understanding of the detector used for later tests. Chapter 2 continues with an explanation of the technical aspects of detector readout, in particular to introduce concepts required for chapter 3. This describes the modifications made to the existing FEC firmware to implement the event counter. In the following chapter 4, the setup used to test this new functionality will be described. Lastly, in chapter 5, the resulting data will be analyzed and discussed. A relatively in-depth outlook in chapter 6 is also included, describing the results of using the VMM3a event counter in a test beam environment as well as future possibilities.

[1] Former collaboration for MPGDs. Today RD51 is succeeded by the DRD1 collaboration which also includes other kinds of gaseous detectors.

1. Physics Background

The Standard Model of particle physics is the theory classifying all known elementary particles and the interactions between them. It is the result of a long search for the most fundamental, undividable constituents of the universe, going to ever smaller scales and masses [7, Ch. 1]. This development required the use of increasingly sophisticated technologies to detect the existence of particles, both fundamental and composite, and measure variables such as their energy, momentum or mass. Devices that do this are called *particle detectors*, or simply *detectors* in this thesis.

To measure properties of particles precisely, a large amount of data is required [1, Ch. 2]. Therefore most modern detectors produce electrical signals which can be digitized and then processed in large quantities using computers. The required steps to go from electrical signals produced in a detector to a collection of digital information in a computer are called *readout*. Detector readout electronics, like detectors themselves, are being actively developed and improved upon. The following section (1.1) explains gaseous detectors, which are a large class of particle detectors. Section 1.2 then briefly describes a source of particles frequently used in a laboratory environment. The following chapter (2) details the functionality of one particular detector readout system called the Scalable Readout System.

1.1. Gaseous Detectors

Gaseous detectors are particle detectors that use a gas as an interaction medium to detect charged particles. The basic principle of this is the ionization of gas molecules by the arriving particle. This enables electronic readout based on the signal induced by the produced electrons and/or ions in electrodes of the detector. Often a region in the gas medium with a strong electric field is used to amplify the amount of electron/ion pairs in a process called gas amplification.

The information in this section is based on [1, ch. 4 and 7] unless noted otherwise.

1.1.1. Drift Velocity and Gas Amplification

When applying an electric field to a region of gas containing free electrons, the electrons will be accelerated until they collide with a gas molecule and lose energy. This happens repeatedly and results in an average velocity of the electrons, called drift velocity. The same process applies to positive ions which are accelerated in the opposite direction.

By applying a sufficiently large electric field, the drift velocity of the electrons can reach a point at which they have enough energy to cause ionization themselves. This is called secondary ionization as opposed to the primary ionization caused directly by collisions of the incoming charged particle with gas molecules. The secondary ionization frees more electrons that can cause further secondary ionization and the process repeats itself, resulting in an avalanche of electrons.

The ratio of the total number of ionisations to primary ionisations is called gas gain G. To a certain extent, this number is independent of the number of primary ionisations, meaning that the total amount of ionisations is proportional to the primary ionisations. This proportionality applies for $10^3 \lesssim G \lesssim 10^5$.

When the field strength is too high or too many primary ionizations have taken place, the electron avalanche can reach a size that enables a conductive plasma to form between the electrodes. This connects them electrically which enables a current to flow between them so that the electrodes discharge in a spark that can cause damage to the electrodes.

1.1.2. The MicroMegas Detector

The kind of detector used in this thesis has a gas volume with a cathode placed above multiple parallel anode strips, see Fig. 1.1. A conductive mesh between them separates the gas volume into a drift region between cathode and mesh, where the electrons only drift towards the mesh, and an amplification region between mesh and anodes, where the field strength is large enough to cause gas amplification. The gas amplification creates a cloud of electrons and ions with electrons drifting toward the anodes and ions toward the mesh, both of them creating induced signals. In the following description the cathode will be considered to be at the top and the anodes at the bottom of the detector, though it should be noted that the detector can be used in any other orientation as well.

The height of the drift region is a few millimeters with a field strength on the order of $1\,\mathrm{kV\,cm^{-1}}$ and the distance between mesh and anodes is roughly $100\,\mu\mathrm{m}$ with a higher field strength around $40\,\mathrm{kV\,cm^{-1}}$. The mesh with holes on a scale of tens of micrometers is used in order to be able to apply an electric potential but still allow electrons to pass from the drift region to the amplification region. Because of it, this kind of detector is called Micromesh Gaseous Structure, or MicroMegas.

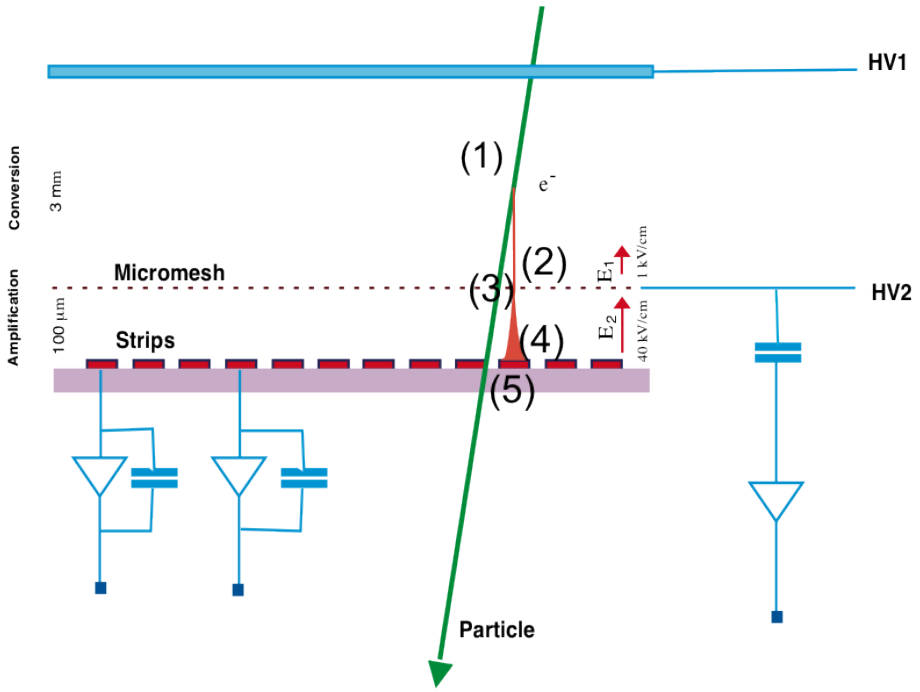


Figure 1.1.: Illustration of the working principle of a MicroMegas detector. A charged particle passes through the detection volume, ionizing gas molecules in the conversion gap (above the mesh, also called drift region) along its way (1). Electrons drift (2) towards the mesh (3) due to the electric field. They pass through the mesh into the amplification region where the electric field is sufficiently large to enable gas amplification (4). The electrons created in this process then induce a signal in the anode (5). The specified distances and field strengths are example values. Picture from [8]

For the test setup described in chapter 4, a variation of the MicroMegas design was used that has the readout strips placed below a layer of a resistive material called Diamond-Like Carbon (DLC) [9]. This has advantages for protection against discharge effects since the resistive layer limits the discharge current. In this case the DLC layer is the anode for the amplification voltage while the strips are used only for readout. The exact configuraton is shown in Fig. 1.2. Below the 100 nm of DLC, the strips are embedded between thicker layers of polyimide (also known as Kapton) and prepreg. Glued below the prepreg is another polyimide layer with another set of strips on its bottom. These are also parallel to each other but at a right angle to the top strips. The bottom strips are wider than the top strips to account for the fact that they are farther away from the gas volume and would thus receive a smaller induced signal from the electrons [9].

The two layers of strips running orthogonally to each other can be used to locate arriving particles two-dimensionally. The strips can be read out by multichannel readout electronics called VMM hybrids (see Sec. 2.1). These are placed on connectors on the detector's PCB which connect each channel to one strip. For any event, the position of the measured strip(s) in one layer can essentially be used as the x coordinate of the particle and the position in the other layer as the y coordinate. Instead of x axis and y axis, the two layers are conventionally called plane 0 and plane 1 when used to reconstruct position information.

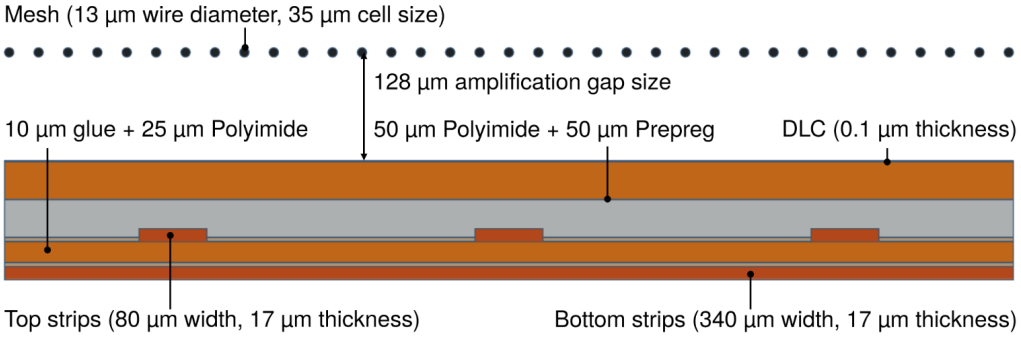


Figure 1.2.: Side-view of the mesh and bottom layer of the detector used in the test setup described in chapter 4, taken from [9]. Closest to the mesh is a 0.1 ţm resistive layer of DLC and below it a 50 ţm layer of polyimide. Below this, the top strips are fixed between a 50 ţm prepreg layer and a thinner polyimide layer. Below the bottom polyimide layer are the bottom strips.

MicroMegas detectors are designed for use in the proportional region of gas gain. For the detector used here, proportionality is given for voltages roughly between 580 V and 690 V when using a 70:30 Ar/CO_2 gas mixture [9]. Another particularity of the detector used is that the wires of the mesh are relatively close together at 730 per inch [9]. This results in a higher ratio between conductor and hole area which provides a more uniform electric field but also leaves less space for electrons to pass through.

It should be noted that this particular detector was mainly chosen because of its availability in the GDD group's[2] laboratory at CERN where most of the tests of the event counter were performed, and not because of its specific characteristics. This is because the analysis of the event counter's functionality has no strict requirements on specific detector characteristics.

[2] `https://gdd.web.cern.ch/`

1.2. Iron-55

To test particle detectors and their readout, often a source of particles is required. Radioactive materials provide reliable and often portable sources of particles. One such material is Iron-55 (^{55}Fe), a radioactive isotope frequently used to test particle detectors in a laboratory environment [1, Sec. 3.5, App. A.2]. It emits X-rays predominantly at an energy of 5.9 keV. In a detector where photons from an ^{55}Fe source arrive, the so-called *photopeak*, caused by photons that leave all of their energy in the detection medium, can be measured. Photons themselves are not charged particles but can also cause ionization in the detection medium by means of the photoelectric effect. The primary photoelectron stays in the detector and causes further ionization, resulting in a number of freed electrons that is proportional to the initially deposited energy of 5.9 keV. The readout electronics then convert this charge to a voltage pulse of proportional amplitude. This is why the terms *energy*, *charge* and *amplitude* are often used interchangeably in the context of detector data.

A second, less energetic peak can occur through processes where a discrete amount of energy escapes from the detector. An example of this is ionization of the K shell of an atom in the detection medium and escaping of the photon emitted when the shell is filled up again from a higher shell. This peak is an effect of the detector medium instead of the source and called *escape peak*. The spectrum of ^{55}Fe with the photopeak and escape peak as measured in a gaseous detector containing Argon is shown in Fig. 1.3.

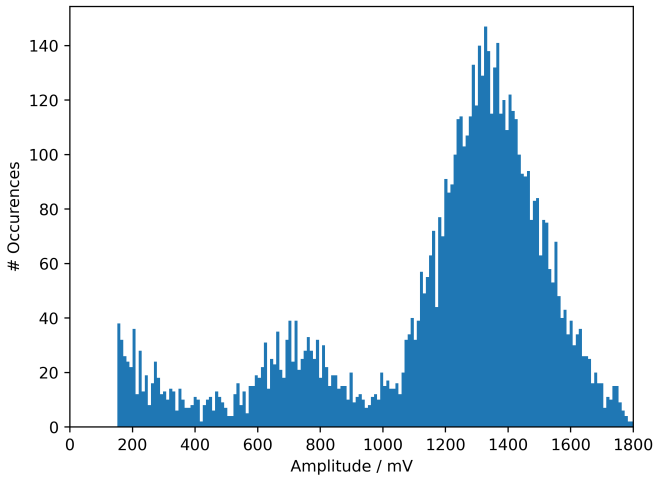


Figure 1.3.: Spectrum of ^{55}Fe measured using the MicroMegas detector described in section 1.1.2 with a 70:30 Ar/CO_2 gas mixture. The detector was read out using a multichannel analyzer. The photopeak can be seen at roughly 1350 mV and the escape peak at roughly 700 mV. The increase of counts toward the lower end of the spectrum is likely due to noise. The voltage values are not comparable to those from chapter 5 since the setup was later rebuilt with different gain settings.

2. Technical Background

This chapter introduces the Scalable Readout System (SRS), which was used to generate the event counter, in section 2.1. It details the functionality of FPGAs, which are an essential component of this system, in section 2.2. Section 2.3 then explains how FPGA configurations, called firmware, are designed and introduces the particular SRS firmware used in this thesis.

2.1. Scalable Readout System

The Scalable Readout System (SRS) was developed by RD51, a collaboration focused on Micro-Pattern Gaseous Detectors, as a unified way to read out different kinds of detectors. It splits readout into a front-end, connected directly to the detector to be read out, and a back-end, responsible for configuring the front-end and transferring the acquired data to a computer [2]. This scheme is shown in Fig. 2.1 and further explained in the following paragraphs.

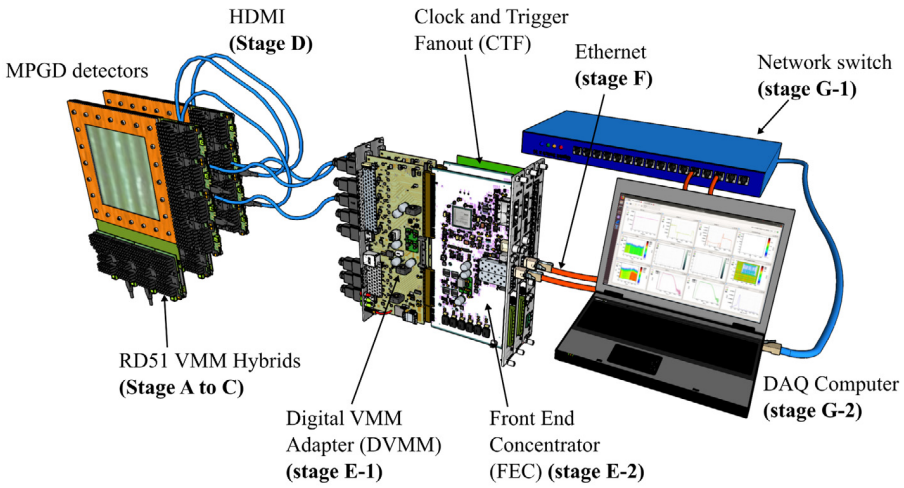


Figure 2.1.: Drawing taken from [10] showing the stages of the SRS readout for the VMM3a front-end. Attached to the detector are the hybrids (A to C) which connect to the FEC (E-2) through HDMI cables (D) and the Digital VMM (DVMM) adapter card (E-1). In the case of using multiple FECs, their clocks and trigger inputs are driven centrally by the Clock and Trigger Fanout (CTF) card. Data is transferred from the FEC(s) to a computer (G-2) over Ethernet (F), with the option to use a network switch (G-1) to connect multiple FECs to the same computer.

The front-end electronics are the step of the readout chain closest to the detector, responsible for extracting and digitizing information from it [4], such as the arrival time and amplitude of the signal. For this purpose, several different Application-Specific Integrated Circuits (ASICs) for multi-channel readout have been developed independently from the SRS. Pre-existing ASICs have been integrated into the SRS by designing PCBs that contain one or multiple front-end ASICs as well as some additional electronics

that, among other things, provide a connector to the detector and a connector for data transfer to the back-end electronics. These PCBs specific to the used front-end ASIC are called *hybrids* [2]. Examples of ASICs integrated into the SRS are the APV25 [2] and VMM3a [4].

The SRS includes a unified, front-end-independent back-end PCB. Besides configuration of front-end electronics, its main purpose is the transfer of data from one or more hybrids to a computer using an Ethernet connection. Since it can concentrate data from several front-end hybrids into one stream, it is called Front-End Concentrator (FEC) [6]. Despite its name, the FEC itself is considered part of the back-end. In order to connect a specific front-end to the FEC, an adapter card is required since the FEC hardware is the same independently of the front-end used [2]. For the VMM3a front-end, this is the Digital VMM (DVMM) adapter card.

The FEC contains a Xilinx Virtex-6 FPGA [11] which is a reprogrammable electronic component, enabling the use of the FEC in combination with various front-end specific data formats without changing its physical configuration [2]. The following section (2.2) further describes the functionality and configuration of FPGAs.

The Ethernet connection is not only used for data transfer but also for configuration of the FEC and VMMs from a computer. The configuration software used for the VMM3a front-end is called *VMM slow control*. It also has functionality to calibrate the behaviour of the VMMs. For example, the analog digital converters (ADCs) of each channel can be calibrated. Apart from the necessary connectors for the Ethernet link and attachment of adapter cards, the FEC has multiple other connectors (see Fig. 2.2). Relevant to this thesis are the following:

- The JTAG connector, used to configure the FPGA from a connected computer.
- LEMO connectors called *NIM in* (NIM input) and *NIM out* (NIM output).
- A connector called *J12* with 16 pins, 12 of which are connected to the FPGA and can be used for differential signaling [12, 13].

Nuclear Instrumentation Modules (NIM) is a standard for modular electronics instrumentation used frequently in detector setups. It defines physical and electrical specifications such as voltage levels for logic signals [14]. The NIM input and output enable the FEC's FPGA to interface with the logic levels generated and expected by NIM modules.

When using multiple FECs at once, an additional card, called Clock and Trigger Fanout (CTF), can be used to provide the same clock and trigger signal to all FECs using Ethernet connections [5]. This enables synchronization of data from the FECs based on time, since they have the same clock. If the FECs each generated their own clock, this would not be possible since the frequencies of independent oscillators are impossible to perfectly match up. Thus the measured clock cycle corresponding to the same point in time would drift apart across the different FECs.

2.2. FPGA Basics

Many of the FEC's capabilities are driven by its included field-programmable gate array (FPGA) [2]. An FPGA is an integrated circuit which functions as a freely reconfigurable logic circuit that can achieve high complexities. This configuration, the *firmware*, is usually described using Hardware Description Languages (HDLs) such as VHDL or Verilog. The event counter implementation in this thesis consists mainly of changes made to the VMM3a-specific firmware on the FEC's FPGA which is largely written in VHDL. In the following, essential FPGA-related concepts needed to understand the changes made to the firmware will be explained, based on [15] unless stated otherwise.

Figure 2.2.: Side (left) and front (right) view of the FEC. All ports on the front panel are labeled. *J1* is the JTAG port. *PX2* are the NIM input and output. *J11* contains a pluggable network interface for the Ethernet connection to a computer. *J12* is an additional connector accessible from the FPGA. The large green 9-pin port is used to connect the power supply. The FPGA sits under the black fan visible in the side view.

2.2.1. Types of Digital Logic

In digital logic, there is a distinction between circuits whose output depends only on their current inputs, and time dependent circuits whose output can additionally depend on previous input/output states. The former is called combinational logic and the latter sequential logic. Both of them can be implemented on FPGAs.

The behaviour of any combinational logic circuit can be entirely described by listing all possible inputs and the resulting outputs. By saving the output states corresponding to every possible input in some kind of reprogrammable storage, one can thus reproduce any combinational circuit. In FPGAs, such storages are called lookup tables (LUTs). The required amount of storage addresses corresponds to the number of possible input combinations which is 2^n for a circuit with n inputs.

Sequential logic depends not only on current inputs but also on previous input/output states of the circuit. Thus there is a concept of "time" or "steps". the time scale is governed by a clock signal which periodically switches between 0 and 1. The transition from 0 to 1 is called a rising clock edge and from 1 to 0 a falling clock edge. The time T of one clock cycle (time between two rising clock edges) is the clock period and its inverse the clock frequency $f = 1/T$. The dependence on previous states can be realized by so-called flip-flops, which are elements that store their input and output the stored value, with the clock signal defining when the input is stored.

Any arbitrary combinational and sequential logic can thus be implemented by using LUTs and flip-flops. This is what an FPGA does, with its free reconfigurability (it is reprogrammable "in the field") being enabled by the possibility of reprogramming LUTs as well as connections between LUTs and flip-flops. In order to receive information from and provide their output state to the outside world, FPGAs usually contain several input/output connections. A clock signal used for the flip-flops can be provided from outside or generated in specialized oscillator components on the FPGA [16].

In practice, configuring the FPGA is primarily the process of translating HDL code into LUT configurations and connections between LUTs and flip-flops, and then mapping the result to the actual hardware available on the targeted FPGA. Both of these steps are done automatically by computer software. The manual work required to design a circuit for an FPGA is mostly the programming of the FPGA firmware, explained in section 2.3.

2.2.2. Flip-Flops

A flip-flop is a logical element that outputs a stored state. It stores the current state of its data input during specific phases of the clock signal, which is a secondary input to the flip-flop (see Fig. 2.3). For example, a flip-flop may store an input signal only on rising clock edges. This component can be used as the basic element enabling storage of information over time, which is required to build sequential logic.

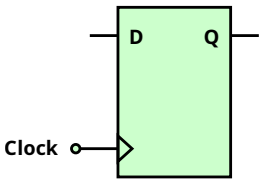


Figure 2.3.: Diagram of a simple flip-flop, adapted from [17]. It stores the value on input D (Data) whenever there is a rising edge of the Clock and outputs the stored value on Q.

In practice, the state stored by the flip-flop not only depends on its input in the moment of the clock edge but also on a time period before and after the edge, called *setup time* and *hold time* respectively.

If the input changes during this time window, the flip-flop's output has a chance to become metastable, meaning that its state is not clearly defined until it resolves into either a 0 or a 1. The time until the state resolves is also governed by chance.

Metastable states should generally be avoided because of their probabilistic nature. Since metastability only occurs randomly in some cases of setup/hold time violation, its probability can be reduced by passing the output of a possibly metastable flip-flop to a second flip-flop driven by the same clock. This technique is called double-flopping. Using more than two flip-flops would decrease the chance even more.

2.2.3. Asynchronous Signals and Clock Domains

There are multiple cases that can lead to unwanted timing or metastable states. Not all signals are synchronous to a clock: Asynchronous signals can include user input or a trigger caused by an arriving particle, for example. Double-flopping can be employed in order to safely use them in sequential logic synchronous to a clock.

In large FPGA designs, there are often multiple different clocks, e.g. due to varying internal constraints or externally connected devices having different clock frequency requirements. Signals driven by the same clock are said to belong to the same *clock domain*. Connecting the output of a flip-flop driven by clock A to the input of another driven by clock B, metastability can occur if the rising edge of clock B arrives shortly after that of clock A. This case cannot be avoided if the clocks do not have a clear phase relationship, e.g. if they come from two separate oscillators. In the case of crossing from a faster to a slower clock domain, there is additionally the potential issue of missing a signal that is shorter than a clock cycle of the slower clock. Avoiding this issue requires additional steps such as lengthening the input signal or using special so-called first in, first out storages.

The Virtex-6 has several additional components besides its collections of LUTs and flip-flops. Among them are Phase-Locked Loops (PLL) [16]. These are components that can produce multiple different clock frequencies at once by multiplying and/or dividing from a base frequency. For example, a PLL can create a 100 MHz clock from a 40 MHz clock by creating a clock frequency multiplied by 5 and then dividing the frequency by 2. Clocks generated by the same PLL, as opposed to separate oscillators, have a predictable phase relationship. Their edges can nonetheless arrive closely after one another, for example if the frequencies of the generated clocks are close to each other but not the same.

2.2.4. Shift Registers

A shift register is a series of multiple flip-flops, the output of each being fed into the input of the next. All of the comprising flip-flops are driven by the same clock and thus the content of the register will be shifted every clock cycle by one bit. Here, the shifting will be considered to go from right to left, so the "rightmost" flip-flop is the one whose input and the "leftmost" flip-flop the one whose output is connected to no other flip-flop in the chain. A 4-bit shift register is shown in Fig. 2.4.

Shift registers can be used to serialize or deserialize data. Consider a register of n flip-flops, also called *bits* when viewed as an element of data. If all bits of the register are written in parallel, i.e. in the same clock cycle, the leftmost bit will go through all the bits' original values from left to right in the following $n - 1$ clock cycles. If a time-dependent signal of n bits is input to the rightmost bit in the chain, the register will simultaneously contain the entire signal after n clock cycles, with the first (oldest) bit in the leftmost flip-flop and the last (newest) in the rightmost flip-flop.

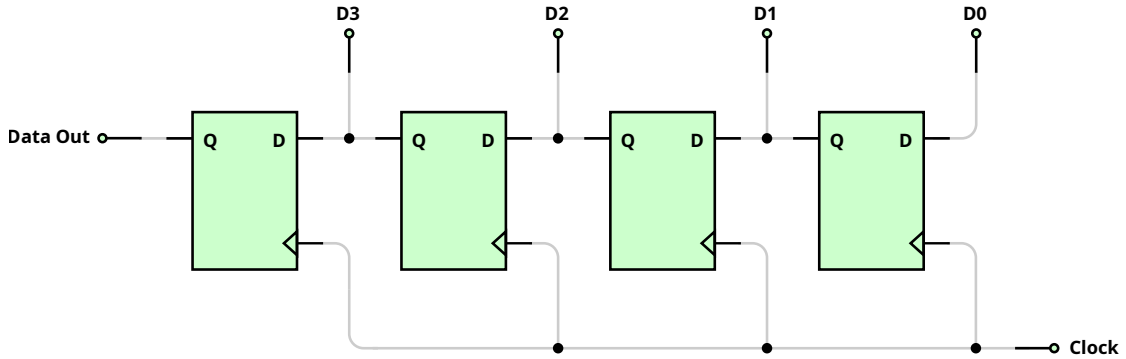


Figure 2.4.: Diagram adapted from [17] of a shift register connected for parallel input (on `D3` to `D0`) and serial output (on `Data Out`). The data inputs are again labeled `D` and the outputs `Q`. Each flip-flop receives the same `Clock` signal. Diagrams of shift registers are often drawn with the flip-flops shifting from left to right. However, the shift register later described in section 3.2 is shifting from right to left if one considers the most significant bit of its parallel input to be on the left.

2.3. FPGA Firmware

This section explains HDL concepts relevant to the description of the VMM3a/SRS firmware and the changes made to it. It then details existing parts of the firmware relevant to the event counter implementation. These descriptions of the firmware are based on the firmware code itself [13].

2.3.1. HDL Basics

This section uses VHDL, based on [18], as an example since the event counter was implemented almost exclusively in VHDL. However, similar concepts apply also to Verilog.

VHDL code has a similar appearance to code of programming languages such as C but works very differently in practice. An FPGA, in contrast to a computer processor (CPU), does not execute instructions one after another, but contains connections between logical elements that can all work in parallel. This is illustrated for example by the statement (using the assignment operator `<=`)

```
C <= A and B;
```

In a conventional programming language, `C` would be set to the current value of `A and B` and not change afterwards. In VHDL on the other hand, this statement permanently *connects* `C` to the output of an AND gate (realized by a LUT) with `A` and `B` as inputs, meaning that `C` will change accordingly to changes in `A` and `B`.

To define sequential logic, one uses `process` blocks which contain instructions to be executed when there is a change in signals set in a sensitivity list. For example the process

```
process(clk)
begin
    if rising_edge(clk) then
        counter <= counter+1;
    end if;
end process;
```

has `clk` in its sensitivity list, meaning that it starts on any change of the value of `clk`. If this change is a rising edge, `counter` will be increased by 1.

VHDL and Verilog allow the grouping of sub-circuits into *modules* with specific inputs and outputs. These modules can be instanced and connected to other modules. Besides inputs and outputs, units of data are mainly defined as so-called *signals*, which are very roughly analogous to a variable in a programming language. Depending on how they are used, signals are simply named connections between logical elements, or they can create flip-flops. The above example would create flip-flops for `counter` since the value of `counter` from the previous clock cycle needs to be known to update `counter` in the current cycle.

Inputs, outputs or signals always have a pre-defined data type. The most important data types are `std_logic` and `std_logic_vector`. These types can be used to represent one bit or multiple bits, respectively. In the above example, `counter` may be defined as a `std_logic_vector` of length 4, initialized to `0000`. The code would then result in the creation of 4 flip-flops and the value of `counter` would overflow after $2^4 = 16$ cycles of `clk`.

2.3.2. The VMM3a-SRS FEC Firmware

The VMM3a ASIC's readout channels have set thresholds and any signal that goes above a channel's threshold (called a *hit*) will be transferred to the FEC [19]. One branch of the FEC firmware implements a triggered mode which does not save all hits but only those corresponding to a specific time around a trigger signal arriving at the FEC. This time interval around the trigger is simply called *window* [12]. The trigger signal may be generated by another detector used specifically for this purpose, such as a scintillation counter for example. The event counter implementation is based on this triggered version of the firmware, since the event counter is inherently a counter of trigger signals.

The trigger is received and the window updated in the module `vmm3unit` which is responsible for receiving data from the VMM hybrids and creating the data stream to be sent over Ethernet. This data stream will also be simply called "VMM data stream" from now on. The module gets an input `trgin` containing either the NIM input signal or the trigger from the CTF, depending on whether a CTF is connected. It contains a process that synchronizes the trigger to the clock and shortens it to be only one clock cycle long. This synchronized, shortened trigger is called `trgin_sync`. The module is also connected to the NIM output through the output `trgout`.

Parameters for the window start time (how many clock cycles before the trigger the window should start) and length (how many clock cycles after its start the window should end) can be set in the VMM slow control software. The window logic is implemented in a module called `vmm3unit` which contains a counter of clock cycles called `trgcounter`[3] and two more signals `window_start_trgcounter` and `window_end_trgcounter`. The latter two are respectively connected to the result of the operation that subtracts the window start time from the `trgcounter` and the operation that subtracts the start time and then adds the window length[4]. This means that they will always be updated relative to the current value of `trgcounter`, analogously to the first example in the previous section (2.3.1). There is a process that saves the *current* values of `window_start_trgcounter` and `window_end_trgcounter` in two other signals when a trigger arrives. These two signals are then compared to the arrival time of each hit; if the hit arrived between the start and end of the window, it is added to the VMM data stream.

The VMM data stream consists of *hits* and *markers*, each 48 bits long. A hit contains the actual information measured by a channel of one of the VMMs while markers provide additional required information

[3] This is not to be confused with the event counter called `trigger_counter` in chapter 3. The two signals are related only by name.

[4] This explanation leaves out complications caused by the fact that these signals overflow frequently. It is only meant to give an understanding of the general principle behind how the window works.

not contained in the hits. There are three different kinds of markers in the triggered mode. The FEC sends the following sequence when it receives a trigger:

1. Relative marker: Contains Additional time information required to calculate the correct absolute time of hits.
2. Hits: Contain the ID of the VMM that registered the hit, the channel number, an ADC value of the signal peak, and time information relative to the trigger.
3. Trigger marker: Contains the time (in clock cycles) of the trigger
4. End marker: Contains no additional information but marks the end of the event.

The absolute time of the peak of each hit can then be calculated in the event reconstruction (see Sec. 5.1.1) using the information from the trigger marker, relative marker and the hit itself.

All markers and hits contain a bit called *data flag* which is always `1` for hits and always `0` for markers. Another 5 bits are used for the VMM ID, a number associated with one of the VMMs, of which there are at most 16 per FEC. For hits, the VMM ID in the data identifies the VMM that registered the hit. Markers are simply sent for all connected VMMs and each of them contains its associated VMM ID. The remaining $42 = 48 - 1 - 5$ bits can be different for hits and each kind of marker. For the trigger marker all 42 bits are used to send the time of the trigger in clock cycles and for the end marker they always contain only ones. The trigger marker is visualized in Fig. 2.5.

Figure 2.5.: The 48 bits of the trigger marker with colour coding. Each square represents one bit. The 5 bits containing the VMM ID are shown in blue and the data flag, which is always `0` for a marker, in green. The rest of the marker is taken up by 42 bits containing the trigger time. The numbers in the brackets show the order of the bits from left to right. The leftmost bit of the marker is the most significant bit (41) and the rightmost bit is the least significant bit (0).

Since the data flag only differentiates between data that is a marker and data that is not, additional information is required to differentiate between the different kinds of markers. The first bit of the VMM ID is used to encode this information, since the 5 bits allow for 32 different IDs while only 16 are needed to identify the maximum number of VMMs. Markers with VMM IDs between 0 and 15 are the relative markers, whereas markers with VMM IDs between 16 and 31 (i.e. with the first bit of the ID set to `1`) are trigger markers or end markers. The latter two can be differentiated by their content in the remaining 42 bits.

3. Event Counter Implementation

The goal of this thesis is to implement event counter functionality in the SRS using the VMM3a front-end. Since the event counter is intended to identify the same event across multiple hybrids and even readouts, it should be implemented at the most unified point in the readout chain possible. This is the FEC, or when using multiple FECs, the CTF. The CTF does not contain an FPGA or other suitable reprogrammable part which means that no new functionality can be added without hardware changes. The functionality of the FEC on the other hand is governed by the firmware contained on its FPGA and it is therefore a good candidate for adding a new feature such as this. Implementing the event counter is then a matter of changes to the FEC's firmware. These will be further described in this chapter.

3.1. General Design Choices

The central idea is to generate the event counter on the FEC, add it to the data stream sent to the computer and also output it to be read in by a different readout. This way both the SRS readout and the other readout save the counter, which can then be used in the offline[5] analysis to match data from the two sources that belongs to the same event. Therefore the main necessary firmware additions are the following:

1. A counter that is increased by 1 whenever a trigger signal is received.
2. Writing this counter to the data sent by the FEC over Ethernet when the trigger signal is received.
3. Outputting the counter to be read by another readout when the trigger signal is received.

As the physical port for output of the counter (point 3. above), the FEC's NIM output was chosen. The port itself is already defined in the firmware and it offers the simplicity of connecting a LEMO cable e.g. to an oscilloscope to read in the counter. Serially outputting the event counter, i.e. sending one bit after another, is the only option for a connector such as LEMO with only a single signal line. Serializing the counter has the additional advantage that it can be read using only a single channel on an oscilloscope or other measurement device. The previous APV25 event counter implementation also serially outputs the counter from the NIM output.

The APV25 implementation is used to synchronize a beam telescope with Picosec [20] detectors. To be able to do the same using the VMM3a implementation, the serialized event counter needs to adhere to the requirements of the readout used. This is either an oscilloscope or a SAMPIC [21, 22] readout chip. The software used for the latter contains a feature developed for the readout of the APV25 Event Counter which uses a 40 MHz (25 ns) clock. It has 16 bits for the counter itself, padded with a `1` at the start and `00` at the end. The VMM3a firmware on the other hand uses a $44.\overline{4}$ MHz (22.5 ns) clock. For serial output at 40 MHz, the PLL was used to generate an additional clock signal at this frequency.

Since the event counter does not only have to be output serially but also in the VMM data stream, it needs to exist in both the 40 MHz and the $44.\overline{4}$ MHz clock domains at some point. There are three main options to achieve this:

1. Receive the trigger signal and generate the counter separately in both clock domains.

[5] *Offline* in this context means after the measurement itself.

2. Generate the counter in the $44.\overline{4}$ MHz domain and transfer it to the 40 MHz domain.

3. Generate the counter in the 40 MHz domain and transfer it to the $44.\overline{4}$ MHz domain.

Option 1 has the advantage of requiring no transfer of the counter between clock domains, however it was ruled out because of the possibility of the counters becoming mismatched. A trigger signal may be missed in specific cases due to the trigger being shorter than one clock cycle or due to inaccuracies in the signal caused e.g. by the electronics of the NIM input port. Since the two clocks are generally out of phase, one of them might miss the trigger while the other does not, which would cause a mismatch between counter values in the two domains for all values after that point. Option 3 was then chosen over option 2 because it is simpler to cross from a slower to a faster clock (see Sec. 2.2.3) and because the event counter serial output is more time critical than adding it to the VMM data stream. This is because the serial event counter should ideally arrive at roughly the same time as the signal waveform whereas the VMM data can be buffered over several clock cycles before being sent.

The event counter implementation was added to `vmm3unit` since changes need to be made to the VMM data stream and the module already contains functionality to receive the trigger. A new module called `trigger_counter` was defined and instanced within `vmm3unit` and most event counter-relevant logic added to it. The 40 MHz clock signal was propagated from the PLL module into an additional input of `vmm3unit` and from there into `trigger_counter`. The pre-existing process for trigger synchronization and shortening (see Sec. 2.3.2) was moved into the new module. Since the trigger also needs to be available in the 40 MHz clock domain, another process which works in the same way but uses the 40 MHz clock was added. It produces a signal `trgin_sync_40` which is only one 40 MHz clock cycle in length. The `trigger_counter` module contains a new 16 bit signal `trigger_counter_40` representing the event counter which is generated in the 40 MHz clock domain. Its value is initialized to 0 and increased for every rising 40 MHz clock edge where `trgin_sync_40` is `1`, subject to an additional condition described in the following section (3.2).

3.2. Serial Event Counter

The `trigger_counter` module contains another signal called `tc_shiftreg` which is used as a shift register[6]. The current value of `trigger_counter_40`, padded with a `1` as the leftmost bit and `00` as the two rightmost bits, is copied into it when a trigger signal is received. The leading one and trailing zeros are added to conform to the format of the APV25 counter described above (Sec. 3.1). Each 40 MHz clock cycle, the register is shifted left with a `0` filling the rightmost bit. The leftmost bit of the register is connected to the NIM output, so the current event counter is output serially, always starting with a `1` followed by the most significant bit and ending with the least significant bit followed by `00`. This takes $19 = 1 + 16 + 2$ clock cycles. Since the counter is written into the shift register in the same clock cycle in which it is increased, the serial output contains the counter value as it was before the increase. This means that the very first event counter on the serial output is `0`.

If a trigger signal starts the serial output of the event counter and a second trigger arrives before the 19 clock cycles have passed, the counter will be increased again and the serial output restarted before the previous output has finished. This can lead to counter values being interpreted incorrectly by the readout to which the counter is sent and is unwanted behaviour. Thus a signal `tc_shiftreg_status` was added containing a counter that is increased from 0 every clock cycle after the start of the serial output, and when this signal reaches the value of 19 another signal `tc_serial_done` is set to `1`. Another increase and output of the event counter is only performed when `tc_serial_done` is `1`. The code for this is shown in the appendix, Fig. A.1.

[6] `tc` is used as an acronym for `trigger_counter` in longer signal names.

The theoretical shortest possible time interval between two consecutive serial counter outputs is 19 clock cycles if the first serial output is not to be cut off by the second. After being reset to 0, `tc_shiftreg_status` takes 20 clock cycles to reach the value of 19. `tc_serial_done` will then be set to 1 and in the next clock cycle another serial output can be started. This means that the shortest possible time interval allowed by the firmware is 20 clock cycles which corresponds to a buffer of 1 clock cycle between consecutive serial outputs.

For the APV25 event counter implementation, the maximum event rate is not limited by the length of the serialized counter but by the capabilities of the readout system. The APV25 itself is designed for trigger rates of up to 100 kHz [23] which sets the upper bound for rates of readout systems using this ASIC. The actual maximum rate may be lower than this due to other steps in the APV25/SRS readout chain. In the case of VMM3a/SRS, the rate capability is on the order of several MHz [19] and therefore the event counter serial output becomes relevant as a factor limiting the rate. With the 20 clock cycle interval, the maximum rate of event counters that can be output is

$$\frac{40\,\text{MHz}}{20} = 2\,\text{MHz}$$

which is higher by a factor of 20 than the upper bound for the rate of the APV25 readout.

3.3. Event Counter in VMM Data

Everything regarding the VMM data stream happens in the $44.\overline{4}$ MHz clock domain used by the majority of the firmware. Therefore, to be able to add the event counter to the VMM data stream, it has to be transferred from the 40 MHz to the $44.\overline{4}$ MHz clock domain. Since the two clocks are generated by the same PLL and have periods of 25 ns and 22.5 ns the worst-case (shortest possible) time difference between their rising edges is relatively short at 2.5 ns. This is because when both clocks start at the same time in the beginning, the second clock cycle will start at 22.5 ns for the faster and at 25 ns for the slower clock, resulting in a difference of 2.5 ns.

To avoid timing issues when transferring the counter from the slower to the faster clock, use was made of the fact that the event counter is constant for several 40 MHz clock cycles as long as the serial output has not finished. Within this time period, the counter's value can be safely transferred to the $44.\overline{4}$ MHz clock domain. To know when the event counter was updated, `tc_serial_done` is written into a 3 bit $44.\overline{4}$ MHz shift register to compare its current and previous value. If the value has changed from 0 to 1 the event counter was updated and the signal `trigger_counter_updated_44` is set to `1` for one $44.\overline{4}$ MHz clock cycle. The 3 bit shift register also serves to double-flop the value of `tc_serial_done`, making the transfer from 40 MHz to $44.\overline{4}$ MHz safe in terms of timing. The code for this is shown in the appendix, fig A.2.

In the new event counter firmware, hits should only be sent when there is a new unique event counter value to associate with them. This does not necessarily happen for every trigger signal, since the firmware waits for the serial output to finish before allowing updates. Since updates of the event counter happen in the 40MHz clock domain however, information on when the counter was updated (in the signal `trigger_counter_updated_44`) has a variable delay relative to the trigger due to the variable phase between the clocks. Updating the window (see Sec. 2.3.2) on this basis would thus add an unwanted inaccuracy. Instead the window is still updated based on the trigger signal but the actual sending of hit data, which is less time critical, is only started when the counter was actually updated (i.e. when `trigger_counter_updated_44` is `1`). If a trigger arrives but does not update the counter, the window will be updated but the corresponding data never sent. A later trigger that does update the counter will again update the window before the new data is sent.

In order not to unnecessarily constrain the bandwidth of the data stream, no new marker type was added to but the existing end marker replaced by the event counter marker. This is possible since

the end marker contains no event-specific information other than the VMM ID. The new event counter marker always starts with `1111` as the first four bits and uses the last 16 available bits to send the event counter. The remaining bits are `0`. This is shown in Fig. 3.1. Setting the first four bits to ones enables

Figure 3.1.: The 48 bits of the event counter marker with colour coding as in Fig. 2.5. The 5 bits containing the VMM ID are shown in blue and the data flag, which is always `0` for a marker, in green. The bits containing the event counter (*TC* standing for "trigger counter") are displayed in red, with the numbers in the brackets showing the order of the bits from left to right. Bit 15 is the most significant and 0 the least significant bit of the event counter. The rest of the bits (in gray) always contain the same pattern.

differentiation between the trigger marker and event counter marker. The trigger marker, which uses 42 bits to encode the number of $44.\overline{4}\,\text{MHz}$ clock cycles (see Fig. 2.5), will only start with `1111` after

$$(2^{41} + 2^{40} + 2^{39} + 2^{38}) \cdot 22.5\,\text{ns} \approx 4.12 \times 10^{12} \cdot 22.5\,\text{ns} \approx 1.07\,\text{days}$$

which means that it will take this long after beginning data acquisition before the trigger marker and event counter marker are not clearly distinguishable anymore. More than one day is enough for most ordinary data taking runs.

4. Test Setup

The main purpose of the event counter is the ability to synchronize multiple detectors measuring the same particles. However, its functioning can also be tested by using only a single detector and reading it out in different ways. For this purpose, a small setup using a MicroMegas detector was assembled. This setup will be described in detail in this section.

4.1. Overview

The MicroMegas detector described in section 1.1.2 was used with a 70:30 Ar/CO_2 mixture. Two VMM hybrids each are connected to plane 0 and plane 1. The hybrids are connected, through HDMI cables and the DVMM adapter card, to a FEC running the event counter firmware. This corresponds to the architecture shown in Fig. 2.1. A picture of the detector is shown in Fig. 4.1.

The second readout uses a preamplifier (ORTEC 142 model) connected to the detector's mesh. The preamplifier has additional connections besides its in- and output, one of them being a bias input which enables setting the base voltage on the preamplifier's input. A $50\,\Omega$ termination resistor is connected to the bias input which leads to the detector's mesh being grounded through a $50\,\Omega$ resistance. A voltage of $-660\,\mathrm{V}$ is applied to the DLC resistive layer, and a voltage of $45\,\mathrm{V}$ to the cathode. To take measurements, an ^{55}Fe source is placed above the detector.

4.2. Readout

See Fig. 4.2 for an overview of the connections in the test setup's readout. The signal from the preamplifier goes into a timing amplifier and is then routed through multiple NIM modules. Besides going directly into the oscilloscope, the amplifier's output is connected to the input of a discriminator module which produces a short pulse whenever the signal passes above a certain threshold value. This trigger is then provided to the NIM input of the FEC. The event counter produced by the FEC then goes from its NIM output to another channel on the oscilloscope. The oscilloscope is set to trigger on a negative edge of this channel. Thus it triggers on the first edge of the first `1` bit which is part of the output for any event counter value (see Sec. 3.2). An example waveform is shown in Fig. 4.3.

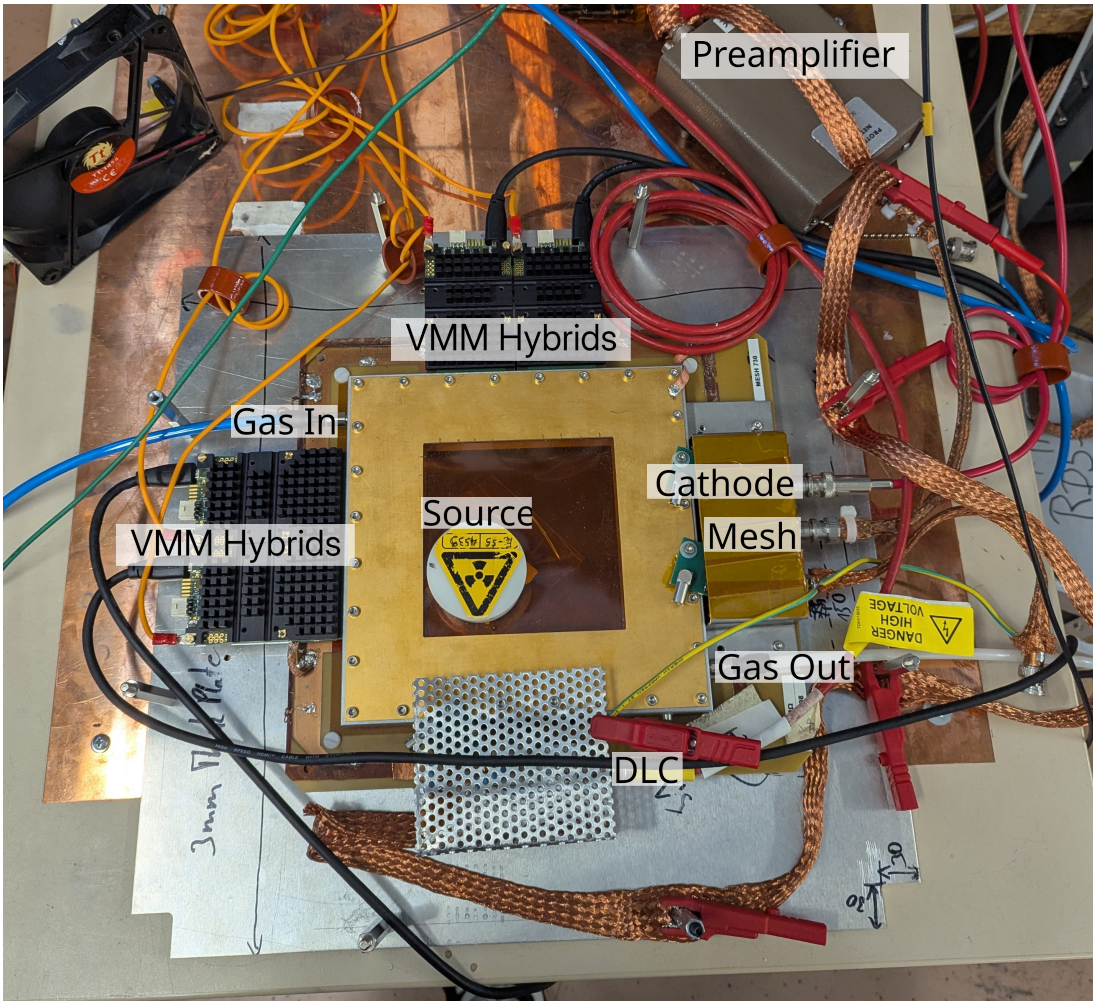


Figure 4.1.: Photograph of the detector as used in the test setup, with labels identifying the preamplifier, the ^{55}Fe source and the VMM hybrids. There are also labels for electrical and gas connections to the detector. Two hybrids each are connected to the x and y strips. The black cables connected to them are the HDMI cables going to the FEC and the orange cables are used for grounding.

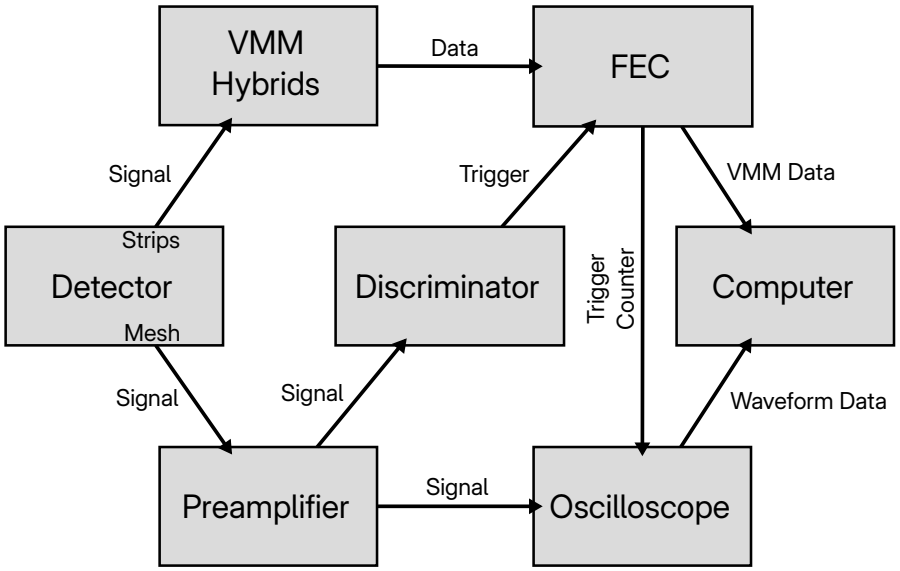


Figure 4.2.: Flow chart of the setup used to test the event counter. "Signal" stands for the waveform caused by charged particles in the detector. The signal from the detector's readout strips goes to the VMMs and the signal from the mesh goes to the preamplifier. "Preamplifier" stands for the combination of preamplifier and timing amplifier that amplify and shape the signal.

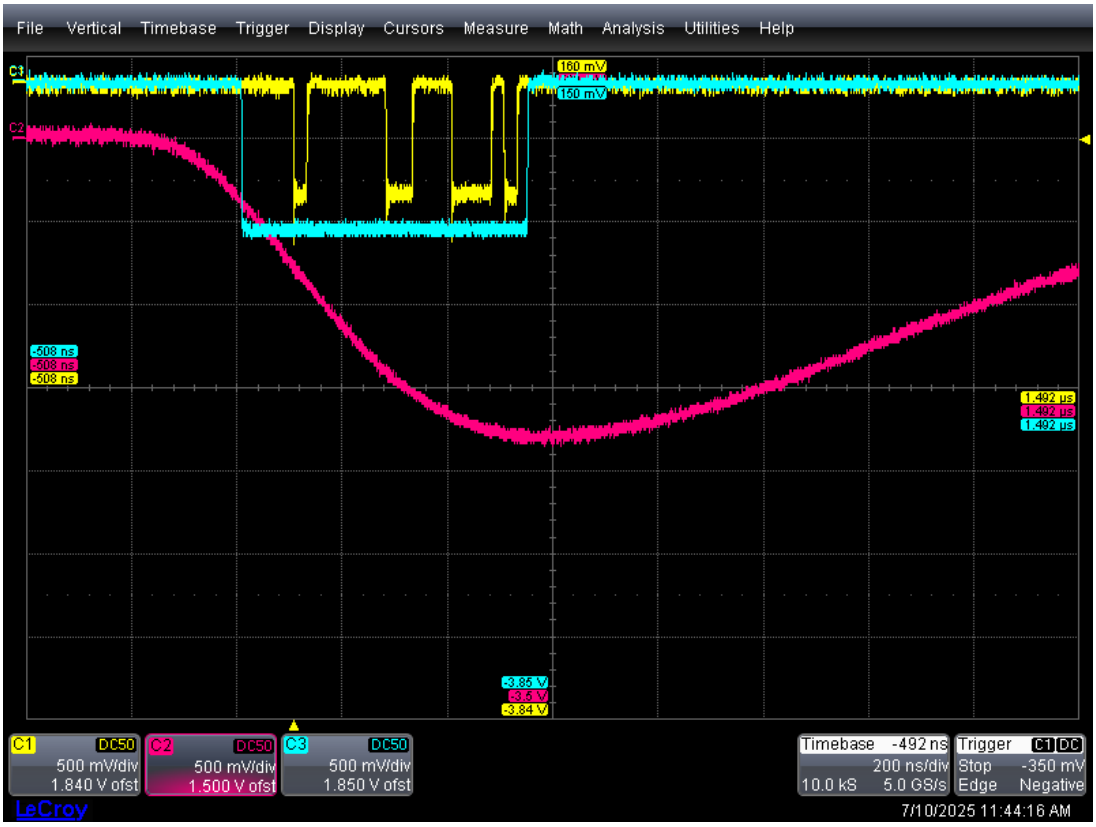


Figure 4.3.: Typical waveform seen on the oscilloscope used to read out the test setup. The oscilloscope is set to trigger on channel 1 (yellow) containing the event counter. The counter uses NIM logic levels, meaning that a `0` is 0 V and a `1` corresponds to a negative voltage level [14]. The first `1` that can be seen is not part of the counter and the last `1` is the least significant bit. The current value is `0000 0011 0001 1101` or 797. The amplifier's signal is connected to channel 2 (pink) and the trigger signal to channel 3 (light blue). the trigger signal is not required on the oscilloscope but was usually connected for reference.

5. Results and Analysis

The test setup provides two data streams:

1. The VMM data in the form of `pcapng` files containing the raw network packages sent by the FEC.
2. The waveforms of all active oscilloscope channels saved in a binary data format specific to the vendor (Teledyne LeCroy).

This chapter describes the steps taken to process these and combine them to analyze whether the event counter functions as intended.

5.1. Initial Data Processing

5.1.1. VMM Data Stream

The `pcapng` files from the VMM data stream contain only markers and hits but no higher-level information. The process of extracting event-related information from the data is called event reconstruction. For VMM3a/SRS data, this is performed by a software called *VMM3a/SRS Data Analysis Tool* (vmm-sdat) [24]. It merges data from multiple hits into *clusters* which are more directly usable to get information about the detected particle, such as its energy.

vmm-sdat first generates clusters for each plane separately by combining hits that are close to each other in time and space (e.g. coming from neighbouring strips). The time and position of each cluster is calculated based on the times and positions of the hits it is made up of. These one-dimensional, plane-specific clusters are then combined into two-dimensional clusters, again based on their time and position. The parameters governing this algorithm can be set by the user. The resulting data is saved in a ROOT[7] file which contains data for both the one-dimensional and the combined two-dimensional clusters. For example, this information includes the ADC count summed over all hits in the cluster which will be called *VMM amplitude* or simply *ADC count* in the following.

For the analysis of data using the triggered readout mode and containing event counter information, vmm-sdat was updated by Dorothea Pfeiffer. The new version adds the event counter of each cluster to the ROOT data. From the ROOT file generated by vmm-sdat from the test setup data, a list of event counter values and corresponding VMM amplitudes was extracted.

5.1.2. Oscilloscope Data

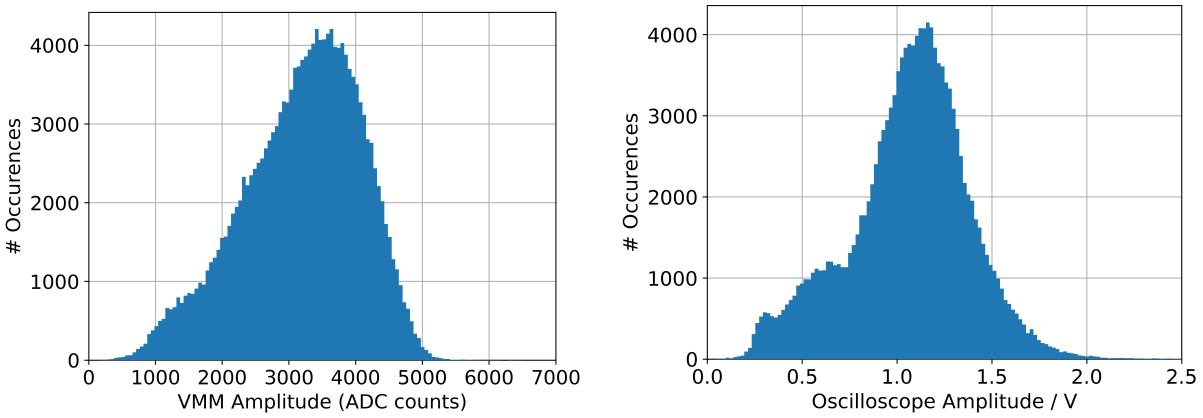
The binary format of the oscilloscope waveform files was read using the Python library `lecroyparser`[8] which provides the oscilloscope waveforms as simple arrays of floating-point numbers, one array for each channel and one for the time data.

The channel containing binary counter values was decoded to get the event counter values. The corresponding signal amplitudes were calculated from the channel containing the signal waveforms as the differences between maximum and minimum values. These amplitudes will be called *oscilloscope amplitudes* in the following. Plots of the the decoded data are shown in the appendix, Fig. A.4.

[7] Data analysis framework for high energy physics developed at CERN [25].
[8] `https://pypi.org/project/lecroyparser/`

5.2. Data From the Individal Readouts

Histograms of the amplitudes from both readouts are shown in Fig. 5.1. The rough shape of the ^{55}Fe spectrum is discernible in the histograms but neither has a clear separation between photopeak and escape peak. This low energy resolution is possibly caused by the preamplifier used in the test setup. The



(a) Data from the VMM readout.

(b) Data from the oscilloscope.

Figure 5.1.: Histograms of amplitudes from the two readouts of the Test Setup. These plots still include events that exist in the oscilloscope data but not in the VMM data, or vice versa. Plotting only events that exist in both data streams, as in section 5.3, does not make significant differences to the shapes of the spectra.

preamplifier had to be replaced by a different model due to unforseeable circumstances a few days before the final measurement was taken and the new preamplifier possibly contained modified electronics. For example, a changed impedance of the mesh to ground would influence the detector itself and may worsen the energy measurement in both the oscilloscope and the VMMs. Energy resolution may additionally have been impacted because there was more electronic noise after replacing the preamplifier, since not much time was available do re-do grounding measures.

Fig. A.3 in the appendix shows data taken before the preamplifier was swapped. Both readouts show spectra with a significantly higher energy resolution. This data set is a combination of two runs, taken with two different older versions of the event counter implementation, and is therefore not meaningful as a test of the final firmware version.

Although the quality of data using the new preamplifier is not ideal, it was considered sufficient for the intended analysis in the following section (5.3).

5.3. Event Matching

To test the proper functioning of the event counter, the two lists of amplitudes were merged by matching VMM amplitudes and oscilloscope amplitudes that correspond to equal event counter values. Visualizing this data in a plot, a positive correlation should be visible between the amplitudes of the two readouts.

A scatter plot of the matched amplitudes (Fig. 5.2) has noticeable vertical lines due to the limited voltage resolution of the saved oscilloscope waveform. This effect was removed from the following plots

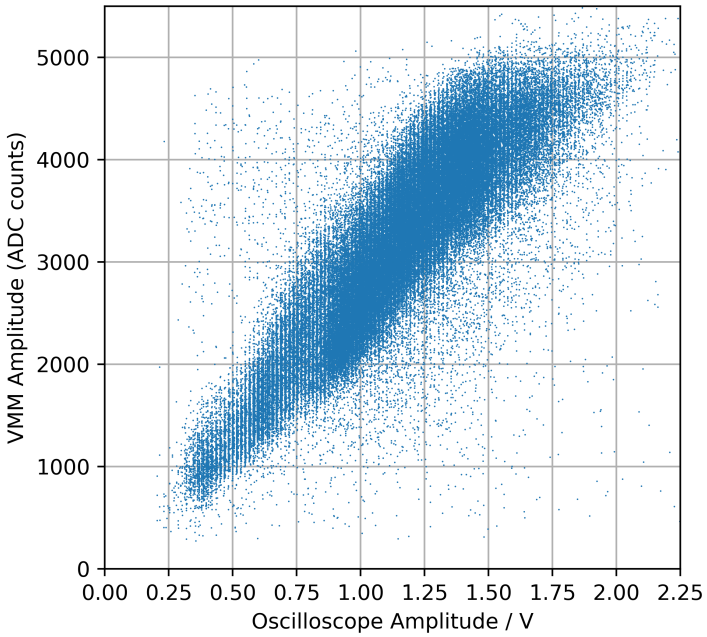


Figure 5.2.: Scatter plot of VMM amplitudes against oscilloscope amplitudes.

by applying a slight smoothing[9] to the signal waveforms before calculating the amplitude. A before-and-after comparison of an example waveform can be found in the appendix, Fig. A.4.

A two-dimensional histogram of matched amplitudes can be seen in Fig. 5.3. Visual inspection shows that the amplitudes are indeed correlated. One can again see the general shape of the ^{55}Fe spectrum, with the photopeak roughly at 1.2 V or an ADC count of 3300 and the escape peak around 0.6 V or 1500. There is a relatively sharp cut-off in the plot above ADC counts of ~ 5000. Another peculiarity is that the general shape of the histogram appears to become wider towards higher counts, as if the energy resolution decreased with increasing energy.

Plotting the one-dimensional clusters of the two planes separately gives insights into these phenomena. Whereas Fig. 5.3 shows data from the combined two-dimensional clusters, plotting data from the one-dimensional clusters shows the differences between the two planes. These histograms are shown in Fig. 5.4. The scale of the VMM amplitudes is about half of that in Fig. 5.3 since the ADC count of each two-dimensional cluster is calculated as the sum of the ADC counts of two one-dimensional clusters. The data from plane 0 in Fig. 5.4 has a much more regular shape than plane 1 though the two planes are generally expected to behave similarly. The fact that there are significant differences between the two indicates that there is misbehaviour in at least one of them. One of the two hybrids on plane 0 could not be properly calibrated by the slow control software and likely did not work correctly. However, the radioactive source was purposefully placed above the strips read out by the other hybrid (see Fig. 4.1). This lead to almost all measured clusters being on the side of the working hybrid, so the faulty hybrid was not the issue here.

The plot of plane 1 is more spread out than plane 0 which can be seen from its general shape and the lower maximum bin count. The photopeak lies around an oscilloscope amplitude of 1.1 V and an ADC count of 1500 for both planes, however in plane 1 it is much more spread out and appears to be

[9] A Savitzky-Golay filter from the SciPy library (`https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol_filter.html`) was used.

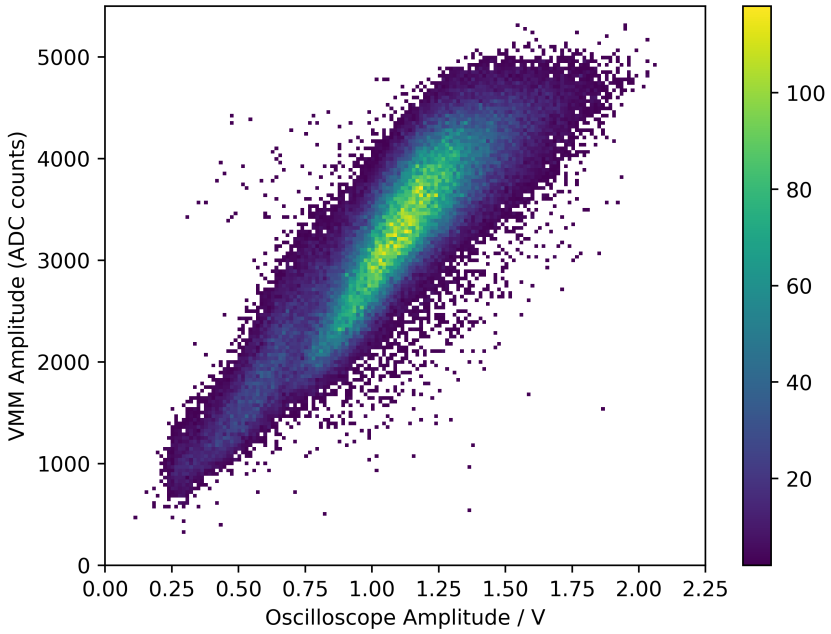


Figure 5.3.: Two-dimensional histogram of VMM amplitudes (sum of ADC counts for both planes) against oscilloscope amplitudes with the event counter matched between oscilloscope and VMM data.

vertically split into two regions, the top one being centered around an ADC count of roughly 2200. Since this splitting has no equivalent in plane 0 it is likely an artifact of some problem in a hybrid or the strips of plane 1. The same applies to the shape in the plane 1 plot that also appears at lower oscilloscope amplitudes but is shifted upwards (towards higher ADC counts) from the main ^{55}Fe spectrum.

Above an ADC count of about 2500, the shape of plane 1 rises significantly more slowly which is likely a saturation effect of the VMM channels: Each channel has a maximum ADC count of 1023, meaning that any charge that would result in a higher count is simply registered as 1023 and the additional charge information is lost. The higher the total charge, the more channels are saturated and the more charge information is lost. This effect does not have an equivalent in the oscilloscope measurement and therefore the ADC counts appear to rise more slowly at high amplitudes. The same effect can be seen in plane 0 at ADC counts around 2400. It is much less pronounced here since the data does not reach ADC counts as high as in plane 1.

Apart from this, plane 0 shows a spreading in its shape when going to higher amplitudes, similarly to Fig. 5.3. At least in part, this might simply be a statistical effect: The photopeak, which has much higher counts per bin, lies at higher energies and it is statistically more likely for a bin farther away from the mean to be filled in a region where there are more counts in general. The cutoff at an ADC count of about 5000 seen in Fig. 5.3 does not appear in either plot of Fig. 5.4 which means that it is likely a chance occurence resulting from the combination of clusters from the two planes.

Looking at data from the two planes seperately also provides another opportunity to inspect which effects may be caused by the new preamplifier. Analogous plots to Fig. 5.4 but with data using the old preamplifier are shown in Fig. 5.5. This data set is much smaller which is why a larger bin size was chosen. The oscilloscope amplitudes are not directly comparable to the new preamplifier since the gain is different.

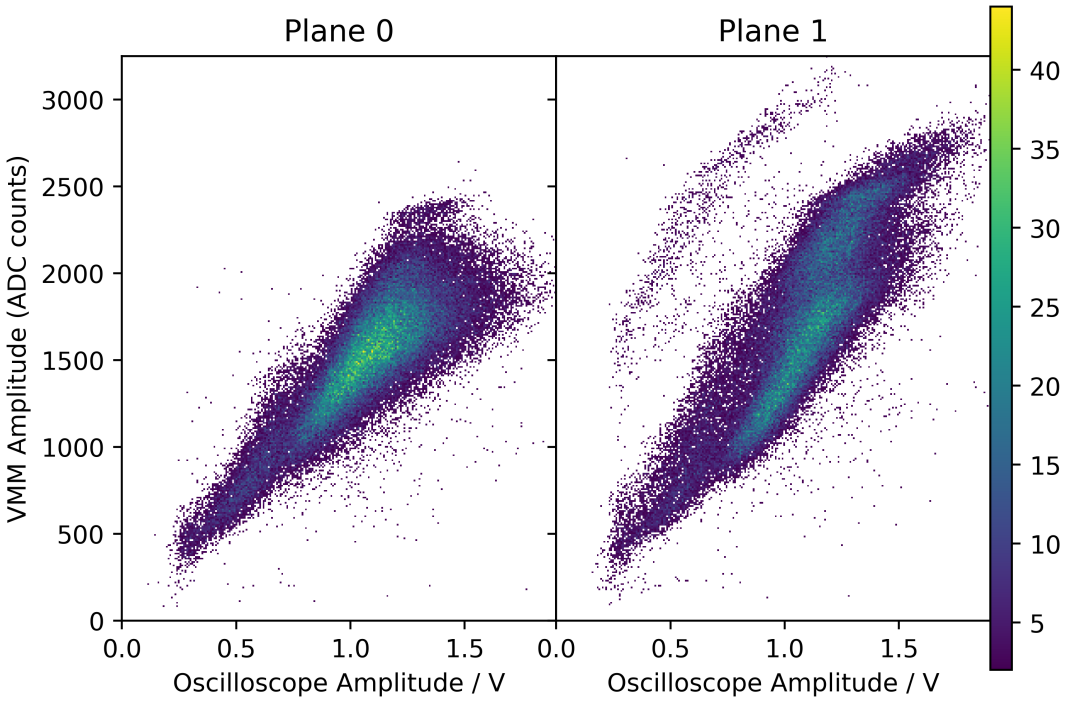


Figure 5.4.: Histograms analogous to Fig. 5.3 but with the two planes plotted separately.

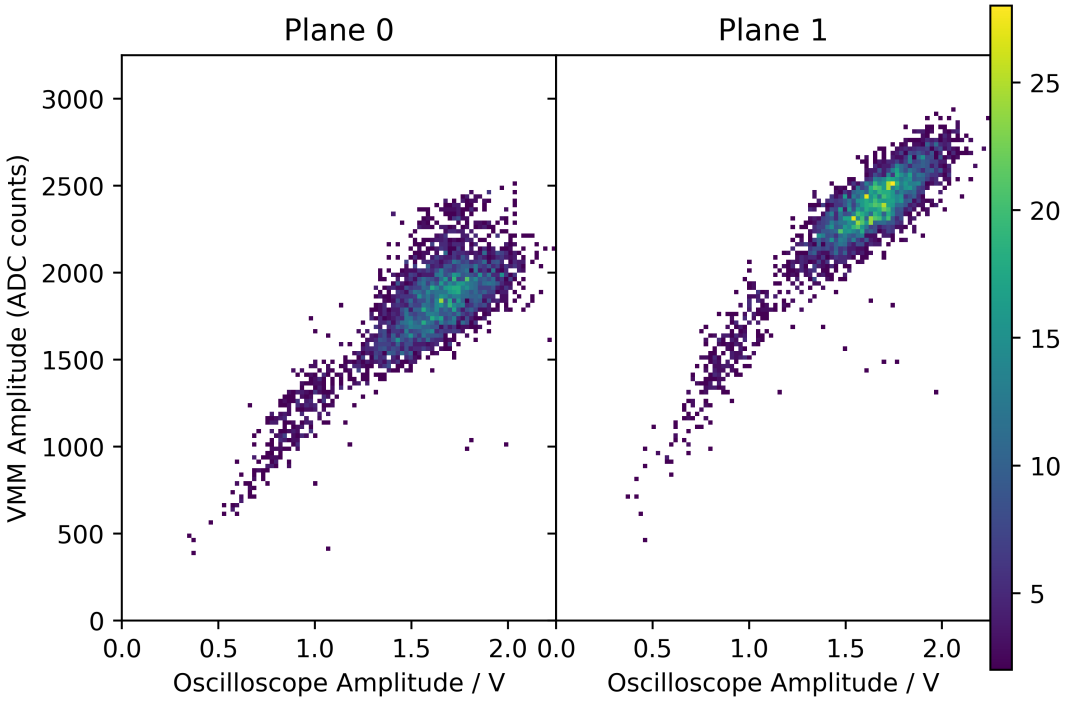


Figure 5.5.: Histograms analogous to Fig. 5.4 with data using the old preamplifier.

Plane 0 appears similar to the corresponding plot in Fig. 5.4, but with the old preamplifier the center of the photopeak is slightly shifted towards higher ADC counts. It is at around 1900 whereas for the new preamplifier the photopeak is centered roughly at 1500. Though there is not enough data for a precise comparison, this suggests that the new preamplifier may have had an effect on the gas gain of the detector.

The data from plane 1 using the new preamplifier (Fig. 5.4) does not have a well discernible shape and thus no clear comparisons can be made here. However, with the old preamplifier the photopeak does not appear to be split as with the new one. Only the upper part of the photopeak exists in Fig. 5.5 and it is again significantly higher than the photopeak seen in plane 0. This suggests that the hybrids of the two planes have different gains or that the strips of the two planes receive different amounts of signal induction. The VMM amplitude around the photopeak rises more slowly than around the escape peak. This can again be attributed to saturation effects. The same applies for plane 0 though again the effect is less visible there.

Another way to visualize the matching of events from the two readouts is to purposefully misalign the list of oscilloscope amplitudes with the list of VMM amplitudes, e.g. by shifting the latter by one element. This should result in visibly less correlation than before. Such a plot is shown in Fig. 5.6, which

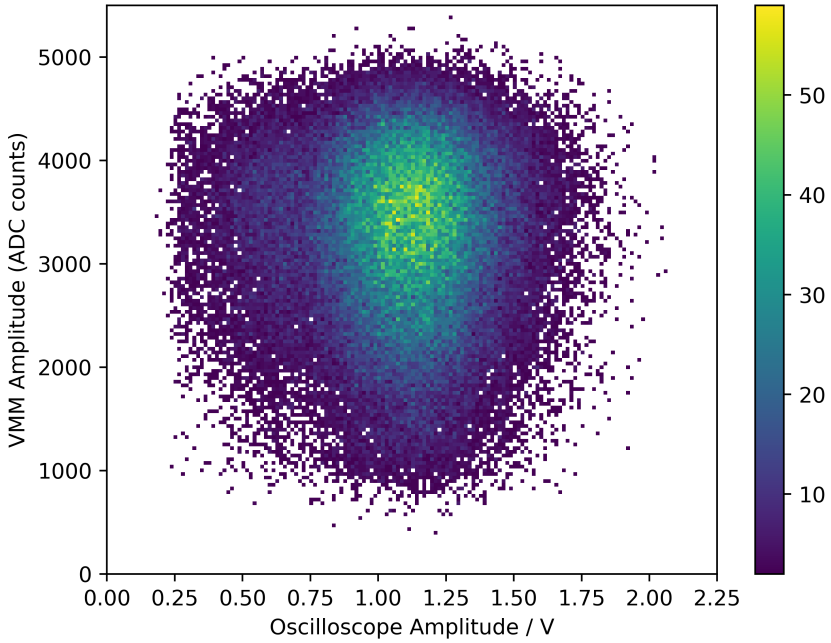


Figure 5.6.: Two-dimensional histogram of VMM amplitudes against oscilloscope amplitudes. A mismatch between oscilloscope and VMM data was introduced by matching the data based on the event counter and then shifting the list of VMM amplitudes by one element.

again uses the combined two-dimensional clusters of the data from the new preamplifier. The resulting shape is roughly circular, indicating no correlation. One can see that it is most likely for two unrelated amplitudes from the two different readouts to both lie around the photopeak, less likely for only one of them to be around the photopeak and even less likely for both to be around the escape peak. The plot is also generally more spread out, as can be seen from the colour scale: In the correctly aligned plot the maximum count is around 120 whereas for the misaligned data it is around 60.

5.4. Results

The goal of the event counter is the ability to reliably and easily match corresponding events from two (ore more) different readouts. Generating the event counter in one readout (the VMM3a/SRS) and sending it out to be read by the other data acquisition system (the oscilloscope) has the advantage that there is no risk of permanent asynchronicity, since the counter is centrally generated. As a counter example, event matching based on a separate event counter in each readout would result in a permanent mismatch if one of the counters missed a trigger. Such data might look similar to Fig. 5.6, i.e. the amplitudes are fully uncorrelated (after the point when a trigger was missed). The fact that the correctly matched data from the test setup (Fig. 5.3) looks qualitatively different to this and shows a visible correlation demonstrates that the event counter implemented in this thesis can be reliably used to synchronize two readouts.

6. Outlook

6.1. Test Beam

The results from chapter 5 show that the event counter can be successfully used to synchronize multiple readouts. The event counter using the APV25 frontend that is replicated by this new event counter implementation has its main use in event synchronization between tracking detectors and a device under test (DUT) in a test beam environment. Thus it should be insightful to also test the adequacy of the new implementation in such an environment. This was done during the DRD1 collaboration's test beam in July 2025. Multiple beam telescopes were set up in the beam line, one of which was read out using the APV25 front-end and another using the VMM3a front-end. The new event counter firmware was used for the FECs of the VMM3a beam telescope, to which the tracking detectors as well as the DUTs were connected. Triggers were generated by multiple scintillation counters connected to a coincidence unit which only outputs a trigger when all of the scintillators produce a signal within a certain time frame.

The test beam took place in CERN's North Area at the H4 beam line which is fed by the Super Proton Synchrotron (SPS). The beam of the SPS hits a target which then releases the particles that go to the beam line. This happens in *spills* of 4.8 s, in between which there is no particle beam [26]. This pattern can be observed when plotting the event counter against the time of events, see Fig. 6.1. The spills at

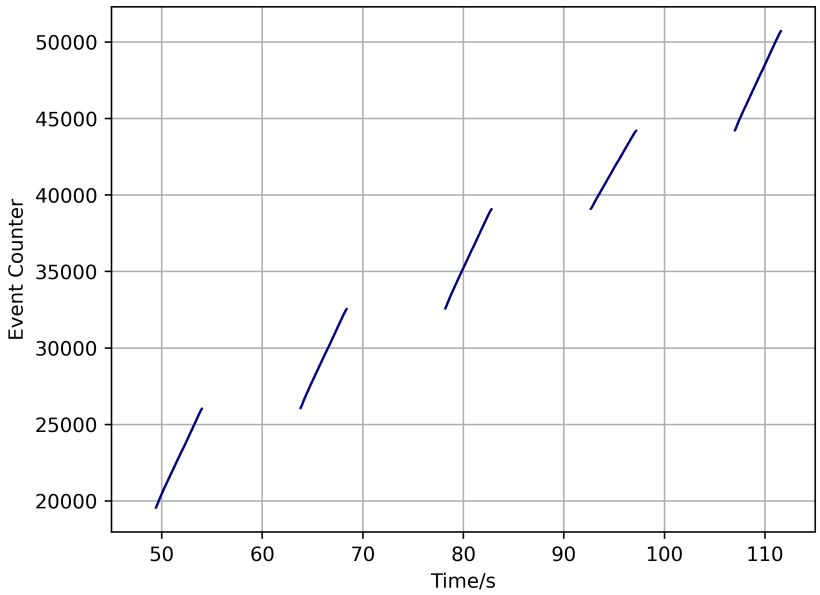


Figure 6.1.: Plot of event counter values against the time of events, produced from data of one of the tracking detectors in the test beam.

regular intervals are visible as rising lines with gaps of no events in between. The length of spills is consistent with the expected $\sim 5\,\mathrm{s}$. The slope of the lines corresponds to the rate of events. This was effectively limited to be on the order of 1 kHz by passing the trigger signal through a timer NIM module that accepts new signals only after a certain time has passed. The slope in the plot is also roughly consistent with this expected rate.

6.2. Event Counter Input

Of the two beam telescopes using different front-ends, both are now able to output an event counter. Also adding the option of reading in an event counter from the outside would enable synchronization of both beam telescopes with each other, additionally to the synchronization between one beam telescope and DUT. This would be a significant addition to the data available per event to reconstruct the trajectory of the particle beam and against which to check the functionality of the DUT.

An implementation of such an event counter input feature in the VMM3a firmware for the FEC was attempted before the test beam but it could not be completely finished in time. It appeared to work correctly in simulations of the VHDL code but testing it in practice, the decoded values were incorrect. This is possibly because the counter was being read from the `trgin` signal, i.e. the NIM input port since no CTF was connected. The electronics of this port, which need to convert from the NIM logic levels to those suitable for the FPGA, are not designed for high-rate signals such as the 40 MHz serial event counter. A port supporting higher-rate signals is the J12 connector on the FEC. The existing event counter input implementation should be easily adaptable to receive the serial event counter from this connector instead of the NIM input.

The data presented in chapter 5 shows that *most* events are correctly matched based on the event counter, but it does not necessarily prove that this is the the case for *all* events, since the analysis is limited by the energy resolution and other imperfections of the detector and readouts. The event counter input would enable an even more precise test of the synchronizing functionality of both event counter input and output: A setup with two FECs could be used, one of which generates an event counter and sends it to the other to be read in. If the clocks of both FECs are synchronized using a CTF, events from the two different VMM data streams could be matched once based on time and once based on event counter values. If both event counter output and input worked correctly, both methods would result in the same matching of events.

6.3. Other Possibilities

The functionality of the event counter was demonstrated in chapter 5, though no measurements were taken at high rate. To investigate if the readout can truly reach the theoretical rate limit of 2 MHz, for example an X-Ray source could be used [19].

The `tc-shiftreg-status` signal limiting the rate could also be used in scenarios where the trigger rate needs to be purposefully lowered. This would require the addition of a configuration option in the VMM slow control software to change the time after which the serial output is considered to be done from 19 clock cycles to some higher value.

Conclusion

In the course of this thesis, the back-end firmware for the VMM3a/SRS readout system was modified to include an event counter which provides a robust method of synchronizing different readout schemes with a VMM3a readout. This implementation sought to replicate the event counter of the APV25 front-end and replicated its characteristics. This provided the challenge that the VMM3a firmware uses a different clock speed than the APV25 event counter. The counter was successfully implemented using this clock speed different from the rest of the firmware. The counter is serially output as well as added to the VMM data stream.

To test the functionality of the event counter, a test setup with two different readouts connected to the same detector was built. The signal amplitudes measured by the two different readouts, one of which was the VMM3a front-end, were correlated to visualize whether events could be correctly matched. Despite limited energy resolution, a positive correlation between amplitudes was observed, which lead to the conclusion that the events were indeed correctly matched. This showed that the VMM3a front-end can be used to synchronize two different readouts. Compared to using the APV25 front-end for event synchronization it can enable a significant speedup of a factor on the order of 10 due to the higher rate capability of the VMM3a ASIC.

The event counter firmware was also used in a test beam environment, though no final results from this were available yet during this thesis. Some preliminary data shows the event counter working as expected, with the regular spills of particles into the beam line showing up as increases of the counter at regular time intervals.

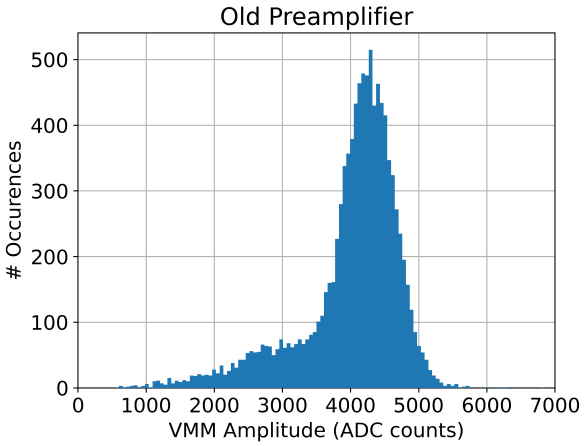
A. Code Extracts and Additional Plots

```
1 -- event counting and serial output of the event counter
2 process(clk_really_40)
3 begin
4     -- do the following for every rising edge of the 40MHz clock
5     if rising_edge(clk_really_40) then
6         -- shift the shift register
7         tc_shiftreg <=
8             tc_shiftreg(tc_shiftreg'left-1 downto 0) & '0';
9         -- increase the counter containing the status of the serial output
10        tc_shiftreg_status <= tc_shiftreg_status + 1;
11
12        -- if there is a trigger and serial output has finished, do the following
13        if trg_sync_40 = '1' and tc_serial_done = '1' then
14            -- reset the counter containing the status of the serial output
15            tc_shiftreg_status <= 0;
16            -- serial output is not done
17            tc_serial_done <= '0';
18            -- start the serial output
19            tc_shiftreg <= "1" & trigger_counter_40 & "00";
20            -- increase the event counter by 1
21            trigger_counter_40 <= trigger_counter_40 + 1;
22        end if;
23
24        -- detect when the serial output has finished
25        if tc_shiftreg_status = 19 then
26            tc_serial_done <= '1';
27        end if;
28    end if;
29 end process;
30
31 -- tc_serial_40 contains the leftmost bit of tc_shiftreg
32 tc_serial_40 <= tc_shiftreg(tc_shiftreg'left);
```

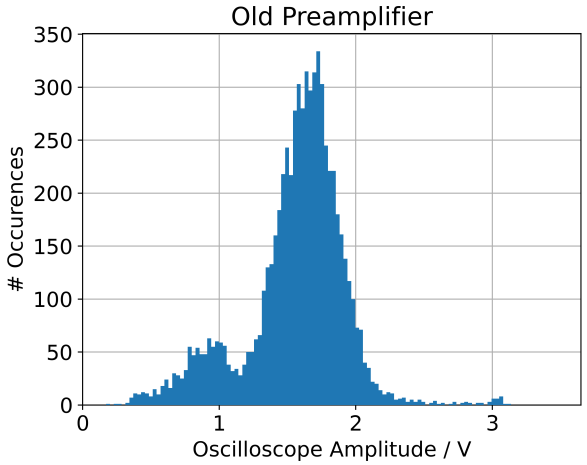
Figure A.1.: VHDL code (slightly simplified) to increase the event counter and start the serial output when a trigger arrives and the previous serial output has finished. The functionality of the code is described in comments, starting with `--`. The 40 MHz clock is called `clk_really_40` to differentiate it from the $44.\overline{4}$ MHz clock which is called simply `clk40` in the `vmm3unit` module and many other parts of the firmware. The serialized event counter will be available in `tc_serial_40` to which the `trgout` output of `vmm3unit` can then be connected.

```vhdl
-- synchronize event counter to 44.44MHz clock
process(clk44)
begin
    -- do the following for every rising edge of the 44.44MHz clock
    if rising_edge(clk44) then
        -- shift the shift register
        tc_serial_done_shiftreg <=
          tc_serial_done_shiftreg(tc_serial_done_shiftreg'left-1 downto 0) &
    tc_serial_done;

        -- if the event counter was updated,
        -- copy its value (subtracted by 1) to the 44.44Mhz clock domain
        if trigger_counter_updated_44 = '1' then
            previous_trigger_counter_44 <= trigger_counter_40 - 1;
        end if;
    end if;
end process;

-- connect trigger_counter_updated_44 to the left two bits of
-- tc_serial_done_shiftreg to detect when the event counter was updated
trigger_counter_updated_44 <=
    tc_serial_done_shiftreg(tc_serial_done_shiftreg'left) and not
    tc_serial_done_shiftreg(tc_serial_done_shiftreg'left-1);
```

Figure A.2.: VHDL code (slightly simplified) that transfers the event counter from the 40 MHz to the $44.\overline{4}$ MHz clock domain. Signals whose names end with `44` belong to the $44.\overline{4}$ MHz clock domain. The code subtracts 1 from the counter in order to match the counter value to that of the serial output, since the serial output always uses the counter value from before the last increase.

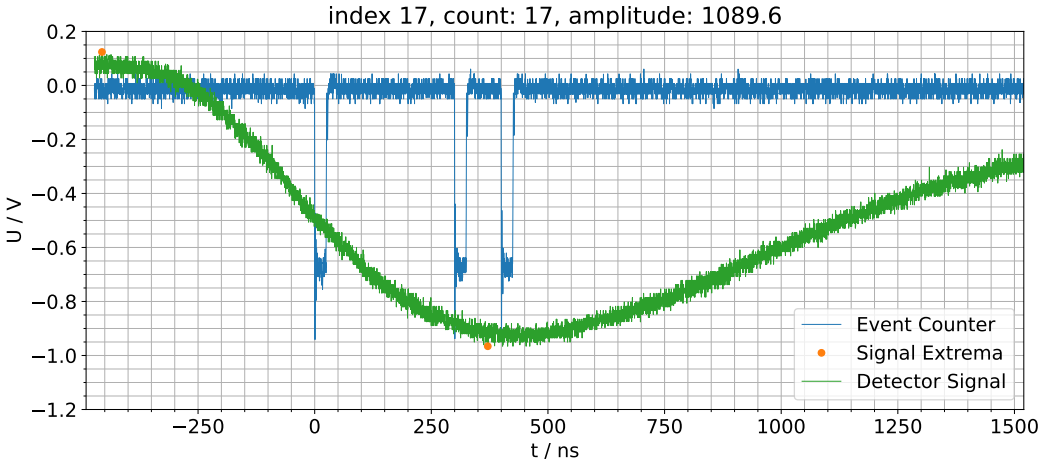


(a) Data from the VMM readout.

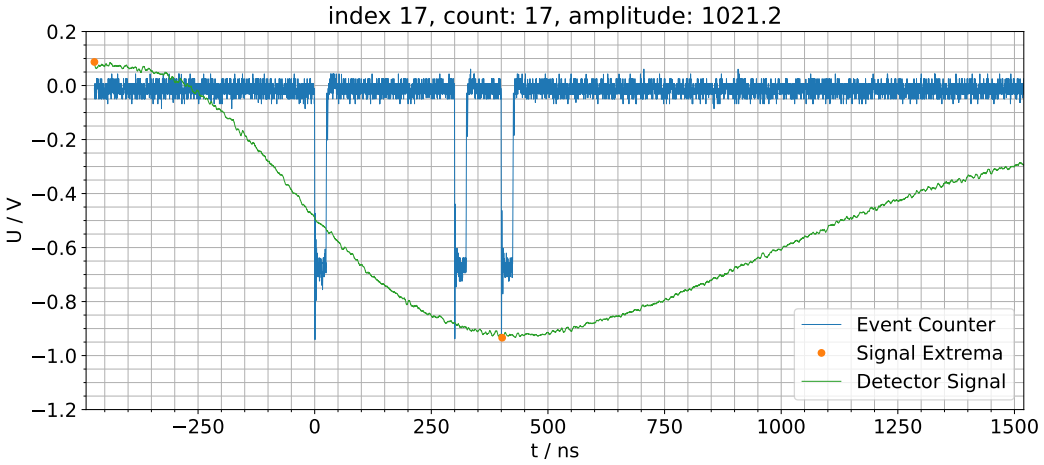


(b) Data from the oscilloscope. The positions of the peaks differ to data from the new preamplifier since the two preamplifiers have different intrinsic gains and the settings of the amplifier were adjusted.

Figure A.3.: Histograms of amplitudes from the two readouts of the test setup, with data taken before the preamplifier was swapped.



(a) Without smoothing.



(b) With smoothing. The signal waveform's line width is increased for better visibility.

Figure A.4.: Example plots of a saved oscilloscope waveform, with and without smoothing applied to the signal channel. The event counter channel is shown in blue and the signal channel in green. The minima and maxima used to calculate the amplitude are marked with orange dots. The plot titles show the index of the waveform in the file, the decoded event counter value (17 or 0000 0000 0001 0001) and the calculated amplitude in mV. One can see that the extrema are less exaggerated when the signal is smoothed.

Acknowledgements

At the end of this thesis, I would like to express my sincere gratitude to all of the people who made it possible and helped me along the way. Thank you firstly to Prof. Dr. Klaus Desch for letting me write a thesis in his research group. Thank you to Dr. Jochen Kaminski for welcoming me in his sub-group, for helping me find a very interesting topic and enabling me to go to CERN for a part of it. This specific topic was proposed by Dr. Michael Lupberger to whom I am very grateful for this and for his help and guidance before and during the thesis. My gratitude also goes to Patrick Schwäbig for his help with FPGAs in Bonn.

A significant part of these four months was spent at CERN and I am very grateful to Dr. Eraldo Oliveri for welcoming me in the GDD group there. My gratitude also goes to Dr. Dorothea Pfeiffer who greatly helped me in understanding the VMM3a/SRS FEC firmware, always answered my questions thoroughly, implemented the event counter in the vmm-sdat software and spent the better part of an afternoon helping me install the software on my laptop. Thank you to Dr. Hans Müller for his additional guidance and help regarding electronics, especially with respect to the proposed event counter input. I am also grateful to all of the GDD members, among them Karl Flöthner and Dr. David Marques, as well as all the summer students in the group for helping me in many different ways and making my stay a very pleasant one.

Taking part in the DRD1 test beam was a very fascinating experience. Thank you to Dr. Florian Brunbauer and Emma Tuovinen for providing me with data which I could not take myself since I had to leave early. Thank you finally to Dr. Lucian Scharenberg for his extensive guidance during and after my stay at CERN and for answering many questions about the VMM3a/SRS readout and everything surrounding it.

Bibliography

[1] H. Kolanoski and N. Wermes. *Teilchendetektoren*. Springer Spektrum Berlin, Heidelberg, 2016.

[2] S Martoiu et al. "Development of the scalable readout system for micro-pattern gas detectors and other applications". In: *Journal of Instrumentation* 8.03 (Mar. 2013), p. C03015. DOI: 10.1088/1748-0221/8/03/C03015. URL: https://dx.doi.org/10.1088/1748-0221/8/03/C03015.

[3] M. Gruber et al. "SRS-based Timepix3 readout system". In: *Journal of Instrumentation* 17.04 (Apr. 2022), p. C04015. DOI: 10.1088/1748-0221/17/04/C04015. URL: https://dx.doi.org/10.1088/1748-0221/17/04/C04015.

[4] M. Lupberger et al. "Implementation of the VMM ASIC in the Scalable Readout System". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 903 (2018), pp. 91–98. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.06.046. URL: https://www.sciencedirect.com/science/article/pii/S016890021830768X.

[5] L. Scharenberg et al. "Development of a high-rate scalable readout system for gaseous detectors". In: *Journal of Instrumentation* 17.12 (Dec. 2022), p. C12014. DOI: 10.1088/1748-0221/17/12/C12014. URL: https://dx.doi.org/10.1088/1748-0221/17/12/C12014.

[6] J Toledo et al. "The Front-End Concentrator card for the RD51 Scalable Readout System". In: *Journal of Instrumentation* 6.11 (Nov. 2011), p. C11028. DOI: 10.1088/1748-0221/6/11/C11028. URL: https://dx.doi.org/10.1088/1748-0221/6/11/C11028.

[7] B. Povh et al. *Teilchen und Kerne: Eine Einführung in die physikalischen Konzepte*. ger. Springer Berlin Heidelberg, 2013. ISBN: 9783642378225.

[8] Wikimedia Commons. *File:MMPrincipe.png*. [Online; accessed 29-June-2025]. 2023. URL: https://commons.wikimedia.org/w/index.php?title=File:MMPrincipe.png&oldid=830223272.

[9] L. Scharenberg et al. "Characterisation of resistive MPGDs with 2D readout". In: *Journal of Instrumentation* 19.05 (May 2024), P05053. DOI: 10.1088/1748-0221/19/05/P05053. URL: https://dx.doi.org/10.1088/1748-0221/19/05/P05053.

[10] D. Pfeiffer et al. "Rate-capability of the VMM3a front-end in the RD51 Scalable Readout System". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1031 (2022), p. 166548. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2022.166548. URL: https://www.sciencedirect.com/science/article/pii/S016890022200153X.

[11] L. Scharenberg, D. Pfeiffer, and H. Muller. *VMM3a/SRS Documentation*. [Online; accessed 21-July-2025]. URL: https://vmm-srs.docs.cern.ch/.

[12] M. Cortesi. *Update on VMM3a/SRS activities at FRIB*. [Online; accessed 26-Jul-2025]. 2024. URL: https://indi.to/cjNDH.

[13] D. Pfeiffer and G. Chang. *VMM3a Virtex-6 FEC Firmware; branch ext_trg_new*. GitLab repository not publicly available.

[14] *Standard NIM Instrumentation System*. US NIM Committee, 1990. URL: https://www.osti.gov/servlets/purl/7120327.

[15] Russell Merrick. *Getting Started with FPGAs*. eng. No Starch Press, 2023. ISBN: 1098163478.

[16] Xilinx. *Virtex-6 Family Overview*. 2015. URL: https://docs.amd.com/v/u/en-US/ds150.

[17] Wikimedia Commons. *File:4-Bit SIPO Shift Register.svg*. [Online; accessed 19-July-2025]. 2024. URL: https://commons.wikimedia.org/w/index.php?title=File:4-Bit_SIPO_Shift_Register.svg&oldid=860905157.

[18] P. Ashenden. *VHDL Tutorial*. 2004. URL: https://www.eecs.umich.edu/courses/doing_dsp/handout/vhdl-tutorial.pdf.

[19] Lucian Scharenberg. *Next-Generation Electronics for the Read-Out of Micro-Pattern Gaseous Detectors*. May 2023.

[20] J. Bortfeldt et al. "PICOSEC: Charged particle timing at sub-25 picosecond precision with a Micromegas based detector". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 903 (2018), pp. 317–325. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2018.04.033. URL: https://www.sciencedirect.com/science/article/pii/S0168900218305369.

[21] E. Delagnes et al. "The SAMPIC Waveform and Time to Digital Converter". In: *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. 2014, pp. 1–9. DOI: 10.1109/NSSMIC.2014.7431231.

[22] M. Lisowska et al. "Towards robust PICOSEC Micromegas precise timing detectors". In: *Journal of Instrumentation* 18.07 (July 2023), p. C07018. ISSN: 1748-0221. DOI: 10.1088/1748-0221/18/07/c07018. URL: http://dx.doi.org/10.1088/1748-0221/18/07/C07018.

[23] M. Raymond et al. "Design and results from the APV25, a deep sub-micron CMOS front-end chip for the CMS tracker". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 466.2 (2001). 4th Int. Symp. on Development and Application of Semiconductor Tracking Detectors, pp. 359–365. ISSN: 0168-9002. DOI: https://doi.org/10.1016/S0168-9002(01)00589-7. URL: https://www.sciencedirect.com/science/article/pii/S0168900201005897.

[24] D. et. al. Pfeiffer. *vmm-sdat repository*. [Online; accessed 30-Jul-2025]. URL: https://github.com/ess-dmsc/vmm-sdat.

[25] *ROOT Data Analysis Framework*. [Online; accessed 22-July-2025]. URL: https://root.cern/.

[26] CERN Experimental Areas Group. *Experimental Hall North - EHN1*. [Online; accessed 26-Jul-2025]. URL: https://be-dep-ea.web.cern.ch/experimental-areas/north-area/ehn1.