DI/OT MoniMod PSI Radiation Test Report

GENERAL INFORMATION AND RESPONSIBILITIES

Concerned equipment:	MoniMod
Group:	BE/CO
Device Under Test (DUT):	MoniMod v0.2p
Test Dates:	July 1–2 2020
Specifications by:	Grzegorz Daniluk, Christos Gentsos, Evangelia Gousiou
Tested by:	Christos Gentsos, Evangelia Gousiou
HW development by:	Christos Gentsos
Report by:	Christos Gentsos
SW development by:	Christos Gentsos

OVERVIEW

This document presents the results derived from the DI/OT MoniMod radiation tests performed at the Proton Irradiation Facility (PIF) of the Paul Scherrer Institute (PSI) using a 200 MeV proton beam. The Devices Under Test (DUT) are version 0.2p MoniMod prototypes. These modules feature fan control circuitry and PMBus-compatible power rail monitoring, with the I2C, ADC and PWM functions these features require being provided by peripherals in a SAMD21 Cortex-M0+ microcontroller.

As part of the DI/OT platform, the MoniMod is to be used in future instrumentation operating in radiation-exposed areas around the HL-LHC, with the worst-case being the top floor (L1) of the RR alcoves at P1 and P5, with TID levels of 25 Gy/yr, HEH fluence of $1.4 \cdot 10^{10}\,\mathrm{cm}^{-2}$/yr, and a 1 MeVn-eq thermal neutron fluence of $7 \cdot 10^{10}\,\mathrm{cm}^{-2}$/yr.[1] Consequently, the purpose of these radiation tests was to evaluate the behavior with respect to Total Ionizing Dose (TID) and Single Event Effects (SEEs). Furthermore, the results of these tests will help with quantifying the efficiency of a software technique that has been used to mitigate Single Event Upsets (SEUs) in the data stored in the microcontroller RAM.

All presented test data were recorded in the campaign of the week of June 29 – July 3, 2020, at the PSI PIF facility.

[1] Information taken from the *Radiation level specifications for HL-LHC* document: https://edms.cern.ch/ui/file/2302154/0.9/HL-LHC_specification_document_draft_1_July_2020.pdf

OUTLINE

1 DUT DESCRIPTION AND OPERATING CONDITIONS

The Distributed I/O Tier project (HL-LHC Work Package 18.2) Monitoring Module (DI/OT MoniMod) is a monitoring module developed for the DI/OT project's power supply and (optional) fan tray and based on the ATSAMD21G18 Cortex-M0+ microcontroller. It can monitor voltage and current consumption for up to three power rails, host up to three temperature sensors, and control up to three fans without requiring them to support PWM. The module is accessed and managed through a PMBus interface.

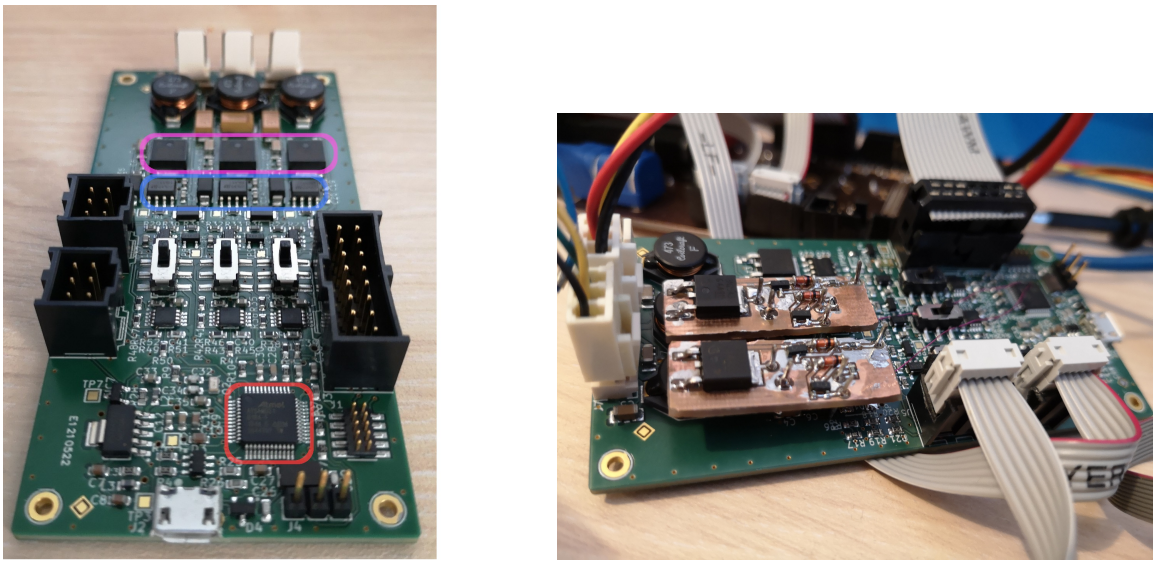
A MoniMod module can be seen on figure 1a: the microcontroller is marked by a red square, the gate drivers are pointed out by the blue one, and the MOSFETs are highlighted in purple. The fan control circuitry needed extensive modifications to work properly, these were implemented by adding a couple of patch wires, removing components from the board and adding extra components on two little "patch boards", seen on figure 1b. Unfortunately, patching the bugs found in the fan control circuits of this prototype has led to issues with signal integrity with the fan PWM adversely affecting the PT100 driver circuits to the point where it is impossible to use them while the fans are spinning.

1.1 DUT Functionality

1.1.1 PMBus Interface

The firmware implements a PMBus command subset that enables voltage, current and temperature monitoring, as well as both fan monitoring and control. To improve data transfer reliability, Packet Error Checking (PEC), as defined by the SMBus standard, is supported.

Non-standard functionality such as resetting the microcontroller, toggling PEC or setting up fan speed control via temperature curves can be accessed using extended PMBus commands.



(a) Before the necessary patches **(b) After applying the necessary patches**

Figure 1: A MoniMod module before and after the necessary patches

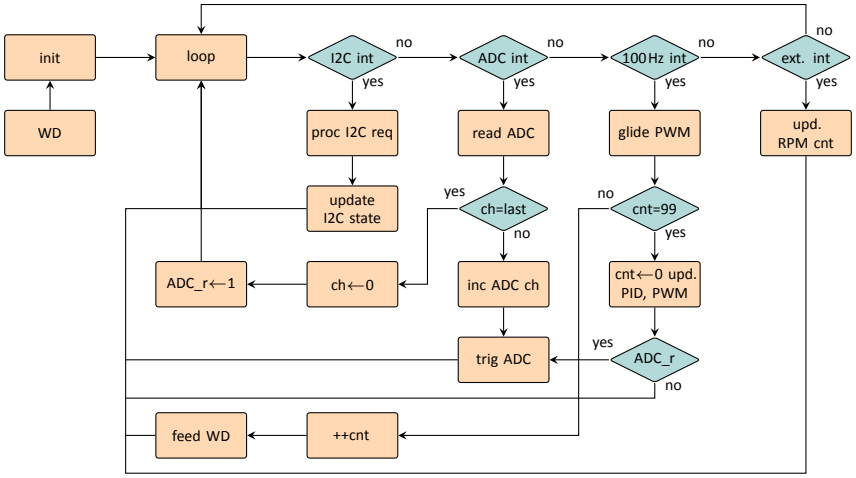


Figure 2: The control flow of the software

1.1.2 Power Rail Monitoring

The ADC peripheral integrated in the microcontroller is used to enable the MoniMod module to monitor the voltage and the current draw of up to three power rails. For the voltage monitoring a simple voltage divider is used in place of an analog front-end as the measurement requirements are relaxed.

The current draw is measured using high-side shunt resistors and a simple circuit based on a rail-to-rail opamp and a BJT that references the voltage differential of the shunt to ground level.

1.1.3 Temperature Monitoring

The current MoniMod prototypes implement circuits to use either PT100 or LM45-type temperature sensors, allowing a comparison of the two approaches in a radiation environment in order to choose which one will be used eventually; each channel is assigned a switch by which the user selects the type of sensor connected. A voltage reference (the same one that is used for the microcontroller ADC peripheral) is used to form a current source and bias the PT100; an amplifier is used to bring the voltage level to the appropriate range.

1.1.4 Fan Control

Three buck converters offer DC fan speed control, increasing the reliability of fans by modulating their speed according to the cooling needs without requiring them to be PWM-enabled which, in a radiation environment, might severely limit the available options.

The fan tachometer can be used to feed a PID controller, implemented in SW, which allows setting the fan speeds.

Temperature monitoring data from the three sensors can be combined using user-provided coefficients to provide an aggregate temperature for each fan, which can then be used to set their speed using user-provided fan curves.

1.1.5 Custom Bootloader

A custom bootloader allows the remote programming of the MoniMod's FW through its PMBus interface. Even though the remote programming functionality is only intended to be used when radiation levels are down, the bootloader should reliably bring up the FW in radiation.

1.2 Software mitigation

Although the MoniMod's function is not critical (hence the use of Atmel START and the ASF4 drivers), some measures have been taken in an effort to reduce interruptions and data corruption, improving the QoS.

The firmware uses TMR for variables stored in the RAM, using the COAST LLVM passes (for more information please visit the COAST readthedocs page). With this approach, each protected variable is stored three times and voting logic is employed when the variable is used. Protection can be enabled individually for each variable and, indeed, one has to cull the number of protected variables as the instruction overhead for the extra stores and the voting might outweigh the benefits in variables that are used too intensively.

The microcontroller also integrates a watchdog peripheral: this is fed every time the main timer callback runs, *i.e.* every 10 ms. The watchdog is set to trigger if it doesn't get fed for 20 ms — as soon as the main loop skips a beat. That should lead to a quick revival of the microcontroller and, consequently, to minimal downtime.

Another measure was employed and even though its impact in robustness would be minimal, its overhead is zero so it was kept: (i) placing the main loop instruction in an address that is a power of two and after the rest of the firmware; (ii) filling the rest of the memory with NOPs; and (iii) adding a trampoline at the end of the memory space. Since the main loop instruction address is a power of two, only one of the bits that make it up is set (it is of the form `0b1000000`). Any Program Counter (PC) bit flips in a bit lower than the one that is set will make the resulting PC value greater (*e.g.* `0b1001000`) and lead execution to the trampoline through the NOPs that populate the higher addresses.

1.3 Normal Operating Conditions

Operationnaly, two variants of the MoniMod module will be provided. One, space-optimised and without the fan control circuitry, will be integrated in the RaToPUS (DI/OT radiation-tolerant power supply) to provide PMBus-compatible monitoring. The other will provide full functionality of the tested prototype to be integrated in the optional fan tray of the DI/OT crate.

2 TEST SETUP AND PROCEDURE

2.1 Test Setup Description

The test setup used to validate a MoniMod's monitoring functionality consists of:

(1) a PC capable of running Python that has two free USB ports

(2) two USB extenders to reach components in the maze from the control room

(3) a test coordinator module, implemented on a "Feather M0 Basic Proto" SAMD21-based development board, that gives access to the MoniMod's PMBus interface (through the USB extender)

(4) a programmable PSU, controlled by the PC (through the second USB extender) .

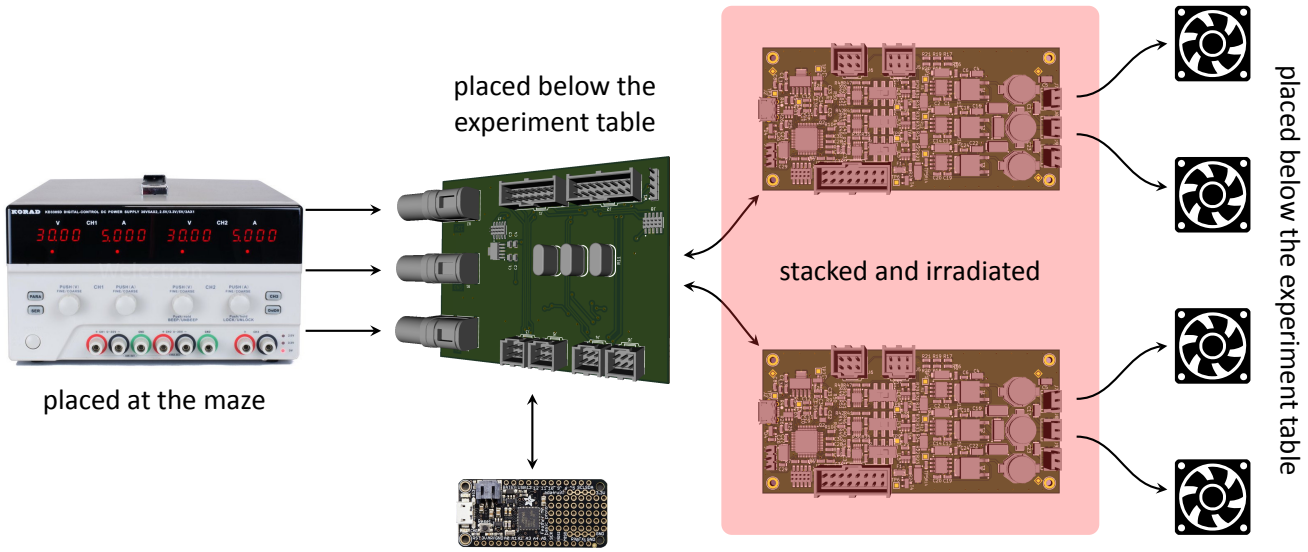


Figure 3: The test setup

2.1.1 DUT Connectivity

A simple board has been designed to provide power to the DUTs and connect them to the test coordinator; it also hosts some temperature sensors for the DUTs to monitor. Banana jacks are used to connect the PSU and power the DUTs with +12 V and +5 V. The board's schematic can be seen on figure 6 on page 10 — it should be noted that for these tests the programming headers (and the voltage regulator that powers them) were not soldered. Figure 3 gives a broad overview of the test setup connectivity and component placement, highlighting the central role of this board, and a picture of it can be seen on figure 4 on the following page.

The DI/OT crate allows a dual redundant power supply. Therefore, the two DUTs are connected to the same PMBus lines, in line with the MoniMods of the two redundant RaToPUS PSUs sharing the same bus, and irradiated together. In theory, this leaves open the possibility of a failure in one DUT taking down the whole bus and rendering the other DUT inaccessible; however, if this is a possibility, it is imperative to be aware of it and have an idea of the cross-section.

The monitoring port connections of the two DUTs are shown on tables 1a and 1b on the next page. It should be noted that, while both can read the same LM45-type temperature sensor, only one of the two can be connected to a PT100 sensor as there is a biasing circuit (a current source) and in such a case the readings would be double. Finally, also noteworthy is the fact that each DUT also monitors the current consumption on the other DUT's +5 V power rail.

2.1.2 Test Script Procedure

Through a simple serial protocol running through the USB-provided tty, the Python script periodically orders the test coordinator to initiate PMBus transactions, querying the DUT for monitoring data and setting the fan speeds; in these tests this period has been set to 5 s. The PSU-reported values for voltage and current are also read out and logged so that they can be compared against the values measured by the DUTs. In an effort to better cover the range of allowed operating conditions, the script also makes the PSU slowly oscillate the +12 V and +5 V power rail voltages and the fan speeds around their set values, within a

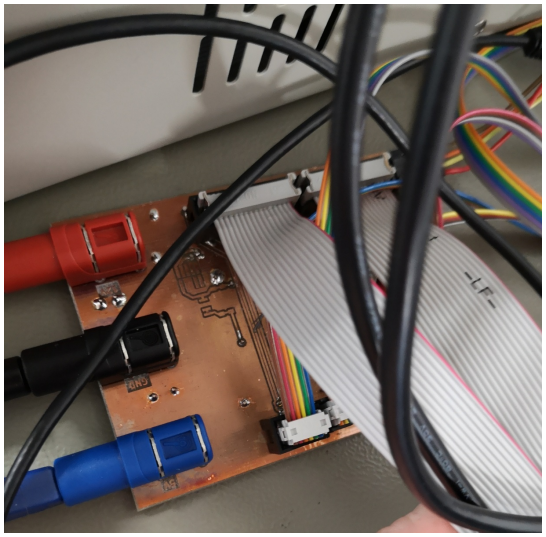


Figure 4: DUT powering and connection board

predefined margin.

The MoniMod can be remotely power-cycled. The test coordinator will automatically power-cycle both DUTs (i) once communicating with a DUT has consecutively failed a user-provided number of times (here set to 3); and (ii) at pre-determined intervals, in order to gather more statistics on the robustness of the bootloader when irradiated.

Timestamped logs are saved in CSV format, easy to be parsed by the pandas libary for offline processing. Each line in the log contains the following information: (i) the timestamp; (ii) a binary identifier of which of the two DUTs the line concerns; (iii) a transaction count; (iv) the software TMR error counter; (v) the I2C return code; and (vi) the monitoring data (voltages, currents, temperatures, tachometer readouts). The return code of a successful I2C transaction is zero. To collect one line of data many transactions are involved, so a single return code aggregate is logged by combining all the return codes with a bitwise OR operation. From these logs, an attempt to quantify the SEUs in the microcontroller will be made. As a first-order approximation, only serious ("obvious") deviations from the PSU-reported values (or from the

Table 1: Connectivity of the DUT sensing inputs

(a) DUT1 connections

Channel	Assignment
VCh1	+12 V
VCh2	+5 V
VCh3	+5 V
ICh1	$Is_{+12\,V}$ (DUT1)
ICh2	$Is_{+5\,V}$ (DUT1)
ICh3	$Is_{+5\,V}$ (DUT2)
TCh1	LM45
TCh2	PT100
TCh3	—

(b) DUT2 connections

Channel	Assignment
VCh1	+12 V
VCh2	+5 V
VCh3	+5 V
ICh1	$Is_{+12\,V}$ (DUT2)
ICh2	$Is_{+5\,V}$ (DUT2)
ICh3	$Is_{+5\,V}$ (DUT1)
TCh1	LM45
TCh2	—
TCh3	PT100

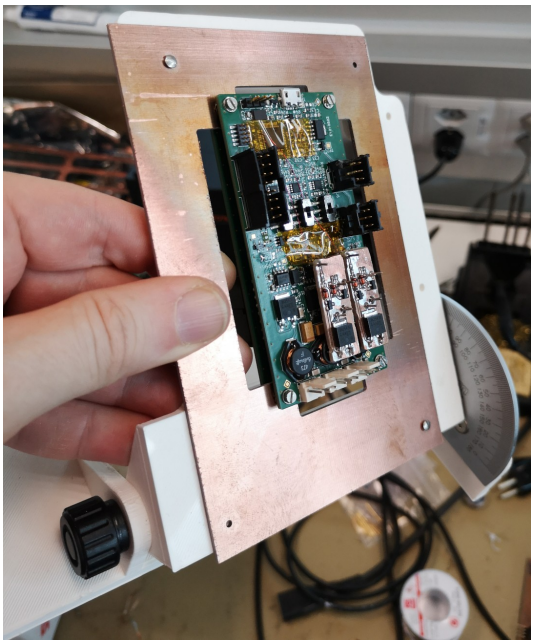


Figure 5: The MoniMod stack mounted on the angled support (which was not used, eventually)

mean of the measured values for the temperatures) will be counted as SEUs.

2.2 DUT Positioning

Two functionally identical boards are irradiated simultaneously; the two DUTs are hereby referred to as *MM0* and *MM1*. Although they are equivalent, there is a visual cue that sets them apart, the "patch boards" on *MM1* are somewhat uneven. Although their function is the same there is a significant difference in their FW, as *MM1*'s FW features the SW TMR technique to (hopefully) reduce SEUs.[2]

Initially an angled support was developed (see figure 5) in order to irradiate a two-module stack at an angle, requiring a single irradiation for the whole test. However, owing to the reduced beam time availability, other DUTs had to be irradiated in the same run; mounting those made it impossible to use the angled support.

Without the angled support, the two-module stack required two irradiations: one to cover the microcontrollers and one for the fan control circuitry — the distance between the centers of the two irradiations is 45 mm. The impact one board would have on the beam intensity should be negligible considering our purposes but, for the sake of completeness, it has to be mentioned that the beam crosses *MM1* first and then *MM0*. The latter sits at the back of the stack, with its orientation represented by that of *MM1* after a 180° rotation about the module's long axis (the microcontroller of *MM0* is located behind that of *MM1*).

2.3 Beam Details and Instrumentation Positioning

Figure 7 on page 12 shows a coarse diagram of the PIF facility irradiation room and the maze, where the instrumentation is placed. The initial proton beam for PIF is delivered from the PROSCAN accelerator with the help of the primary energy degrader, which allows setting a few discrete initial beam energies

[2] It follows that the TMR error counter is only valid for *MM1*.

in the range from 200 MeV down to 30 MeV. The beam is subsequently guided to the Experimental Area where the PIF facility is located. For these tests, the primary energy was set to 200 MeV. The beam flux is measured by means of two ionization chambers IC1 and IC2, placed upstream and downstream the collimators, respectively.

The counting of the IC1 and IC2 are calibrated to measure the beam flux at the position of the DUT, which is 5 cm far away from the IC2 (downstream the collimator). That operation is done before the test by comparing the counting of the IC1 and IC2 with respect to the output of a scintillator detector, which is mounted in place of the DUT (5 cm from the IC2). The TID is provided by the facility and is given in Gy(Si). A 5 cm collimator is used to make the beam homogenous before exposing the DUTs to it.

The location of the DUT in front of the beamline can also be seen on figure 7 on page 12. The test coordinator is placed near the DUT, but well outside of the influence of the proton beam, under the experiment table. Its USB port is plugged into the far end of an Ethernet USB extender, which is connected to the other end of the extender, at the control room computer, via the patch board on the wall. The PSU is located in the Maze and is also reached with the help of a USB extender via a cable that runs through the same patch board.

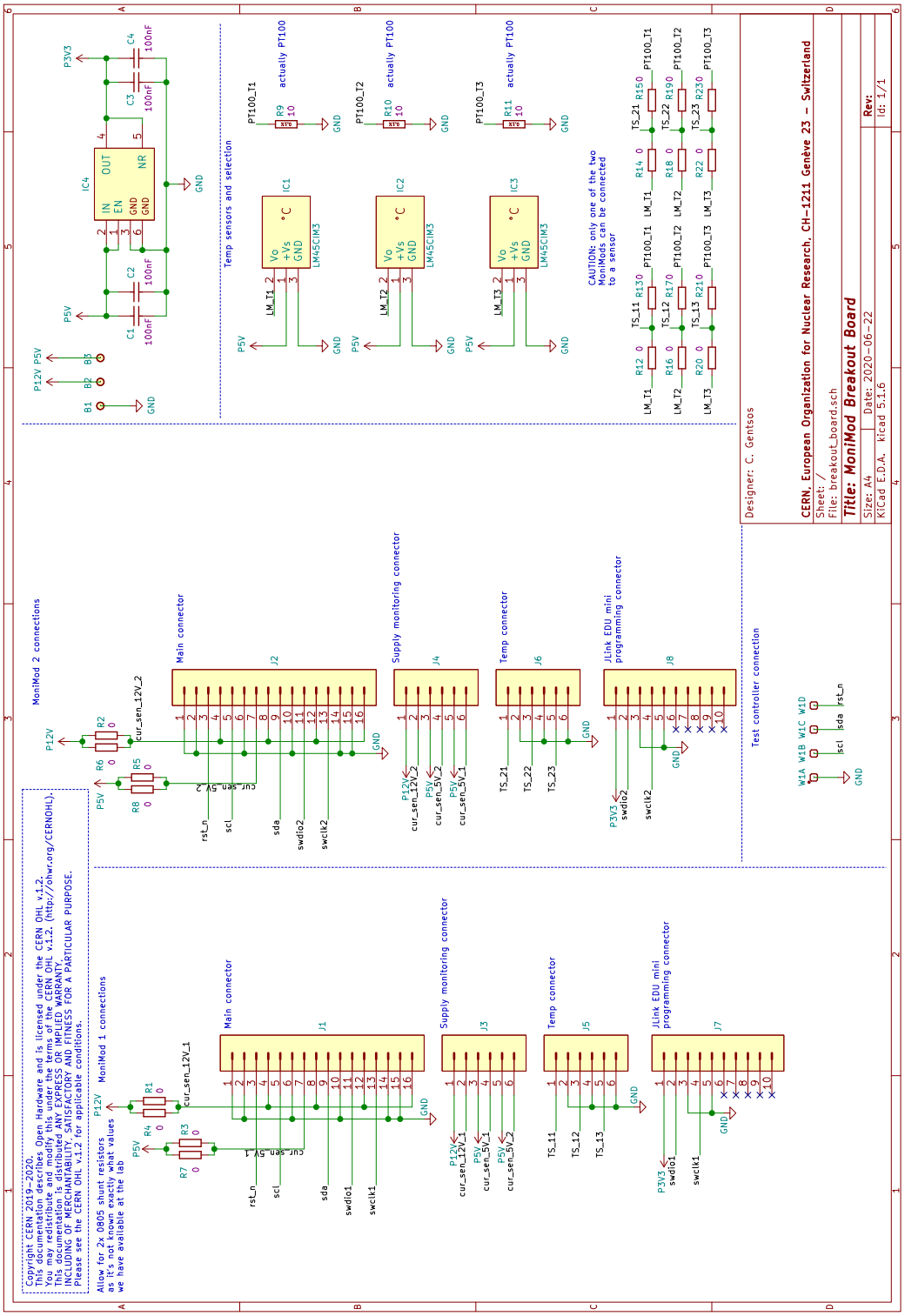


Figure 6: Schematics of the DUT powering and connection board

```
  usage: run_test.py [-h] [--log-interval LOG_INTERVAL] [--v1-period V1_PERIOD]           ↩
  [--v1-range V1_RANGE] [--v2-period V2_PERIOD] [--v2-range V2_RANGE] [--f1-setpoint      ↩
  F1_SETPOINT] [--f1-period F1_PERIOD]
                  [--f1-range F1_RANGE] [--f2-setpoint F2_SETPOINT] [--f2-period         ↩
                  F2_PERIOD] [--f2-range F2_RANGE] [--max-cons-errors MAX_CONS_ERRORS]   ↩
                  [--power-cycle-period POWER_CYCLE_PERIOD]
                  [--psu-log-file PSU_LOG_FILE] [--mm-log-file MM_LOG_FILE] [--dry-run]

Control the MoniMod test setup and log the results.

optional arguments:
  -h, --help            show this help message and exit
  --log-interval LOG_INTERVAL
                        Logging interval (2s-30s, default=5).
  --v1-period V1_PERIOD
                        12V fluctuation period (2s-3600s, default=600).
  --v1-range V1_RANGE   12V +- fluctuation percent (0%-10%, default=10%).
  --v2-period V2_PERIOD
                        5V fluctuation period (2s-3600s, default=600).
  --v2-range V2_RANGE   5V +- fluctuation percent (0%-10%, default=10%).
  --f1-setpoint F1_SETPOINT
                        Fan 1 setpoint (0-1000, default=500).
  --f1-period F1_PERIOD
                        Fan 1 fluctuation period (2s-3600s, default=300).
  --f1-range F1_RANGE   Fan 1 +- fluctuation percent (0%-50%, default=10%).
  --f2-setpoint F2_SETPOINT
                        Fan 2 setpoint (0-1000, default=400).
  --f2-period F2_PERIOD
                        Fan 2 fluctuation period (2s-3600s, default=300).
  --f2-range F2_RANGE   Fan 2 +- fluctuation percent (0%-50%, default=10%).
  --max-cons-errors MAX_CONS_ERRORS
                        Maximum consecutive errors before power cycling the MoniMods     ↩
                        (default is 2).
  --power-cycle-period POWER_CYCLE_PERIOD
                        Forcibly power cycle every POWER_CYCLE_PERIOD seconds (0-3600s,  ↩
                        0 for off, default is 600).
  --psu-log-file PSU_LOG_FILE
                        PSU log file.
  --mm-log-file MM_LOG_FILE
                        MoniMod log file.
  --dry-run             Do not write anything to the PSU or MoniMods.
```

Code Listing 1: Python Test Script usage

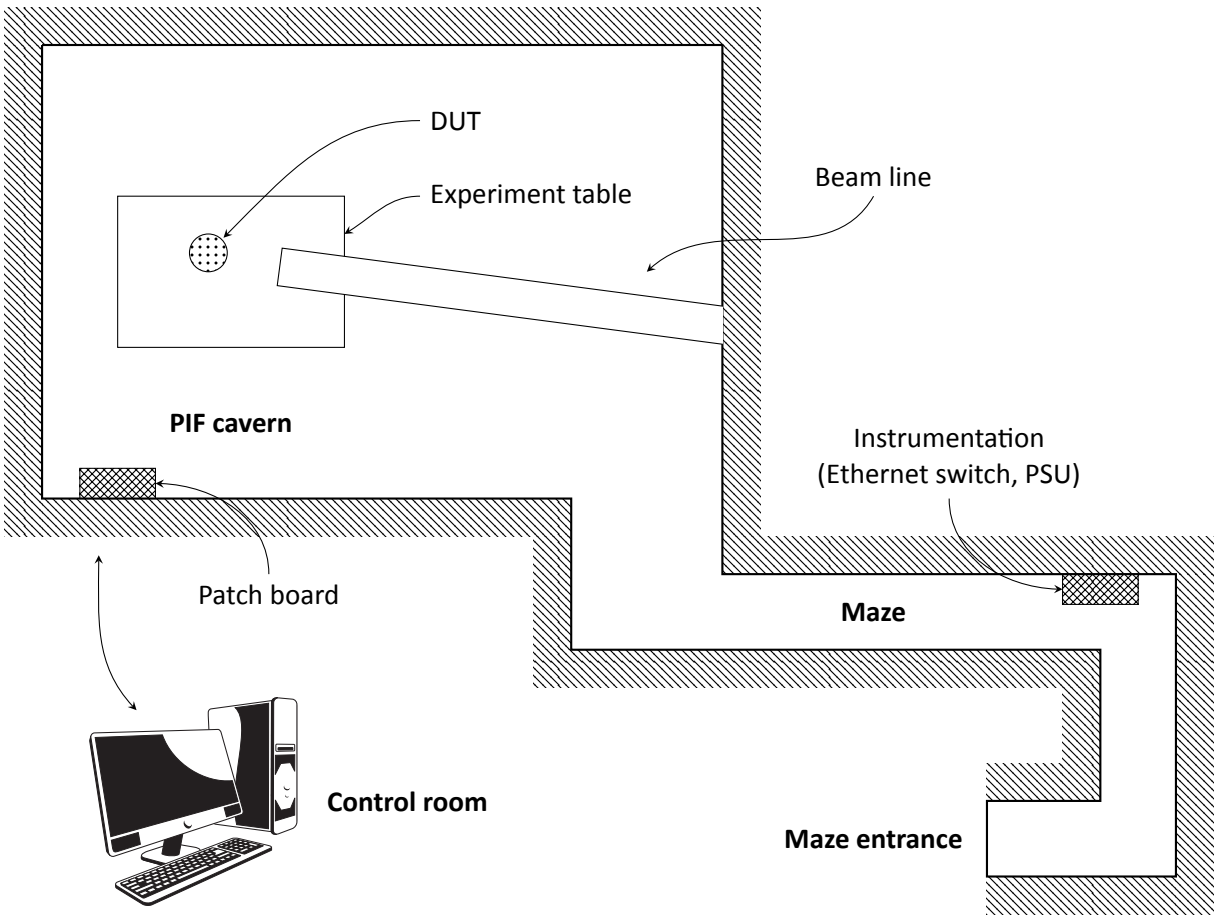


Figure 7: The PIF facility

3 EXPERIMENTAL RESULTS

3.1 Overview of the Runs

Runs 1–5 were carried out on the first day of the MoniMod tests, July 1. The first four runs reached the end-of-life point of the microcontrollers. As the examination of the gafchromic film after the irradiation showed, *the gate drivers were also accidentally irradiated in the first four runs* (see figure 8b on the next page for a clear view of this). Then, two more runs were carried out on the second day, July 2, after approximately 18 hours of annealing. For these last runs we decided to only irradiate and use in the setup a single DUT, in order to discount the possibility of any complex interactions between the two microcontrollers affecting the PMBus failures we were seeing. An overview of the beam conditions for each run with the fluence and TID figures can be seen on tables 2 and 3 for the microcontroller and the fan circuit irradiations, respectively.

Runs 1 and 2 are set up the same except that the voltage and fan oscillation ranges in the test script have been set to 10% for Run 1 and 5% for Run 2. For Run 3, the flux has been lowered to observe whether the SEFI cross-section would be affected. The log plots for these runs have been combined to figure 9 on page 16. I2C errors for *MM0* are indicated by red vertical dotted lines; errors for *MM1* are indicated by orange dotted lines. Power-cycles are marked by thicker, blue dotted lines.

For Run 4, since the SEFI cross-section did not seem to be affected by the flux, we set the flux close to the maximum attainable value to quickly irradiate the microcontrollers up to 500 Gy. Five minutes into this run, the fans stop spinning; this cannot be explained by a malfunction of the tachometer readout, as the +12 V current, as reported by the PSU, drops to zero. It can be explained by either the PWM peripheral or the gate driver chip being sensitive to radiation.

Moving on to the MOSFETs and Run 5: here, since the beam seemed quite stable, we managed to set the flux even higher, to $2.7 \cdot 10^8$, to quickly irradiate the fan control MOSFETs. The +12 V current starts increasing after $\sim$150 Gy and an interesting event happens, the second fan on each MoniMod starts spinning again.

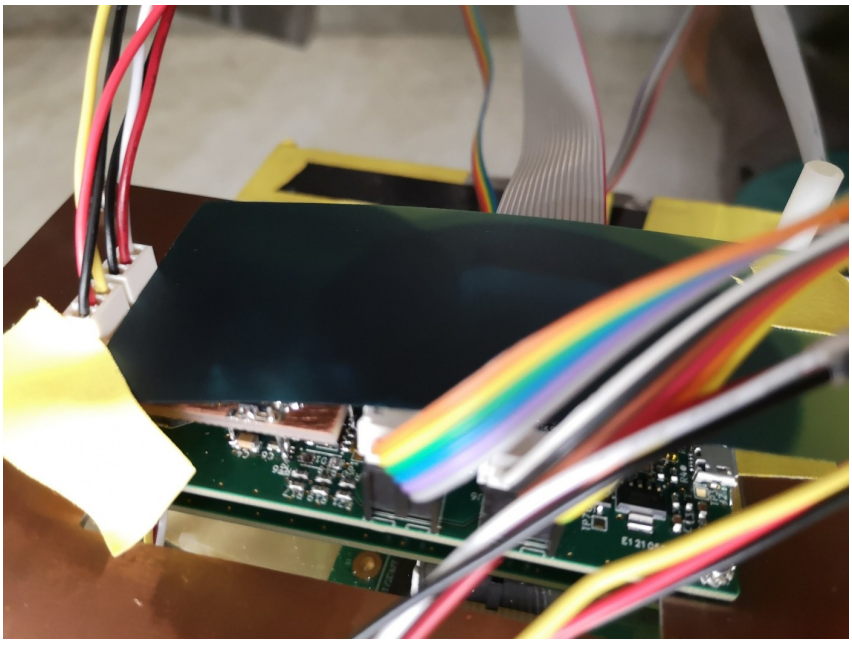
After the end of run 5, at the end of the day, the script was restarted to obtain another short log. At that point, there was another very interesting occurrence: the +12 V power rails of *both* boards started consuming as much current as they were given, hitting the PSU's current limiter of 3.3 A, but the fans weren't spinning (as reported by the two DUTs). Monitoring data from the MoniMods also shows that this

Table 2: Microcontroller irradiation run details

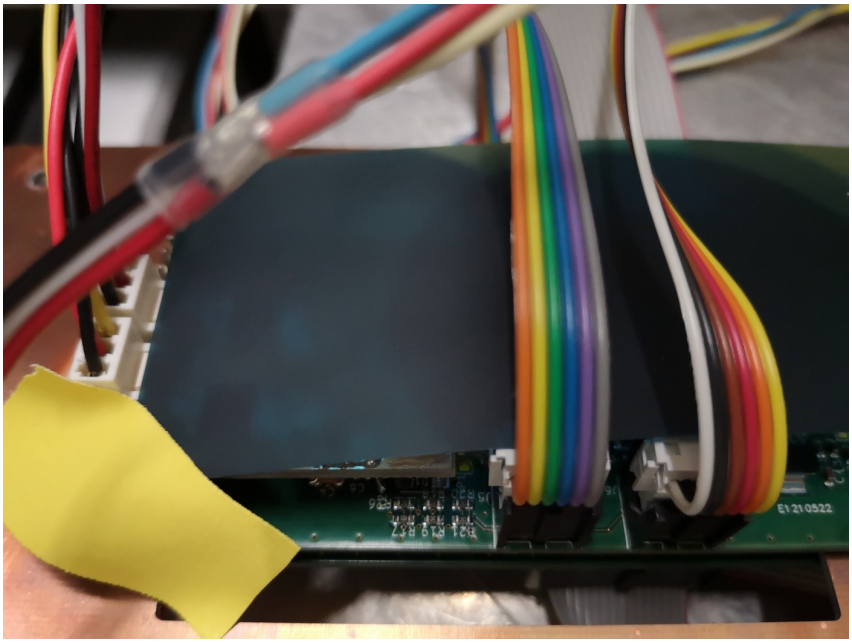
Run #	Flux	Fluence	Cum. Fluence	TID	Cum. TID
Run 1	$1 \cdot 10^8$	$1.69 \cdot 10^{11}$	$1.69 \cdot 10^{11}$	100	100
Run 2	$1 \cdot 10^8$	$1.69 \cdot 10^{11}$	$3.38 \cdot 10^{11}$	100	200
Run 3	$5 \cdot 10^7$	$5.07 \cdot 10^{10}$	$3.89 \cdot 10^{11}$	30	230
Run 4	$2.5 \cdot 10^8$	$4.56 \cdot 10^{11}$	$8.45 \cdot 10^{11}$	270	500
		18 hrs of annealing			
Run 6	$2.3 \cdot 10^8$	$4.95 \cdot 10^{11}$	$1.34 \cdot 10^{12}$	293	793
Run 7	$2.3 \cdot 10^8$	$1.47 \cdot 10^{11}$	$1.49 \cdot 10^{12}$	87	880

Table 3: Fan circuit irradiation run details

Run #	Flux	Fluence	Cum. Fluence	TID	Cum. TID
Run 5	$2.7 \cdot 10^8$	$5.66 \cdot 10^{11}$	$5.66 \cdot 10^{11}$	335	335



(a) View 1, the microcontroller sits well inside the irradiation circle



(b) View 2, the gate drivers were also inside the circle of the microcontroller irradiation

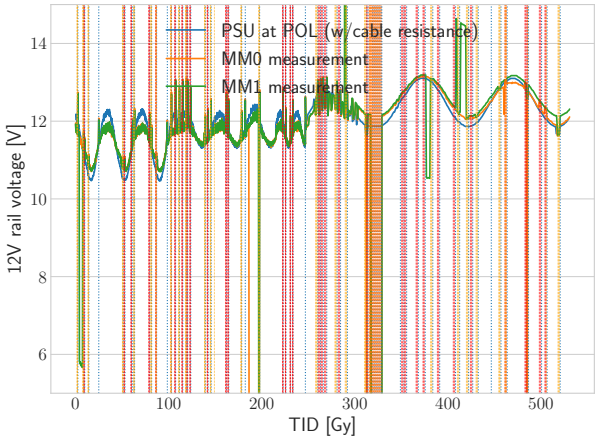
Figure 8: Two slightly different views of the gafchromic film after the two irradiations on the first day

current is distributed equally. The PSU was quickly switched off to prevent any catastrophic damage to the DUTs and, at the next power-up, the setup behaved normally — what's more, both MoniMods had their second fan working again.

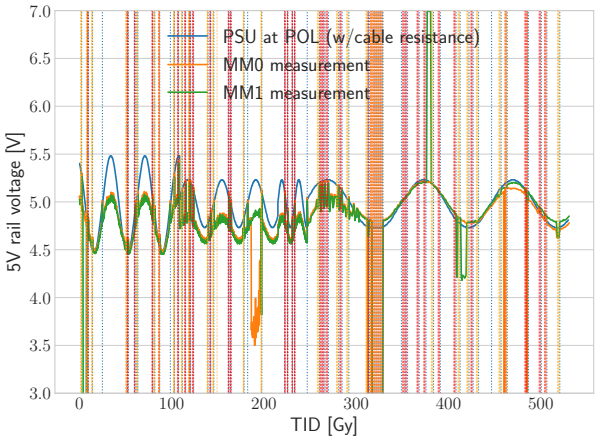
On the second day of irradiations, *MM0* was removed from the test setup and only the microcontroller of *MM1* was irradiated. Starting run 6, one can see on figure 9 on the following page that the fans don't seem to work (again) but there are no shorts observed, either. Around 100 Gy in, the +5 V power consumption dropped to 14 mA (from the 18 mA it was hovering about to that point) and the DUT stopped responding to remote power-cycling. After a full PSU power cycle, however, it came back to life. One could assume that the regulator started misbehaving after the total dose of 600 Gy it had been irradiated with thus far. The beam went out after providing us with a dose of 293 Gy.

In run 7, at 87 Gy we lost communication with the microcontroller again, with a slightly different signature with respect to the previous run: as evidenced on figure 12d on page 19, the +5 V current went slightly up to 19 mA instead of falling. The microcontroller never recovered after this point, having sustained a total dose of 880 Gy.

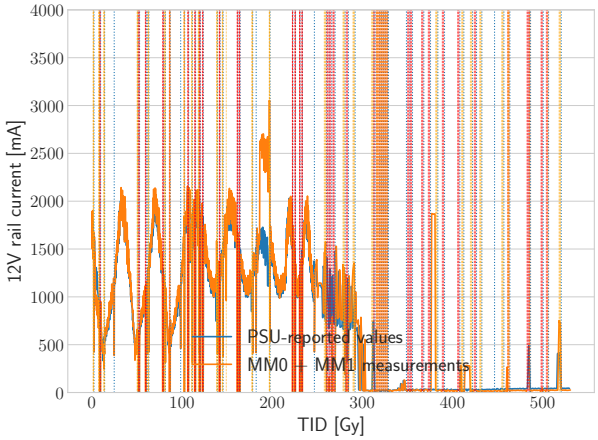
An interesting development emerged when trying to see whether the microcontroller would eventually wake up after this last run: even when shutting down the +5 V rail, the +12 V rail would report a 600 to 800 mA current draw. This cannot be explained by TID-induced damage in the MOSFETs, as those were irradiated on the first day.



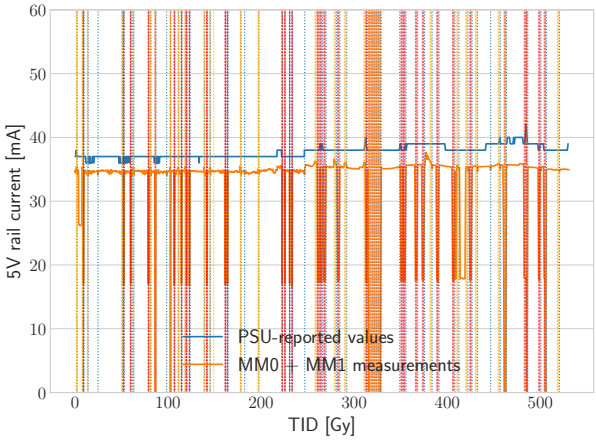
(a) +12 V rail voltage measurements



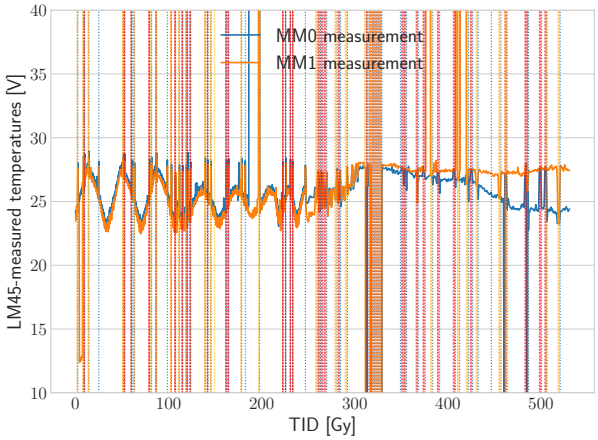
(b) +5 V rail voltage measurements



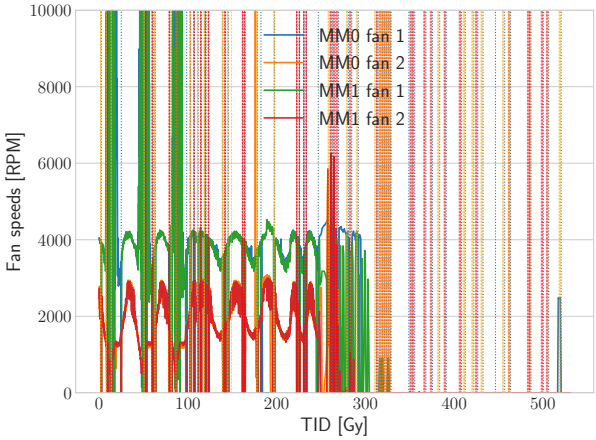
(c) +12 V rail current measurements



(d) +5 V rail current measurements

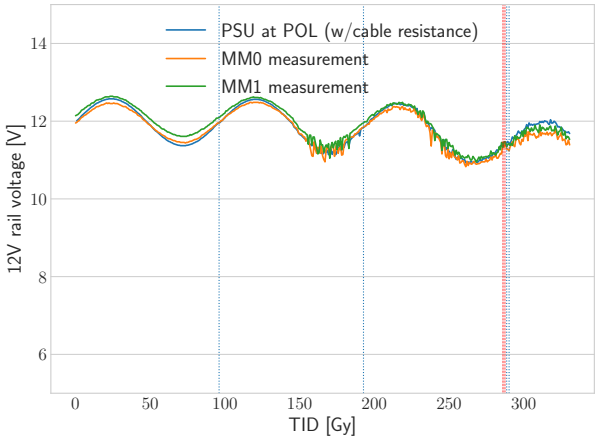


(e) LM45 temperature measurements

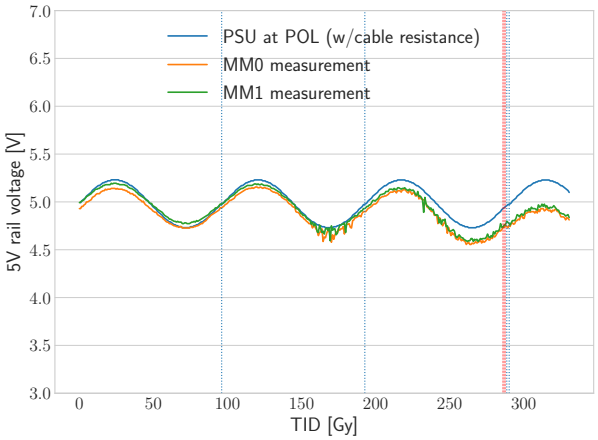


(f) Fan speed measurements

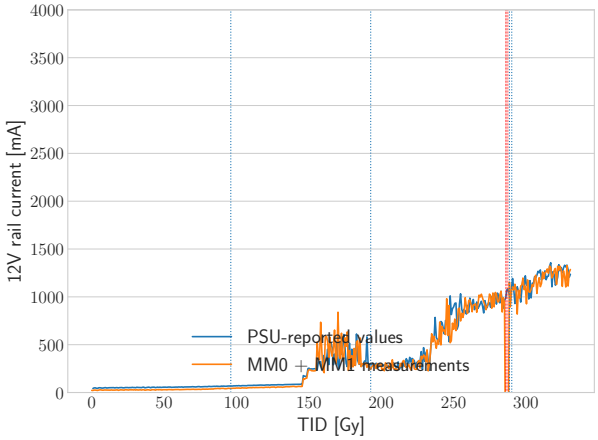
Figure 9: Day 1 runs



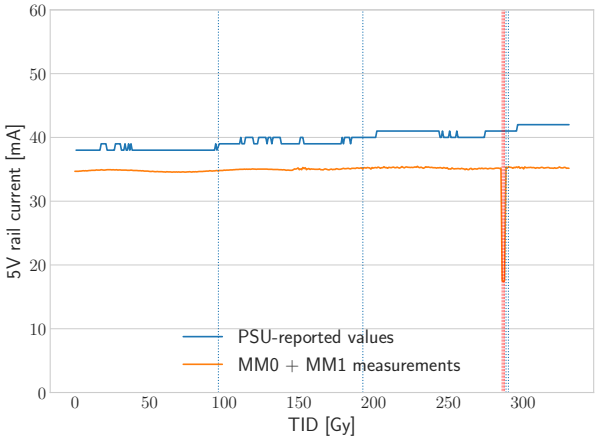
(a) +12 V rail voltage measurements



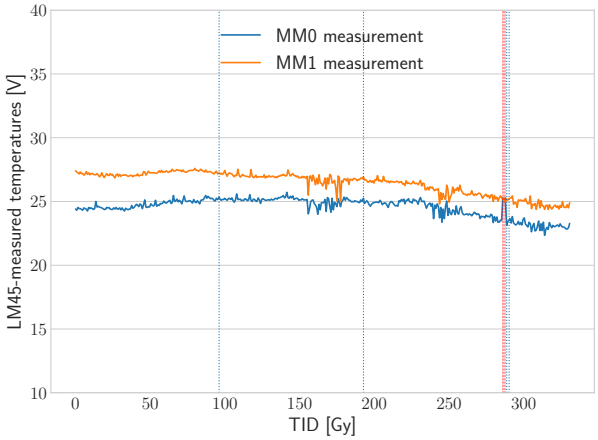
(b) +5 V rail voltage measurements



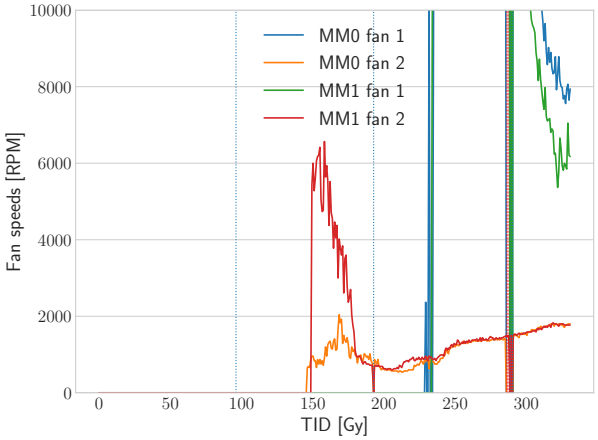
(c) +12 V rail current measurements



(d) +5 V rail current measurements

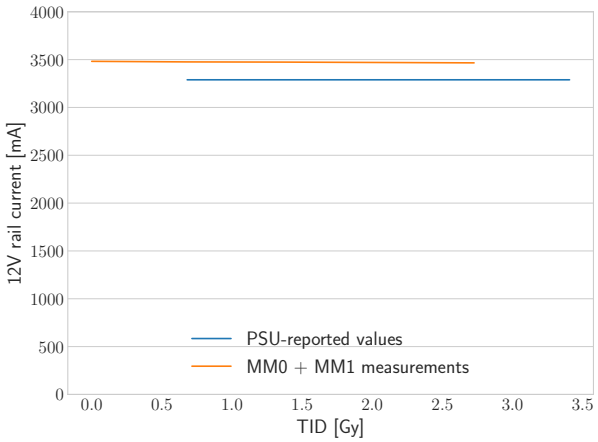


(e) LM45 temperature measurements

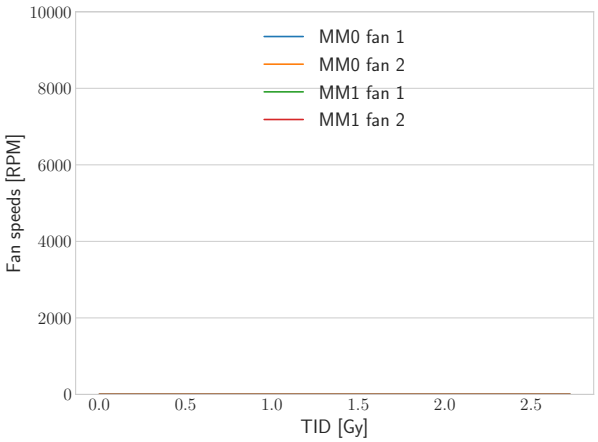


(f) Fan speed measurements

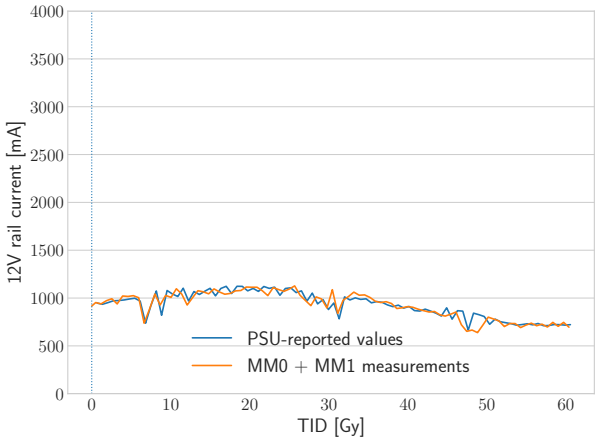
Figure 10: Run 5 (fan driver irradiation) measurements



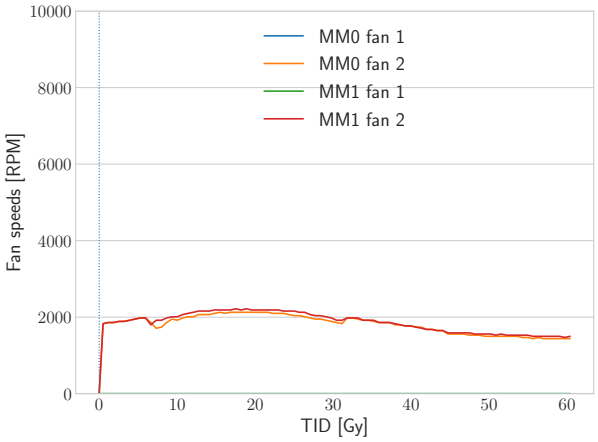
(a) +12 V rail current measurements



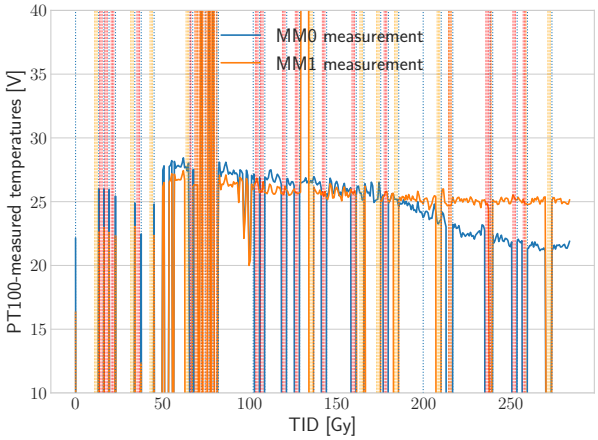
(b) Fan speed measurements



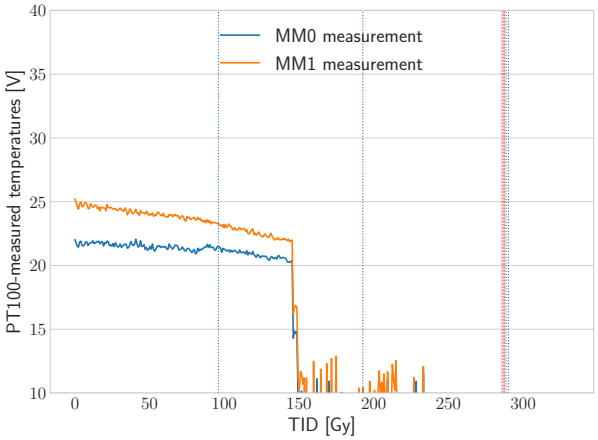
(c) +12 V rail current measurements



(d) Fan speed measurements

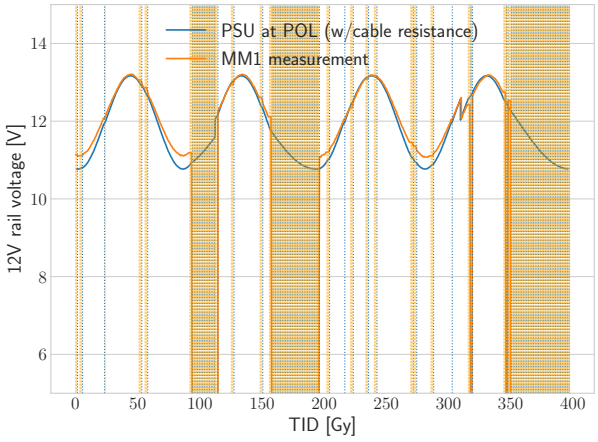


(e) Run 4 PT100 temperature measurements

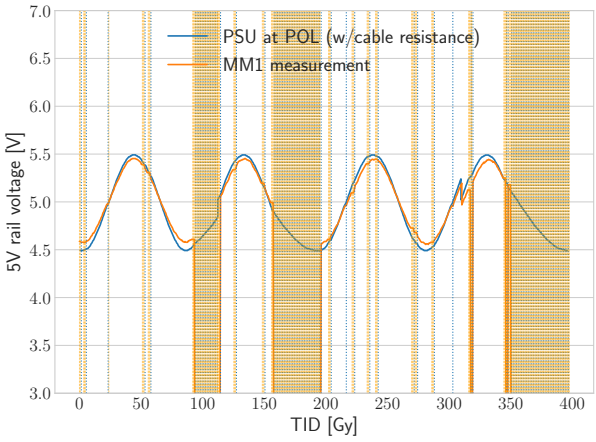


(f) Run 5 PT100 temperature measurements

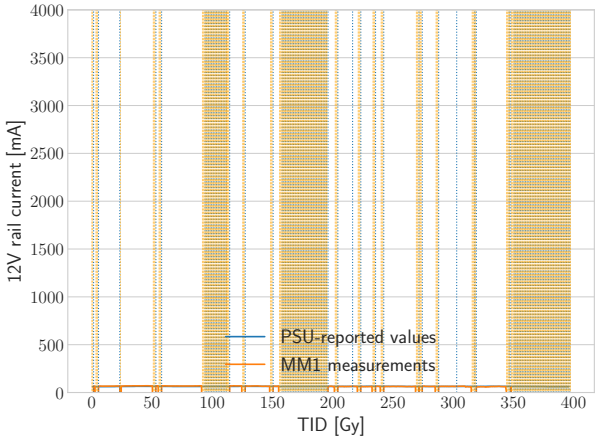
Figure 11: Short condition incident after MOSFET irradiation ended (subfigures a and b) and a short log at the power-up following that (subfigures c and d); run 4 and 5 PT100 temperature measurements (subfigures e and f)



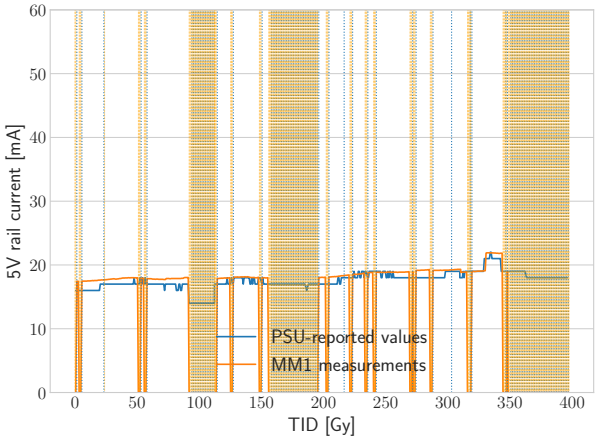
(a) +12 V rail voltage measurements



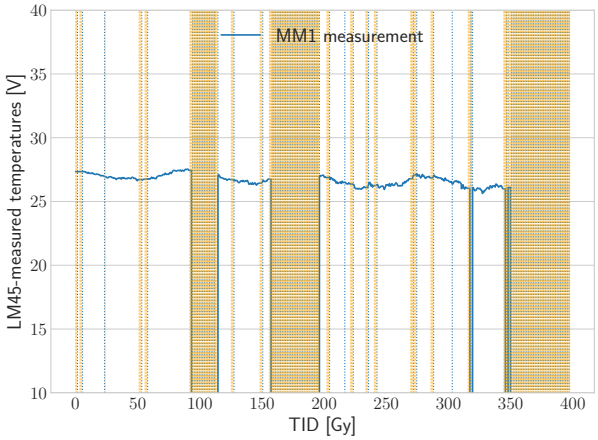
(b) +5 V rail voltage measurements



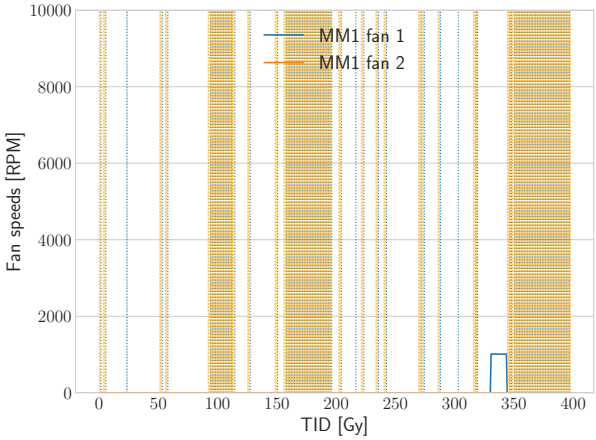
(c) +12 V rail current measurements



(d) +5 V rail current measurements



(e) LM45 temperature measurements



(f) Fan speed measurements

Figure 12: Day 2 runs

3.2 Analysis of the Test Results

To analyze various aspects of the log data and produce the plots, a Python script was written.

3.3 Functional interrupts and Interference Errors

Some upsets will cause the microcontroller to stop responding properly to I2C transactions until it is reset. It has not been investigated whether the microcontroller software can "know" that the transaction has failed so it can reset itself after a failed transaction, or if the I2C peripheral starts silently failing to properly perform transactions. Assuming the latter, we will consider those events to be Single Event Functional Interrupts (SEFIs),[3] and after three consecutive failed transactions both MoniMods are remotely power-cycled together.

If one takes a closer look at some of the plots, it can be observed that there are instances where right after a DUTs starts producing I2C transaction errors the other one follows, until they are power-cycled; one such instance is illustrated by figure 13. Since both units share the same I2C bus, it is a reasonable assumption that the erratic MoniMod can sometimes interfere with the other one's transactions. With the help of a Python script, these instances are detected and counted separately as IERRs (Interference Errors). The consecutive failures that make up the SEFIs are also counted separately.

The counts of SEFIs, isolated errors and interference errors, along with their cross-sections, calculated using the fluence figures from table 2 on page 13, are shown on table 5 on page 23.[4] It can be seen that *MM1* features 30% lower SEFI cross-sections, an order of magnitude less isolated errors, and causes an order of magnitude less interference errors to *MM0*. This could be attributed to the mitigation measures implemented in its firmware.

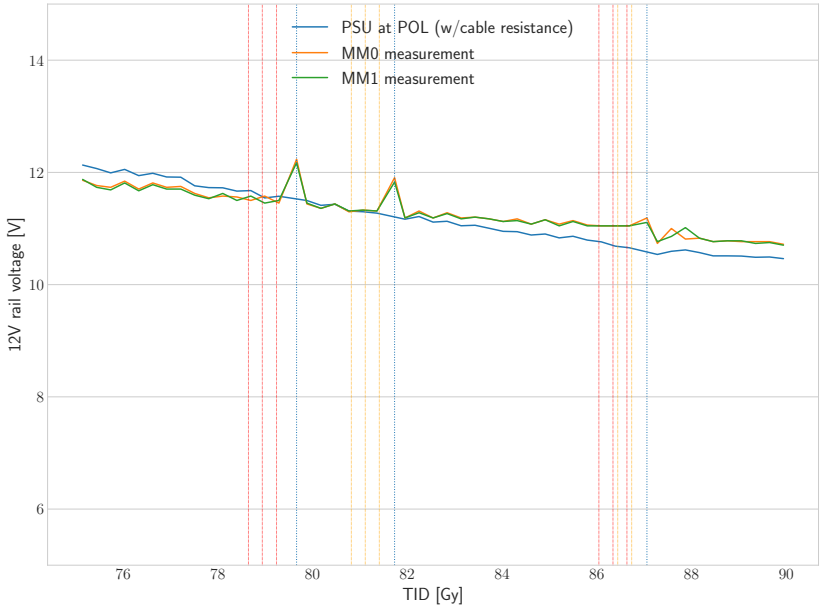


Figure 13: Interference errors within a SEFI pattern

[3] As many other radiation reports, this document eschews the JEDEC definition of SEFI and instead takes it to mean that a processor either locks up, goes to standby mode, produces exceptions continuously, or goes into an unknown state.

[4] For runs 6 and 7 the non-responsive periods were removed to get the cross-sections with the rationale that since the microcontroller had already reached a dose greater than 500 Gy, it would make sense that it would start misbehaving.

3.3.1 I2C Error Codes

The possible I2C error codes that are defined on the ASF I2C driver header can be seen on table 4. First, a few words on the error codes are warranted. Three of the error codes on that table, **I2C_ACK**, **I2C_ERR_ARBLOST** and **I2C_ERR_PACKAGE_COLLISION**, aren't actually used by the driver code (hence the strikethrough). When the "arbitration lost" flag is set by the peripheral, however, one of **I2C_ERR_BAD_ADDRESS** or **I2C_ERR_BUS** is returned, depending on whether the "bus error" flag is also set. Then, the **I2C_ERR_BUSY** can be returned when the bus is already busy when the test coordinator attempts to start a transaction; this would be seen in case a DUT were to hijack the bus. Finally, the driver has been extended to support PMBus-compliant Packet Error Checking (PEC) in I2C transactions; an error in the PEC checksum is indicated by an error code of **ERR_BAD_DATA**.

As mentioned in subsection 2.1.2 on page 6, the error codes from all the transactions of a monitoring readout cycle are combined through a bitwise OR, to simplify logging. In retrospect, that was not a great choice: since the error codes are negative and in two's complement, there are too many combinations that will return -1 (any combination of **I2C_NACK** or **I2C_ERR_BAD_ADDRESS** with either **I2C_ERR_BUS** or **I2C_ERR_BAD_DATA**), hampering any sort of extensive analysis, *e.g.* on the error signature of SEFIs.

In any case, the (combined) error codes have been retreived from the logs and plotted on figure 14 on the next page. From their distribution, it it quite possible that all the -1 errors (showing up as **COMB**) are actually **I2C_NACK** combined with **I2C_ERR_BUS**, but the possibility of other errors being involved as well cannot be discounted.

3.4 Fan Driver Short

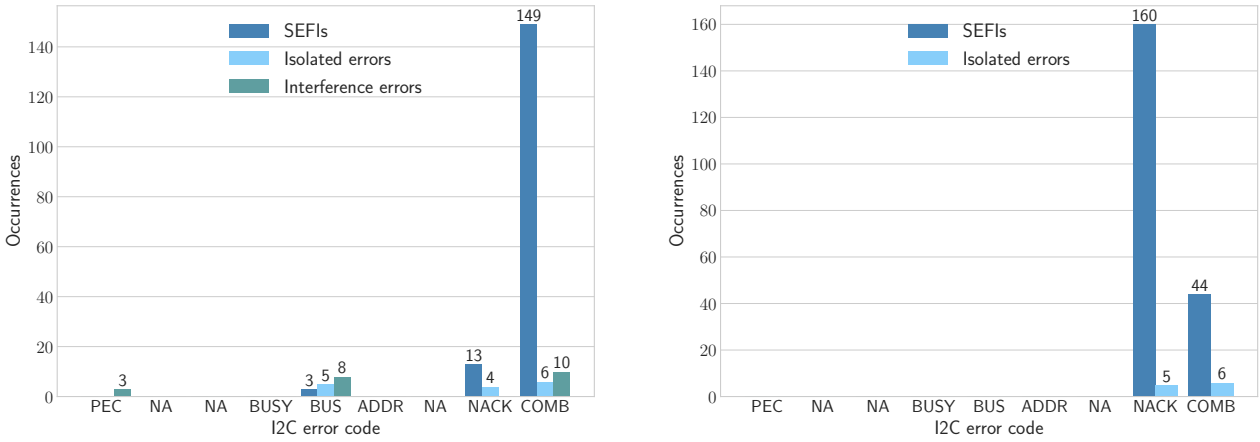
The fan driver short observed after the end of run 5 (see figures 11a and 11b on page 18) somehow resembles a Single Event Latch-up (SEL): there is a constant high current flow that only stops with a power cycle. However, the two DUTs report equal current draws; many devices must have suffered a SEL at the same time for that to happen, which should be near impossible.

One could perhaps try to attribute that to the threshold voltage shift of the fan driver MOSFETs but there are some points that would require an explanation: (i) at a rate of 6.3 mV/Gy[5] that shouldn't happen yet; (ii) the threshold must have continued shifting after the irradiation ended, as the current was double what it was at the end of run 5; and (iii) annealing appears to have happened in a *very* short time, as there was

Table 4: I2C error codes

Error	Code
I2C_OK	0
~~I2C_ACK~~	-1
I2C_NACK	-2
~~I2C_ERR_ARBLOST~~	-3
I2C_ERR_BAD_ADDRESS	-4
I2C_ERR_BUS	-5
I2C_ERR_BUSY	-6
~~I2C_ERR_PACKAGE_COLLISION~~	-7
ERR_BAD_DATA	-9

[5] As measured in the radiation report at https://edms.cern.ch/document/2207602 for a V_{gs} of 10 V.



(a) Day 1 (two MoniMods on the bus) **(b) Day 2 (only one MoniMod on the bus)**

Figure 14: I2C error code distribution, broken down by error type

no such behavior observed after powering up the setup again. It might be plausible for the quick annealing to have occured as the MOSFETs reached extremely high temperatures, due to the high current passing through them, but the two remaining arguments are left unexplained, making this theory less likely. For now, this shall remain unexplained.

3.5 Early Fan Driver Failure

Five minutes into run 4, the fans stop spinning; this cannot be explained by a malfunction of the tachometer readout, as the +12 V current (as reported by the PSU) drops to zero. It can be explained by either the PWM peripheral or the gate driver chip being sensitive to radiation.

As mentioned in subsection 3.1 on page 13, during run 4, the fans stopped spinning after a dose of 273 Gy. Two explanations can be offered: either the PWM peripheral is sensitive to radiation, or the gate driver (which also happened to be in the irradiated area) is. The position of the latter in the fan control circuit can be seen on the schematic diagram on figure 15 on page 24.

Unfortunately, both of these explanations are problematic. The PWM should be a relatively simple peripheral and since the rest of the microcontroller behaves uniformly well there is no obvious reason for the PWM to differ. However, fan driver functionality briefly returns on run 5 (see figure 10 on page 17), when we take the beam away from the microcontroller, allowing annealing effects to explain this but also failing to explain the consecutive loss of functionality on the second day. Concerning the gate driver chip, this has been tested to work fine up to 500 Gy in a previous campaign.[6] A difference in behavior in this setup might be explained by the use of a different (and problematic) lot in the prototypes but other explanations should be exhausted, first. There is a number of ways the radiation test differs that can be pointed out: (i) 14 V are used instead of 12 V; (ii) a 5 V input level is used instead of 3.3 V; (iii) an external bootstrap diode is used; and (iv) the **OUT** pin is grounded, changing the conditions for the the high-side driver internal level shifter. For now, this shall also remain unexplained.

[6] this document can be found at https://edms.cern.ch/document/2333834

3.6 ADC Behavior

Between the time the prototypes were received and the date of the radiation campaign there wasn't enough time to fully characterize them. Thus, a portion of that happened during the analysis of the radiation test data, along with the discovery of certain flaws in the design of these prototypes. First, the voltage measurements obtained on the first runs needed some scaling to match the PSU-reported values. These ADC inputs being unbuffered, it was assumed that the input resistance of the ADC was simply too low, skewing the results (indeed, it is less than 3.5 kΩ, which is a bit too low). A multivariate fit was performed using the Python script and scaling the measurements with the results of that leading to values that match the ones the PSU reports perfectly. However, things got more complicated when looking at data from run 4 and later: when the fan driving circuits stop drawing current, the adjusted measurements of the +5 V rail drift away from the PSU-reported values; no scaling is required to produce the right values anymore. It follows that this prototype suffers from some interference between channels, as the +12 V current goes up, the +5 V and LM45 channels report lower measurements. With that effect in mind, the scaling coefficients that were computed were eschewed and the measurements are plotted as-is, leading to the voltage measurement mismatches mostly observed in the plots of the first three runs — and the end of run 5, as the current at the +12 V rail starts going up.

The interference phenomenon aside, the ADC and its reference behaved very well with respect to TID, producing practically perfectly consistent results (as far as this testing method can discern) up to the end of the tests.

One interesting observation regarding the ADC is its apparent occasional misconfiguration. One can notice this by looking at *MM1*'s measurements at the beginning of run 1, around 5 Gy in, and *MM0*'s values just before the 200 Gy mark. Many values shift to invalid ranges concurrently, but if one notices closely they still track the waveform shape — they are really just shifted. This could be easily solved by having the firmware occasionally re-write the ADC configuration (blind scrubbing).

Table 5: Microcontroller Single Event Functional Interrupts

Run #	MM0 SEFIs	MM1 SEFIs	MM0 ERRs	MM1 ERRs	MM0 IERRs	MM1 IERRs
Run 1	6	7	0	0	1	2
Run 2	6	6	0	1	0	0
Run 3	4	0	0	0	0	0
Run 4	16	10	14	0	0	18
Day 1 total:	32	23	14	1	1	20
Cross-sections:	$3.8 \cdot 10^{-11}$	$2.7 \cdot 10^{-11}$	$1.6 \cdot 10^{-11}$	$1.2 \cdot 10^{-12}$	$2.4 \cdot 10^{-12}$	$2.4 \cdot 10^{-11}$
Run 6	—	15	—	3	—	—
Run 7	—	3	—	4	—	—
Day 2 total:	—	18	—	7	—	—
Cross-sections:	—	$2.8 \cdot 10^{-11}$	—	$1.1 \cdot 10^{-11}$	—	—

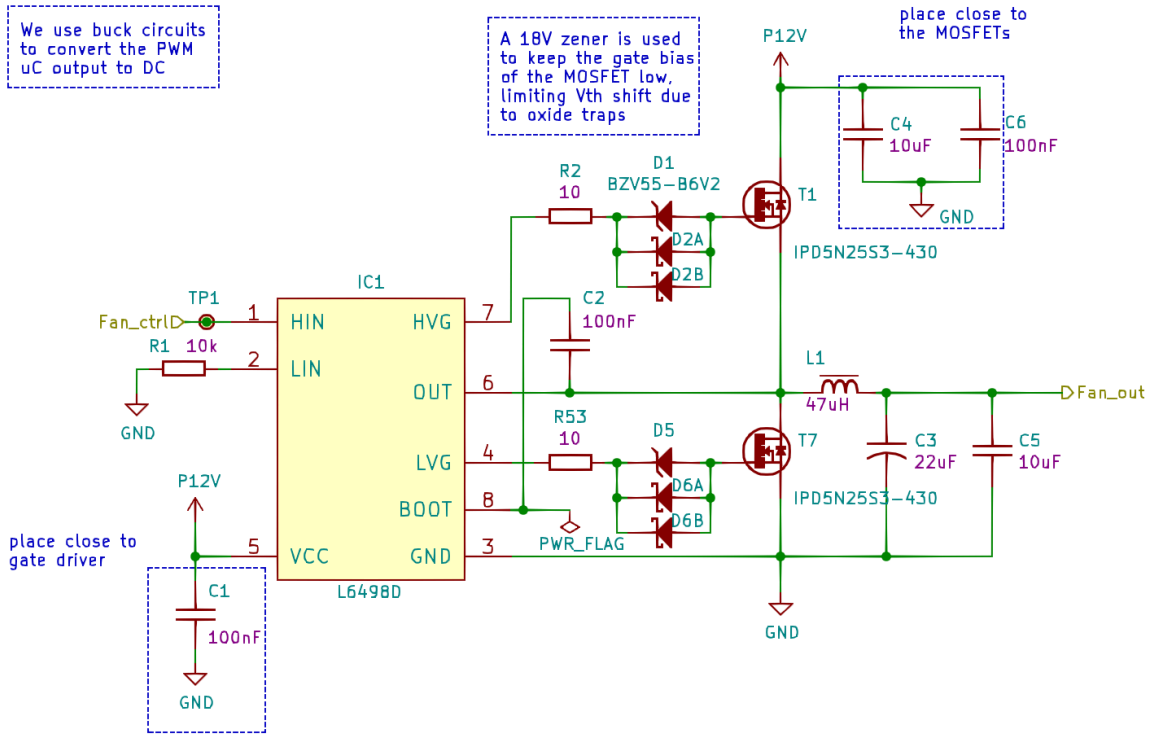


Figure 15: Fan control circuit schematics

4 CONCLUSIONS

The cross-section of SEFIs in the form of communication getting lost is $2.8 \cdot 10^{-11}$, a figure perfectly consistent with the estimates of the SAMD21 radiation campaign. Functionality can be quickly restored after a SEFI by remotely power-cycling the microcontroller.

The software fault mitigation approach has a measurable effect, reducing the cross-section of SEFIs by 30% and that of other I2C errors (including interference with the other device) by an order of magnitude.

Fan driving capability became erratic and then subsequently lost after a dose of ∼275 Gy. Either the PWM peripheral or the gate driver is responsible and there is not enough data to infer which one with any confidence. A short appeared in both MoniMods — at the same time — and current explanations for it are lacking.

The ADC and its reference are stable enough to continue gathering valid measurements well after 500 Gy and only suffer from the occasional configuration corruption, which can be fixed by implementing blind scrubbing.

Some flaws in the prototype design were found and need to be fixed for the next revision, *e.g.* the fan control interference in voltage and PT100 measurements.