**NuPRISM April 2021 Xilinx Firmware Status Report**

**Author(s):** J. Cronin, L. Bidulka

**Release:** 2.0

**Release Date:** 27/04/2021

**Document Type:** Report

**Keywords:** IWCD, NuPRISM, Firmware, Xilinx, Altera, UltraScale+, Zynq, MAX10, Main Board, Electronics, mPMT, Co-op

**History of Changes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Release No.** | **Date** | **Description** | **Authour(s)** |
| 1.0 | 21/12/2020 | Initial Release | L. Bidulka |
| 2.0 | 27/04/2021 | Updated Release | J. Cronin |
|  |  |  |  |
|  |  |  |  |

Contents

[1 Introduction 3](#_Toc59454523)

[2 Experiment 4](#_Toc59454524)

[3 Mainboard Electronics 5](#_Toc59454525)

[4 Project Status 7](#_Toc59454526)

[5 Resources 8](#_Toc59454527)

# 1 Introduction

This document is intended to act as an onboarding device for the May 2021 firmware developer working on the NuPRISM mPMT mainboard. This document will introduce you to the experiment and provide an overview of the mainboard, which is the hardware platform you will be working on. Next, it will describe the project status from the point of view of the TRIUMF firmware developer and try to recommend a path moving forwards. Additionally, there will be a section referencing and describing resources will be helpful for getting up and running.

If you have any further questions, your new colleagues in the edev team are an excellent resource. You can also contact either of the previous two co-ops, though the current team members would be a better choice.

Jacob Cronin at [jcronin97@live.ca](mailto:jcronin97@live.ca) (January 2021 – April 2021)

Luke Bidulka at [lukebidulka@gmail.com](mailto:lukebidulka@gmail.com) (September 2020 – December 2020)

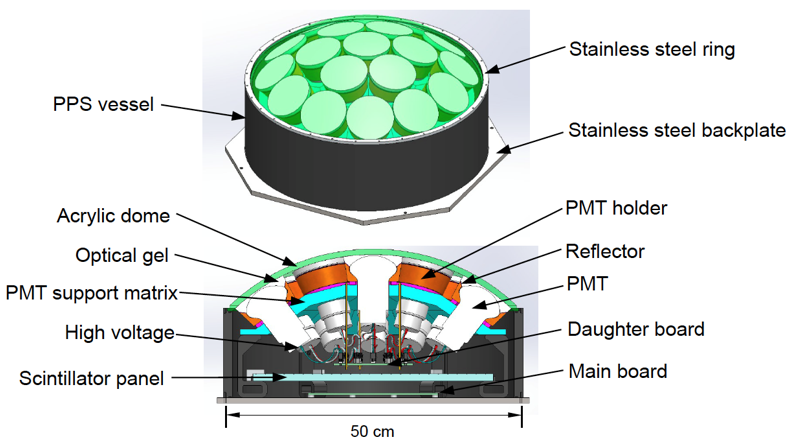
# 2 Experiment

The IWCD experiment will place a water Cherenkov detector in a deep vertical pit in Japan. Approximately 500 multi-photomultiplier tubes (mPMT) make up the Cherenkov detectors photo-sensors. The experiment setup is shown in figure 1-1. Each mPMT consists of 20 photomultiplier tubes connected to the main board electronics. The mPMT is shown in figure 1-2. A mPMT concentrator card is responsible for providing higher level control and monitoring of the mPMTs. The experiment will be observing controlled events generated by a neutrino beam and a LED-based calibration source. It will also observe more random events from cosmic-ray muons.

A picture containing table

Description automatically generated

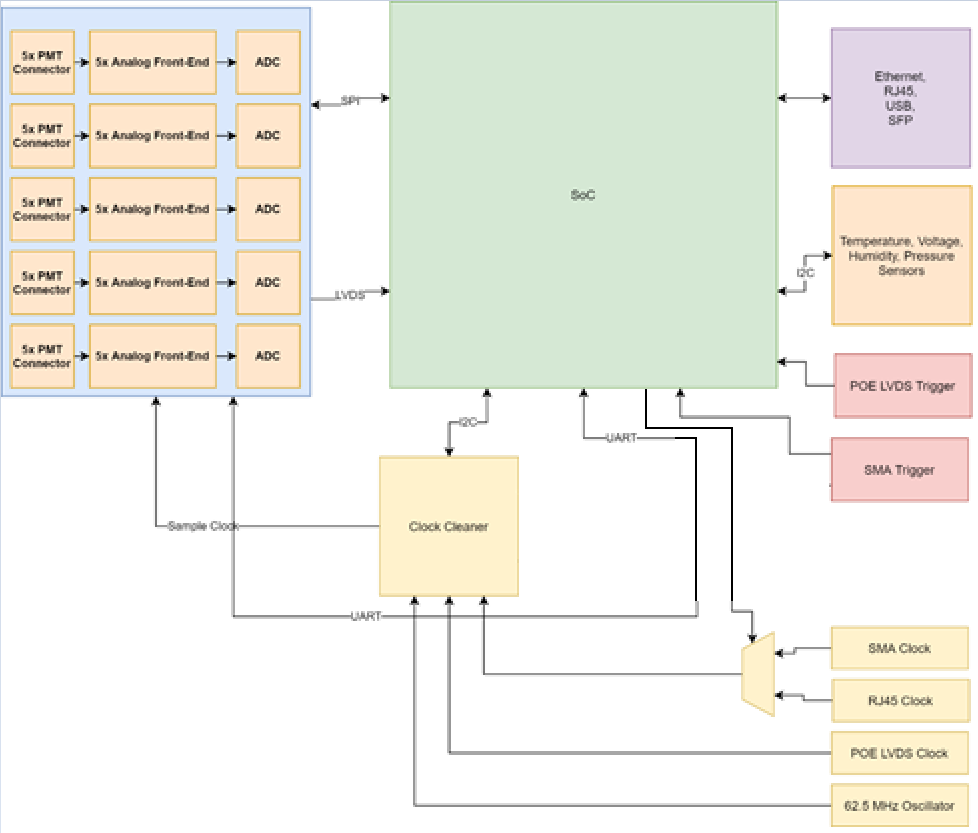
**Figure 1-1**: IWCD experiment setup



**Figure 1-2**: Diagram of the mPMT

# 3 Mainboard Electronics

A high-level hardware block diagram is shown in figure 1-3. The SoC deserializes and processes data from 5 quad-channel ADCs. The ADC clock is generated by a clock-cleaner which can derive its clock from a crystal oscillator, SMA input, RJ45 input, or POE input. For connectivity, the SoC provides connection to RJ45, USB, Ethernet, and SFP. For system monitoring, the SoC is connected by I2C to several temperature, humidity, pressure, and voltage sensors. In the newest revision of the board (rev 1a) the USB input provides a serial connection to the Enclustra SoM, previously this was routed through a MAX10 FPGA unit.



**Figure 1-3**: High-level hardware block diagram for mPMT main board

The clock cleaner takes its inputs from the SMA clock, RJ45 clock, POE clock, and a reference oscillator. The SMA clock and RJ45 clock are muxed and selected by the Enclustra. On reset/startup the clock cleaner will use the oscillator as a reference. If either of the other two inputs are detected as valid, it will switch its reference to the valid clock. If both inputs are valid, a user-settable priority bit is referenced. While the input and output clocks are locked, device GPIO is activated and can be monitored. The generated clock is distributed to the 5 ADCs and used as a sample clock.

The main processor currently in place is the Xilinx Zynq Ultrascale+ EG MPSoC. A block diagram depicting its internals is shown in figure 1-4. The MPSoC consists of a processing system (PS) and programmable logic (PL). The PL is the FPGA portion of the MPSoC and contains RAM blocks, DSP blocks, and high-speed connectivity blocks. The PS is a highly integrated system. Most notable it contains it a quad-core 64-bit application processing unit (APU), and dual-core 32-bit real-time processing unit (RPU). To manage the power consumption of this device, the platform management unit (PMU) can be used to suspend, wakeup, and power-down different components of the entire architecture. There are several ways to manage a wakeup of the APU. Some notable methods are by GPIO input or by request from the RPU.

A screenshot of a cell phone

Description automatically generated

**Figure 1-4**: Zynq Ultrascale+ EG MPSoC architecture diagram

For the incoming firmware developer, it is worth noting that the platform supports to hot-pluggable SoCs. The one you will be developing firmware on is the Enclustra XU1 which hosts Xilinx’s Zynq Ultrascale+ EG MPSoC. The other is the Enclustra SA2 which hosts Altera’s Cyclone V. Revision 0 of the mainboard was specifically designed for the SA2, but later versions will use the XU1. A Rev 1 board is in development right now, and you will likely hear all about it.

# 4 Project Status

The firmware portion of this project is undergoing 3 branches of development. The Altera branch, the Xilinx RTOS branch, and the Xilinx Petalinux branch.

The Altera branch is being worked on by Dr. Yair Linn and is the most developed branch. Current testing of the integrated mPMT is being done at TRIUMF using this project.

The Xilinx RTOS branch is in early stages of development. This branch is being worked on by the Polish colleagues. It is similar to the Xilinx Petalinux branch, and so I have used some of their code while they have used some of mine. You will likely continue sharing code with them and developing thing

There is also a Xilinx Bare Metal branch developed by a previous coop, Ryan Payne that was used to test board level function and run the deserializer HDL. While this branch is useful to quickly test low-level function and run altered deserializer HDL, it should not undergo major development. It has been useful for understanding some functionality and can be used a reference point for some code. This project and a guide to using it can be found here, you will need to get Gitlab access from Bryerton to access the repository:

<https://edev-group.triumf.ca/fw/exp/nuprism/xu1/rev0>

The Xilinx Petalinux development branch (most recent is rev1a-integration) is the branch created and maintained by TRIUMF edev co-ops. Petalinux is a Linux operating System (OS) developed by Xilinx for their boards and which we are evaluating for future use through this branch. Currently, this project runs a demo of MIDAS (data acquisition software developed at TRIUMF), tests a number of peripherals, and streams data from the PL to the PS using DMA hardware and associated drivers. It’s also important to realize that this repository will likely not host the polished project, think of it as a workspace for developing various subsystems on the Xilinx platforms, the work you do on this branch will probably remain as an executable application to demonstrate functionality, rather than integrating into the system. Due to this style, the repo can be confusing as some folders were used to test out features and may be depreciated, to get you started I suggest focusing only on the petalinux/nupism and Vivado\_NuPRISM/NuPRISM as they contain the software and hardware projects. Go through the most recent [Petalinux Project Setup Guide](https://github.com/nuPRISM/iwcd-xu1-petalinux-rev0-2020-01/tree/canada-develop/documents) to see the commands and tools you will use and get the project running on the board.

To brief you on the status of the co-op developers progress, myself and my predecessors have successfully added preliminary DMA hardware, ADC deserialization for all channels, GPIO support and other system management blocks for the programable logic design. The DMA is where you will likely begin your work as our next goal is connecting the ADC deserializers to the DMA blocks on a larger scale. As for the software component, you will find a completed ADC control app that uses the Linux SPI driver, a DMA test app, and various I2C test apps that correspond to different sensors, I suggest you review these as you will need to make additions to the I2C code.

**4.1 Next Steps**

A high-level list of tasks to focus on might include:

* Getting the deserializers to stream ADC data to the PS by connecting them through DMA
  + The Polish group claims to have done this successfully for a single DMA channel, it would be a good idea to review their branch to see their progress, but you will likely use the project completed by Dr. Linn’s capstone group.
* Add I2C control for sensors on the mainboard
  + This is nearly complete, I recommend reviewing the I2C related apps in the repo, all except eeprom-i2c and current-i2c are complete.
  + To create the sensor API’s, you will need to review the datasheet for each sensor to learn about their specific communication standards.
* Add UART control for PMT bases
* Add MIDAS control of ADC, clock cleaner, sensor
* Add MIDAS readout of ADC data, sensor data

The route I suggest you take is the following:

1. Read the background information from the resources section below so that you understand more about the toolchain, system, and technologies.
2. Run through the tutorial in later section to setup your board, Petalinux environment, run the feature limited MIDAS demo. This will help you get familiarized with Vivado, Petalinux, and MIDAS
3. (Extra, might be worth the time but is somewhat to the side of our work) Run through the project guide foundhere:

<https://edev-group.triumf.ca/masoudm/darkside-data-processor-project>

This project was developed by Masoud Mokhtari, another co-op student, for an experiment called Darkside. It uses petalinux and the same chip as us; you could use this project to better understand DMA and MIDAS. Modify the MIDAS front end to stream the DMA data over ethernet to your MIDAS host.

1. Consult with the team as to what the next steps are, you will likely start with I2C code or the DMA framework.

As a note:

The Gitlab repo has been where Ryan pushed code that was irrelevant to our Polish colleagues or the physicist. For the most part this code was developed as a learning exercise for him.

All main code relevant to the project that others may need to run should be placed on the NuPRISM GitHub located here:

<https://github.com/nuPRISM>

The XU1\_petalinux\_rev0 repo holds the 2020.1 Petalinux project, what you will primarily be working on. There are two branches being developed, one for Poland and one for Canada.

The XU1\_rev0 repo holds the 2018.3 standalone project.

The MAX10\_rev0 project holds the code which runs on the MAX10 (only if you use the rev 0 board)

For access to this code please contact Thomas Lidner or Bryerton Shaw.

# 5 Resources

**5.1 Mainboard Electronics**

Given Gitlab access, the schematics for the Rev1A mainboard are saved in the documents folder of your machine.

Additional schematics and documentation for the Enclustra module can be found here:

<https://download.enclustra.com/>

**5.2 Embedded Linux**

You will be working with Petalinux, a flavour of embedded Linux developed by Xilinx. It will be helpful to know some general concepts, [here](https://www.thirtythreeforty.net/posts/2019/08/mastering-embedded-linux-part-1-concepts/) is the first of a series which outlines some of the broad ideas/concepts/topics for embedded linux and its differences/similarities compared to desktop linux. [Here](https://thenewstack.io/the-major-components-of-an-embedded-linux-system/) is another short overview. You will also frequently be working with [drivers](https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch01.html) and [device file](https://opensource.com/article/16/11/managing-devices-linux) (link has links to further reading) interfaces.

**5.3 Vivado and Petalinux**

Vivado is the tool you will use to develop the fpga hardware design and Petalinux is the OS that runs on the PS. You will need to understand how to use Vivado and be familiar with its interface and build process and you will need to be familiar with Petalinux to develop the software needed for the project. As an intro to Vivado and Petalinux, Mohammad Sadri’s video series on ZYNQ and Petalinux are an excellent resource. I highly, highly recommend watching them as you go to understand how to use to Xilinx tools. The videos can be found [here](https://www.youtube.com/user/mamsadegh2/videos).

The [Petalinux 2020.1 user guide](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug1144-petalinux-tools-reference-guide.pdf) from Xilinx is a good reference for Petalinux.

To quickly set up an Ubuntu VM with Petalinux 2020.1, please follow the [Petalinux Project Setup Guide](https://github.com/nuPRISM/iwcd-xu1-petalinux-rev0-2020-01/tree/canada-develop/documents) mentioned above.

**5.4 Deserializer and Clocking Scheme**

The ADCs use DDR 2-wire serialization of 12 bit sampled at 125Msps. Special FPGA fabric is used to deserialize this data. The current deserializer HDL is based off another project done by the Stanford Linear Accelerator (SLAC). That project can be found [here](https://github.com/slaclab/amc-carrier-core/blob/master/AppHardware/RtmRfInterlock/core/Ad9229Core.vhd).

The deserializer components inside of the FPGA logic take a fair bit of reading through documentation to understand. To summarize what Ryan learnt in his research, he wrote a report, which can be found with this package under:

\Supporting\_Documents\XU1\_NuPRISM\_Derserializer\_and\_Clocking\_v1\_0

I highly recommend reading this document and before attempting to make any changes to the current deserializer.

**5.5 DMA**

The addition of DMA took up a large chunk of my time. I highly recommend reading about DMA and understanding how to use the drivers before playing around with it as it can be tricky.

Start with [this](https://www.xilinx.com/video/soc/linux-dma-from-user-space.html) video from Xilinx to get an understanding of the interactions DMA involves. Much of what is discussed can be found in the dma-proxy driver, which has already been integrated into the project.

[This](https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842418/Linux+DMA+From+User+Space) is the Xilinx dma-proxy-test setup I started from, and is worth a look.

[Here](https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf) is the Xilinx documentation for their AXI DMA IP, for reference.

[Masoud’s Darkside project](https://edev-group.triumf.ca/masoudm/darkside-data-processor-project/-/tree/master/), which uses DMA, could be used for reference as well.

A [2020.1 version](https://edev-group.triumf.ca/fw/exp/nuprism/xu1/darkside-2020.1) of Masoud’s project is also available.

**5.6 MIDAS**

MIDAS is a front-end data acquisition software that will be used at TRIUMF (and potentially in Poland) to test the mPMT. Its [wiki](https://midas.triumf.ca/MidasWiki/index.php/Main_Page) has a lot of info about how it works.

The wiki also contains a setup article, but I recommend you follow the [Petalinux Project Starter Guide](https://github.com/nuPRISM/iwcd-xu1-petalinux-rev0-2020-01/tree/canada-develop/documents) to set it up instead, as you do not need to access all the features present in the software.

**5.7 SPI**

SPI uses a combination of the Linux device driver 3-wire interface and GPIO to communicate with the ADC’s. It would be worthwhile to review the Linux documentation and the completed apps to understand how SPI and I2C function using their respective message structures and the kernel space function ioctl(), but you mainly need to understand how the bit flags function. The Enclustra uses the SPI 3-wire mode with a single chip select wire which is logical OR’d with 5 GPIO pins corresponding to each ADC, essentially the Linux kernel sees only one SPI device and GPIO handles chip select

**5.8 I2C**

I2C operates similarly to SPI in that in user space, you need to populate a message structure, and pass the struct to ioctl() to execute the transfer. When working with the sensor units you will need to modify different flag bits as specified each datasheet to correctly execute the transaction.