Prepared for submission to JINST

Very Large Volume Neutrino Telescope Conference
18-21 of May 2021
Valencia (Online)

Embedded software developments in KM3NeT phase I

V. van Beveren,[a,1] **D. Real**[b] **T. Chiarusi**[c] **D. Calvo**[b] **S. Mastroianni**[d] **P. Musico**[e] **G. Pellegrini**[c] **P. Jansweijer**[a] **S. Colognes**[f] **C. Bozza**[g] **F. Filippini**[c] **C. Nicolau**[h] **A. Díaz**[i] **on behalf of the KM3NeT collaboration**

[a] *Nikhef, National Institute for Subatomic Physics, PO Box 41882, Amsterdam, 1009 DB Netherlands*
[b] *IFIC - Instituto de Física Corpuscular (CSIC - Universitat de València), c/Catedrático José Beltrán, 2, 46980 Paterna, Valencia, Spain*
[c] *INFN, Sezione di Bologna, v.le C. Berti-Pichat, 6/2, Bologna, 40127 Italy*
[d] *INFN, Sezione di Napoli, Complesso Universitario di Monte S. Angelo, Via Cintia ed. G, Napoli, 80126 Italy*
[e] *INFN, Sezione di Genova, Via Dodecaneso 33, Genova, 16146 Italy*
[f] *APC, Université Paris Diderot, CNRS/IN2P3, CEA/IRFU, Observatoire de Paris, Sorbonne Paris Cité, 75205 Paris, France*
[g] *Università di Salerno e INFN Gruppo Collegato di Salerno, Dipartimento di Fisica, Via Giovanni Paolo II 132, Fisciano, 84084 Italy*
[h] *INFN, Sezione di Roma, Piazzale Aldo Moro 2, Roma, 00185 Italy*
[i] *INFN, Sezione di Catania, Via Santa Sofia 64, Catania, 95123 Italy*

E-mail: v.van.beveren@nikhef.nl

Abstract: The KM3NeT Collaboration has already produced more than one thousand acquisition boards, used for building two deep-sea neutrino detectors at the bottom of the Mediterranean Sea, with the aim of instrumenting a volume of several cubic kilometers with light sensors to detect the Cherenkov radiation produced in neutrino interactions. The the so-called Digital Optical Modules, house the PMTs and the acquisition and control electronics of the module, the Central Logic Board, which includes a Xilinx FPGA and embedded soft processor. The present work presents the architecture and functionalities of the software embedded in the soft processor of the Central Logic Board.

Keywords: Neutrino detectors, Software Engineering

[1] Corresponding author.

Contents

1 Embedded software in the KM3NeT detector

The KM3NeT detector currently being constructed in the Mediterranean Sea [1] consists of various modules which need to be actively controlled. For this purpose the Central Logic Board (CLB) has been designed. The CLB is an electronics board with a Xilinx Kintex 7 Field Programmable Gate Array (FPGA) as central controller. Additionally, it contains an optical transceiver for network connectivity and, various sensors and connectors for attaching data acquisition hardware and instrumentation. The design of the CLB is primarily driven by the need to fit into a KM3NeT Digital Optical Module (DOM), but it is not specific to it. The CLB is also used as the main controller in Detector Unit Base (DU-Base), and the Calibration Unit Base (CU-Base). Depending in which module the CLB is located different firmware can be loaded, specific to its purpose.

The FPGA firmware contains all digital logic required to perform its function and is a composition of gateware and embedded software. Gateware is the logic present inside the programmable logic of the FPGA fabric and generally written in a hardware description language, while the embedded software is written in a programming language and running on a processor. The gateware contains three DAQ units: The Time to Digital Converter (TDC) receiving pulses from the Photo Multiplier Tubes (PMTs) and conditioning hardware, the Audio Engineering Society (AES)-standard encoder receiving data from the acoustic sensor (a hydrophone), and finally MONitoring, which periodically generate run-time performance information. The Data AcQuisition (DAQ) modules stream data into the HWStateMachine, which subsequently annotates and frames the data and finally forwards it to the IPMux. The IPMux wraps the annotated data into User Datagram Protocol (UDP) packets and dispatches them through the WhiteRabbit Precision Timing Protocol (PTP)-core Media Access Control (MAC) over the seafloor network to the shore data processing facility. In addition to MAC-functions, the WhiteRabbit core is responsible for providing a detector-wide synchronized clock[2].

The gateware contains two LatticeMico32 (LM32) micro-processors[6], coded in Hardware Description Language (HDL): the WhiteRabbit processor for timing control, and the CLB processor for DAQ control and sensor readout. The IPMux acts as a gateway to the detector network for the CLB LM32 software, allowing both reception and transmission of Ethernet packets. This channel

is used for the slow-control and for network functions such as Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP) and Internet Control Message Protocol (ICMP). This work is mostly focused on the CLB LM32 processor, which runs the KM3NeT specific software. For more information about electronics and readout, the reader is referred to [3].

The KM3NeT telescope is a heterogeneous detector and there is not one firmware which can run on all its modules. The firmware comprises of three types of binaries:

- The hardware specific FPGA bit-file (CLB v2 or CLB v4)
- The WhiteRabbit embedded software for either a KM3NeT custom version (denoted as 'broadcast') [4], or standard WhiteRabbit.
- The CLB application software which can be either DOM, DU-Base, CU-Base or Golden, for start-up.

When building the firmware, different combinations of such three-types of binaries are merged into a single FPGA firmware image, creating multiple firmware products, each specific to an application and hardware version. The CLB can store up to four different firmware images, but only two are needed: The golden image and the run-time image. Though most firmware images are for a specific module of the detector, the golden image is generic to all modules. It is the first firmware image loaded after powering the detector and contains only minimal hardware initialization. Its purpose is to boot the actual run-time application, but also to allow firmware updates in case the application requires updates, or is not operational. The run-time image is specific to the module the CLB is contained in.

The First Generation (FG) of the KM3NeT firmware has been in development since 2012 and was deployed in the beginning of 2016. Since 2019 development of the Next Generation (NG) Firmware has started, containing many improvements with respect to the first generation, both in project semantics and program architecture. Project improvements include usage of modern development methodologies and tools such as the usage GIT with sub-modules, Continuous Integration and Docker containers, for a consistent and reusable build environment. Program architecture improvements include a refined software state-machine, integration with of the current production version of WhiteRabbit, and improved error handling. Some of these features which will be covered in more detail ahead in the text.

2 Software Architecture

The KM3NeT embedded software is a bare metal application, using no preemptive operating system. When compiled the total program, including runtime memory, takes up less than 256 KB. The application is mostly written in C with some assembly for start-up and interrupt handling. The software is split into two main layers: the system-software layer and the application layer. For each kind of firmware image the system-software layer is the same. This layer contains OS-like features, such as a simple cooperative multitasking scheduler, a firmware update unit, various peripheral drivers, a UDP based network stack and support utilities for logging and error handling. The application layer contains a software state-machine and a number of subsystems, each responsible for a different aspect of the application.

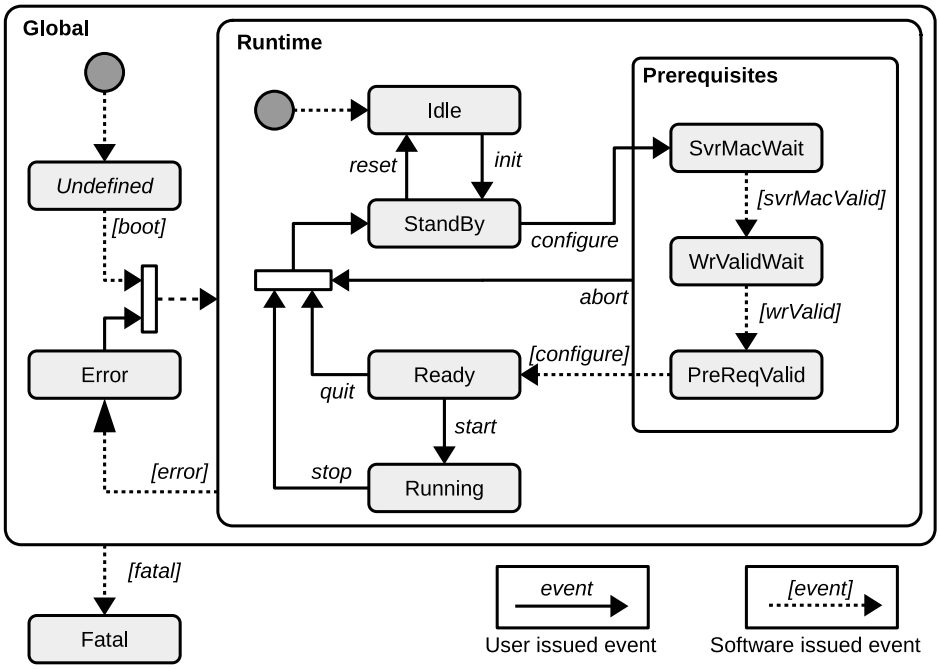

Figure 1: The Next Generation Software State Machine

The state-machine, shown in Figure 1, drives state of the application and outside of the control of state-machine little application-level code is run. Implementing the application software as a state-machine obligates the program to always be in a clear and consistent state. The state-machine can be moved by issuing events, generally driven over slow-control by the Control Unit [5]. Some events are issued autonomously by the embedded software, such as when an error occurs or during system start-up. The application code can be attached to state-machine transitions or to a timer, which calls the code periodically.

As previously mentioned, application code is grouped into *subsystems*. A subsystem is a unit of code and data responsible for a specific aspect of CLB operation. It controls hardware peripherals, through a driver abstraction, associated with that aspect. The following subsystems are known to the CLB software:

- **System** – Application function not specific to other subsystems
- **Optics** – Control of PMTs and TDCs (only in DOM firmware)
- **Acoustics** – Acoustic sensor and AES control
- **Instrumentation** – Sensor readout, generally over the i2c bus (Temperature, humidity, etc.)
- **Networking** – IPMux control and WhiteRabbit monitoring
- **Base** – DU-Base container control (only in DU-Base firmware).

By registering C-functions to state transitions a subsystem can control hardware at specific points in the state-machine graph. For example, the *start* event moves the state-machine from Ready to Running. In this transition the data-acquisition hardware is enabled to start data taking.

The CLB slow control from remote is implemented by means of a custom protocol, on top of UDP. The Slow-control protocol consists of three layers. The highest layer is called the Message

layer and binds to C-functions at the application level. Messages have a type, e.g. 'retrieve firmware version' or 'state-machine event', but also a class, being either Command, Reply, Event or Error. The combination of type and class specifies the content format and interpretation of the message payload. For example, the 'retrieve firmware version' type has no parameters as a Command, but contains a string with the firmware version as Reply. Slow-control messages are the primary method for remote control and has functions for moving the software state-machine, request or set process-variables and many others. Messages are bundled together at the Message Container Format (MCF) layer, binding multiple messages into a single payload for efficiency. The lowest slow-control layer is Simple Retransmission Protocol (SRP) and is responsible for transmission control. It implements a simple packet-based re-transmission scheme where packet contains an identifier ordinal which must be acknowledged within a specific time window and is otherwise re-transmitted.

The primary way to set configuration values or get sensor data from remote is by using the process-variable mechanism. Process-variables are defined in a json5-formatted[7] variable dictionary file and using a code generation tool various files representing the variable dictionary for different programming languages can be generated. For the embedded software C-code is generated for the defined variables, while for remote Java-code is generated. For the embedded software all process-variables are exposed as plain-C variables and are accessible for read and write but from remote access may be restricted during a run, or in the case of sensor data always be read-only.

3 Build environment

The build environment consists of four GIT repositories:

- **clb-sw** - Builds program binary for different applications (DOM, DU-Base, CU-Base and golden) and hardware versions (CLB v2 and v4). It executes tests and builds documentation.
- **wrpc-sw** - A special branch of the WhiteRabbit PTP core is maintained and the program binaries for the WhiteRabbit PTP core for both the standard and the KM3NeT specific versions is build from this.
- **clb-hdl** - Builds the FPGA image binaries (gateware) for different hardware versions (CLB v2 and v4). This project also uses parts of the WhiteRabbit gateware respository to build the WhiteRabbit PTP core logic.
- **clb** - Includes all of the above projects as GIT sub-modules. It contains the super-build script which builds all sub-projects and merges all generates binaries into a number of application and hardware specific firmware images.

CMake is used as the primary build tool. The entire project can be build inside Gitlab-CI, using a KM3NeT specific docker container.

4 Conclusions

The First Generation KM3NeT embedded software has been in development since 2012 and works reliably on the seafloor since 2016 but is now scheduled to be superseded by the Next Generation KM3NeT firmware. The NG firmware has an improved software design for consistent control, contains more reliability features for robust operation and the project now uses modern development tools for reliable and consistent builds. The NG firmware is on track to be deployed this fall 2021.

References

[1] Adrián-Martínez, S et al., *Letter of intent for KM3NeT 2.0*, *J. Phys. G: Nuclear and Particle Physics* **43** (2016)

[2] Daniluk, G and Włostowski, T, *White Rabbit: Sub-nanosecond synchronization for embedded system*, *PTTI* (2011)

[3] Aiello, S et al., *KM3NeT front-end and readout electronics system: hardware, firmware, and software*, *JATIS* , **5(4)** (2019) arXiv:1907.06453v2

[4] Pellegrino, C and Chiarusi, T on behalf of the KM3NeT Collaboration. *The Trigger and Data Acquisition System for the KM3NeT neutrino telescope EPJ Web of Conferences* **116, 05005** (2016)

[5] Aiello, S et al., *The Control Unit of the KM3NeT Data Acquisition System*, *CPC*, **256** (2020) arXiv:1910.00112v1

[6] LatticeMico32 Open, Free 32-Bit Soft Processor. (2021, July 1). retrieved from https://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/LatticeMico32.aspx

[7] JSON5 - JSON for Humans. (2021, July 1). retrieved from https://json5.org/