





Publication Year	2022
Acceptance in OA @INAF	2023-12-29T14:45:43Z
Title	GAMMA-FLASH Software Design Document of the Data Acquisition System
Authors	ABOUDAN, ALESSIO; ADDIS, ANTONIO; BULGARELLI, ANDREA; VIRGILLI, Enrico
Handle	http://hdl.handle.net/20.500.12386/34499
Number	GAMMA-FLASH-001-SDD

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	1/27

GAMMA-FLASH

Software Design Document of the Data Acquisition System

Prepared by:	Name:	A. Aboudan, A Addis, A. Bulgarelli E. Virgilli	Date:	May 11, 2022
Approved by:	Name:	A. Bulgarelli	Date:	May 11, 2022

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	2/27

Main Authors: A. Aboudan, A. Addis

Contributor Authors: A. Bulgarelli




		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	3/27

Table of Contents


1. Introduction	6
1.1. Purpose	6
1.2. Scope	7
1.3. Content	7
1.4. Abbreviations and acronyms	7
2. Applicable and reference documents	8
2.1. Applicable documents	8
2.2. Reference documents	8
3. System overview	9
3.1. Overall Description	9
3.2. System Context	9
3.3. Operational Modes	10
4. System design	12
4.1. Decomposition description	12
4.2. Physical description	14
4.3. DAM FW and ASW description	14
4.3.1. RedPitaya overview	15
4.3.2. ASW description	16
4.3.2.1. Internal structure	16
4.3.2.2. Boot sequence	17
4.3.2.3. ASW Configuration	18
4.3.3. DAM client	19
4.4. MCC ASW software description	19
4.4.1. DAM Server	20
4.4.2. GAMMA-FLASH pipeline	20
4.4.2.1. Configuration file	20
4.4.2.2. Class Diagram: pipelines	22
4.4.2.3. Class Diagram: datasource	22
4.4.2.4. Class diagram: Output Handlers	23
4.4.2.5. Class diagram: Utils	24
4.5. GAMMA-FLASH quick-look GUI	25
5. Data model	26
6. Deployment view	27

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	4/27

INDEX OF FIGURES & TABLES

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	5/27

Document History		
Version	Date	Modification
1.0	Nov 5, 2020	
1.1	May 11, 2022	Updates on new release for Summer 2022 campaign acquisition

GAMMA-FLASH							
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	6/27

1. Introduction

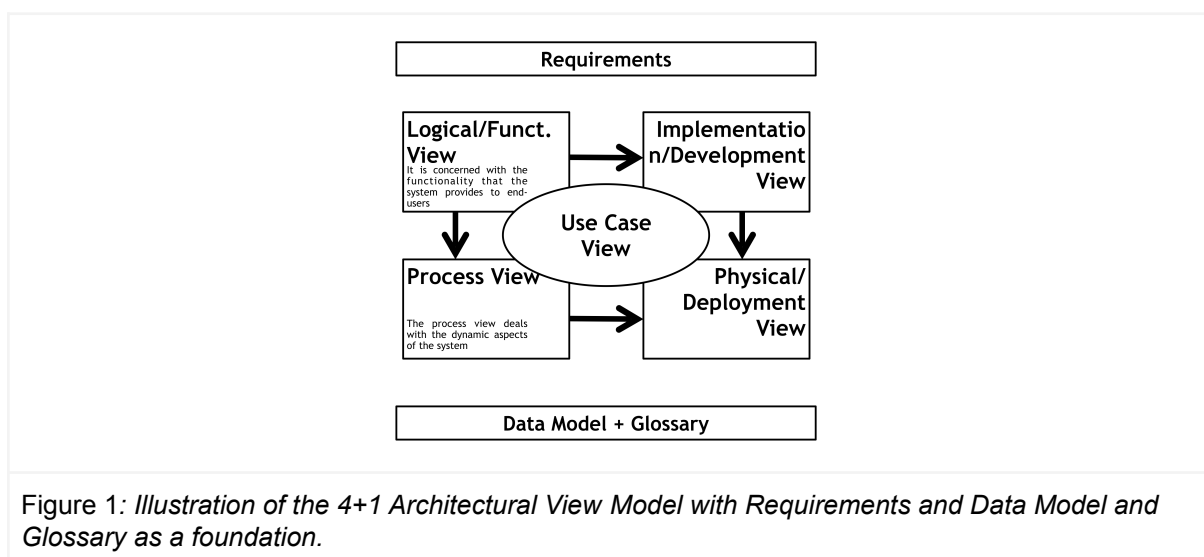
1.1. Purpose


The present document defines and describes the software architecture of the Data Acquisition and Control System (DACS) of the GAMMA-FLASH project.

The intended audience of this document are the potential users of the GAMMA-FLASH project, systems engineers, instrument scientists, designers, developers, testers (either unit or integration), and any contractor involved in the GAMMA-FLASH project who has in charge of the production of any sub-system which interfaces the DACS.

The architecture is defined by looking at the system from different viewpoints and is then illustrated through different views (see Fig. 1):

- 1) **Logical/Functional View:** a functional decomposition of the system with the description of the global information flow (based on the analysis of Use Cases and Data Models).
- 2) **Process View:** deals with the dynamic aspect of the system.
- 3) **Implementation/Development View:** represents the detailed design of the implemented system.
- 4) **Physical/Deployment View:** The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. The physical view is more concerned with the physical layer of the system, the deployment view with the allocation of computing resources on physical nodes;
- 5) **Use Case View:** A use case is a list of actions or event steps typically defining the interactions between an actor and a system to achieve a goal. The actors can be the user or other systems. In this context, UCs specify directly functional requirements, and each UC constitutes a functional specification.



		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	7/27

1.2. Scope

The DACS is a hardware/software system that provides data acquisition, storage, online data processing and reduction, quick look both for gamma and neutron detectors of the GAMMA-FLASH project.

1.3. Content

Section 2 provides applicable and reference documents.

Section 3 provides an overview of the overall GAMMA-FLASH DACS system.

Section 4 provides a detailed description of the DACS system.

Section 5 provides an overview of the data model.

Section 6 describes the technical infrastructure of the system, i.e. how the system is deployed.

1.4. Abbreviations and acronyms

ASW: Application Software


DAM: Data Acquisition Module

FW: Firmware

MCC: Main Control Computer

RP: RedPitaya

OS: Operating System

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	8/27

2. Applicable and reference documents

2.1. Applicable documents


[AD1] A. Aboudan, GAMMA-FLASH System and Software Requirements Specification of the Data Acquisition and Control System, GAMMA-FLASH-001-SRS, issue 1.0

[AD2] A. Aboudan, A. Addis, GAMMA-FLASH DACS Software Interface Control Document, GAMMA-FLASH-002-ICD, issue 1.1

2.2. Reference documents

The following reference documents provide background information as to the present specification and are not considered applicable to the present specification, in part or in full.

[RD1]

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	9/27

3. System overview

3.1. Overall Description

The DACS is designed to configure detectors, to collect the data (waveforms) from one or more different detectors of both neutron and gamma-ray events and provides a quick-look of the data and of the system health during the acquisition. Each detector can be configured and controlled independently from the others, the events detection, the waveforms acquisition, and the processing are performed automatically.

All the data acquired are processed online and the results are stored on the DACS and can be monitored on-site or remotely.

3.2. System Context

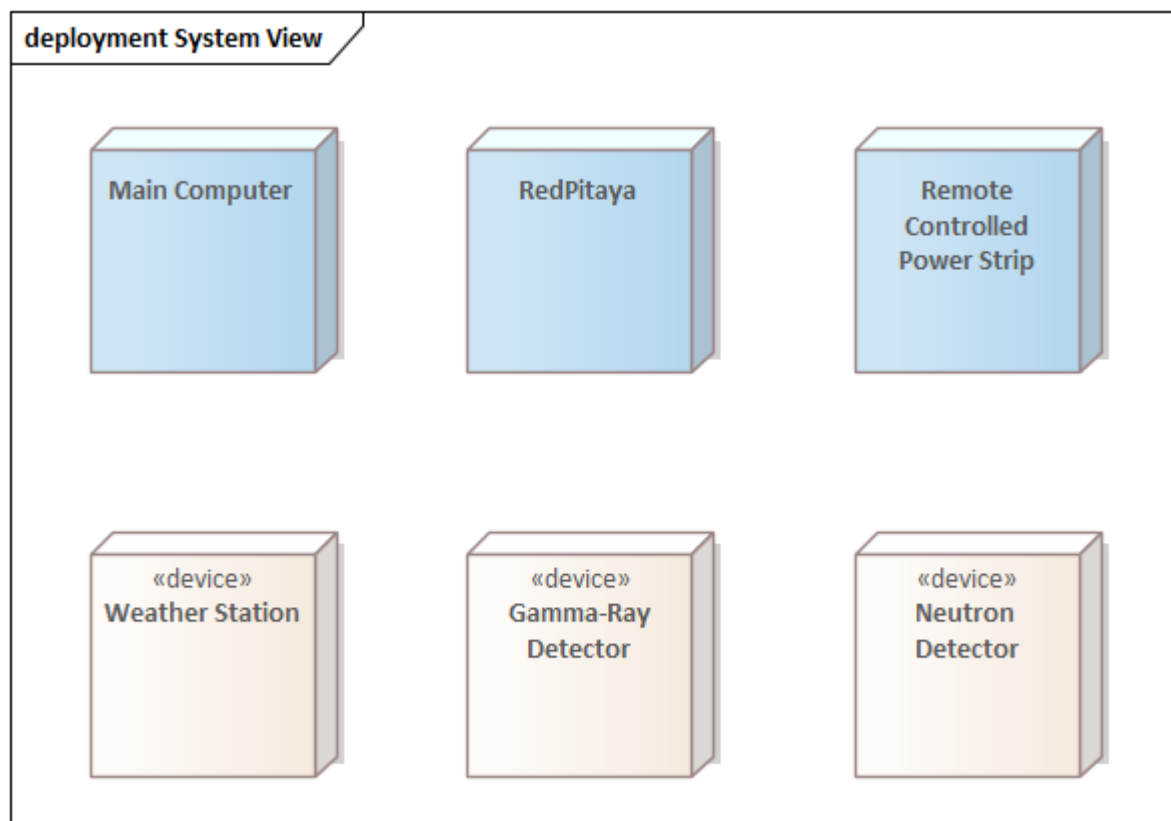



Figure 2: system context

Figure 2 shows the deployment system view, it contains the following nodes:

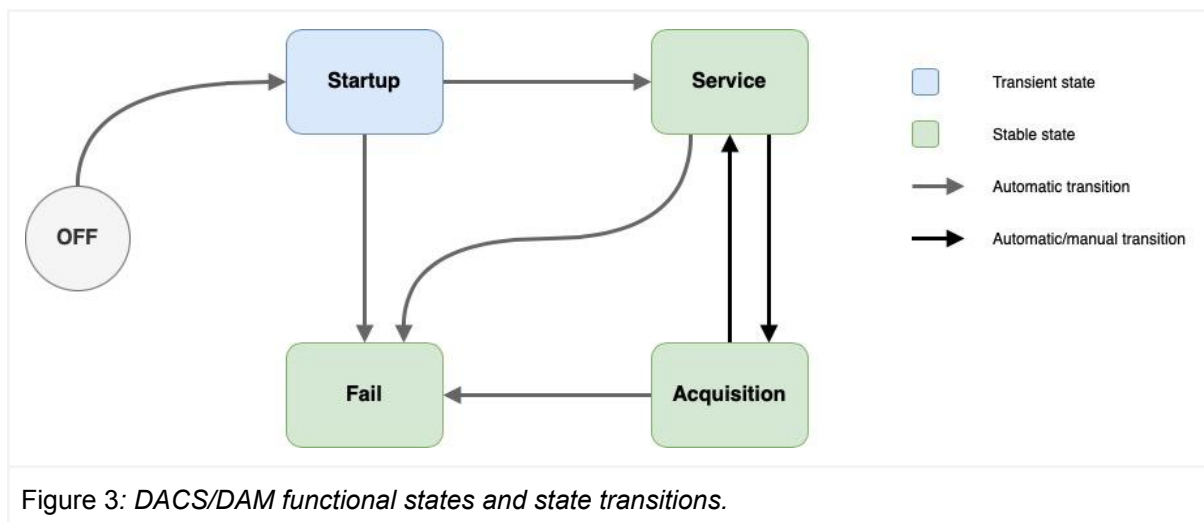
- Main Computer is a high performance machine where collected data are stored and processed. The MCC is connected with the external world using an Internet connection for data transfer and remote control.

GAMMA-FLASH							
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	10/27

- Gamma-Ray and Neutron Detector are photomultiplier tubes for Gamma and neutron events respectively
- RedPitaya is a Linux-FPGA board for acquiring the events detected by Gamma-Ray and Neutron Detector
- The Remote controlled power strip is used to power up the system. With its integrated network card and a web server it is possible to control the startup session remotely
- The weather station monitors the environmental temperature and triggers alerts when a specific threshold is reached. It is also used at the beginning of the startup session to decide if the system can be turned on.

3.3. Operational Modes

The operational modes foreseen by the DACS reflect the operational modes of each DAM described in [AD1] Sect. 4. and are shown in the following Figure 3.



Startup:


When powered ON each DAM performs the boot and then automatically executes the Startup procedure to check both the HW and SW status.

Service:

After the completion of the Startup procedure, the DAM enters the Service mode. In this mode, it is possible to:

- Access, check, download the scientific data stored onboard;
- Update and dump the data acquisition configuration parameters;
- Control and monitor detector PE;
- Start the data acquisition;

Acquisition:


		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	11/27

In this operational mode, the DAM is able to detect an event on the connected detector and store on-board the acquired waveform. Each waveform is properly time-tagged and .

After the completion of the event acquisition, new acquisition can be performed (continuous mode) or the DAM goes back into Service mode (single shot).

Fail:

At the Startup, the DAM performs an integrity check and can enter Fail mode in case of problems. During both Service and Acquisition modes, the DAM monitors the detector PE, in case of repeated OOL or other internal failures, the DAM enters Fail mode.

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	12/27

4. System design

This section describes the the Implementation/Development View, i.e. represents the detailed design of the implemented system and provides a description of

1. the software components, constituting the software item;
2. the relationship between the software components;
3. a summary of the purpose of each software component.

4.1. Decomposition description

The software components are summarized in this section and provide the Implementation View of the system.

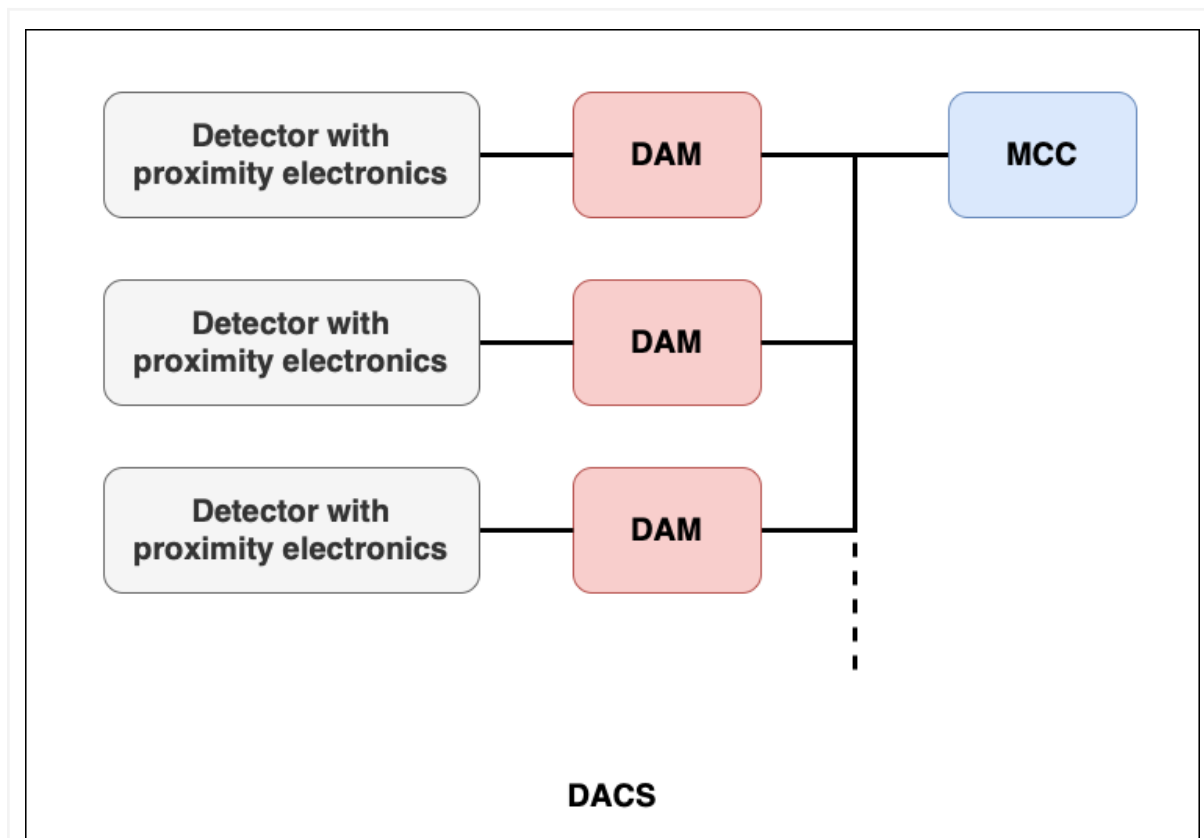



Figure 4: *DACS component decomposition*

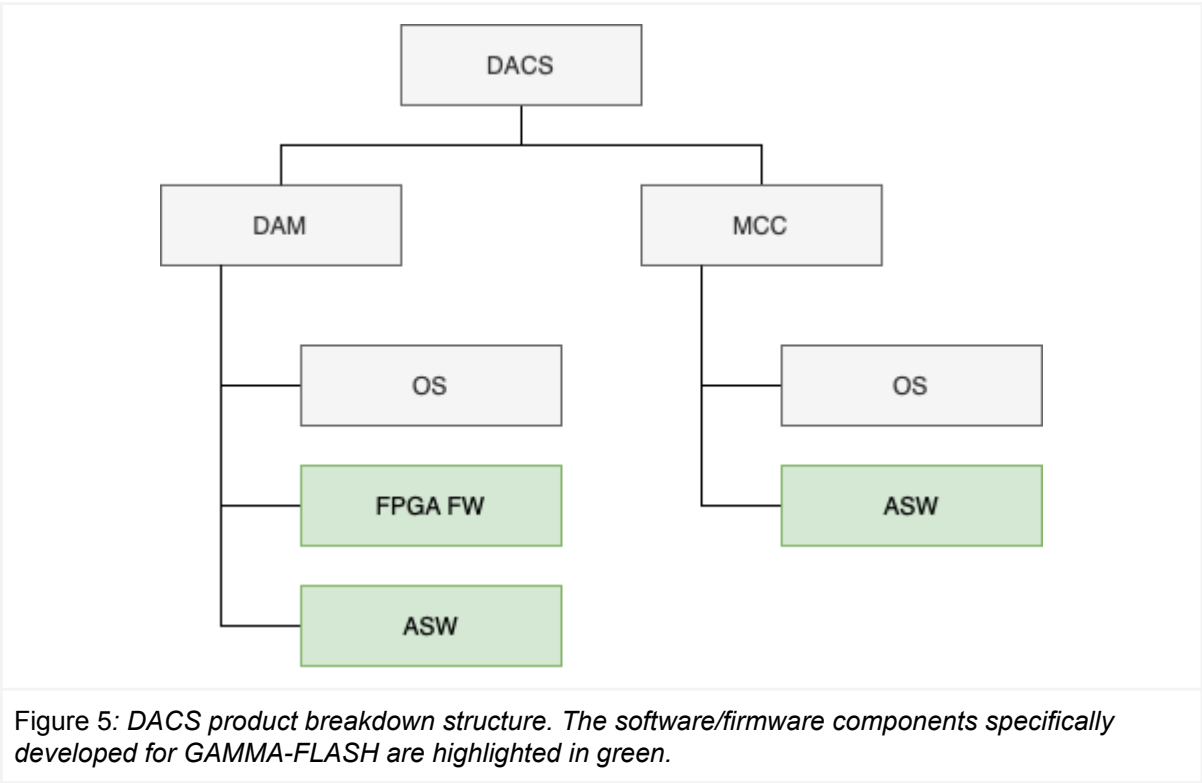
The DACS (Figure 4) is designed to be modular and host a variable number of detectors. Each detector is connected to a:

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	13/27

- Proximity Electronics (PE): containing a voltage booster, a charge sensitive amplifier, and the hardware needed to power on/off the detector monitoring its temperature and the supplied voltage.
- Data Acquisition Module (DAM): each DAM contains a complete data acquisition chain made up of an ADC, an event triggering logic, and a microcontroller used to handle the data acquisition. Each unit has some local data processing and storage capabilities.
- The DAMs are then connected to the Main Control Computer (MCC) that is in charge of controlling the overall data acquisition process providing data storage, online data processing, quick-look capabilities, and a unique interface for remote control of the experiment.


Ethernet connections are used between each DAM, the MCC, and the external world. In particular, the communication between the MCC and each DAM is based on a proprietary protocol described in [AD2].

The following component breakdown emphasizes the relationships among the components. It should define the system using one or more of the identity, type, purpose, dependency and resource parts of the component description.



This DACS software components can be decomposed as follows:

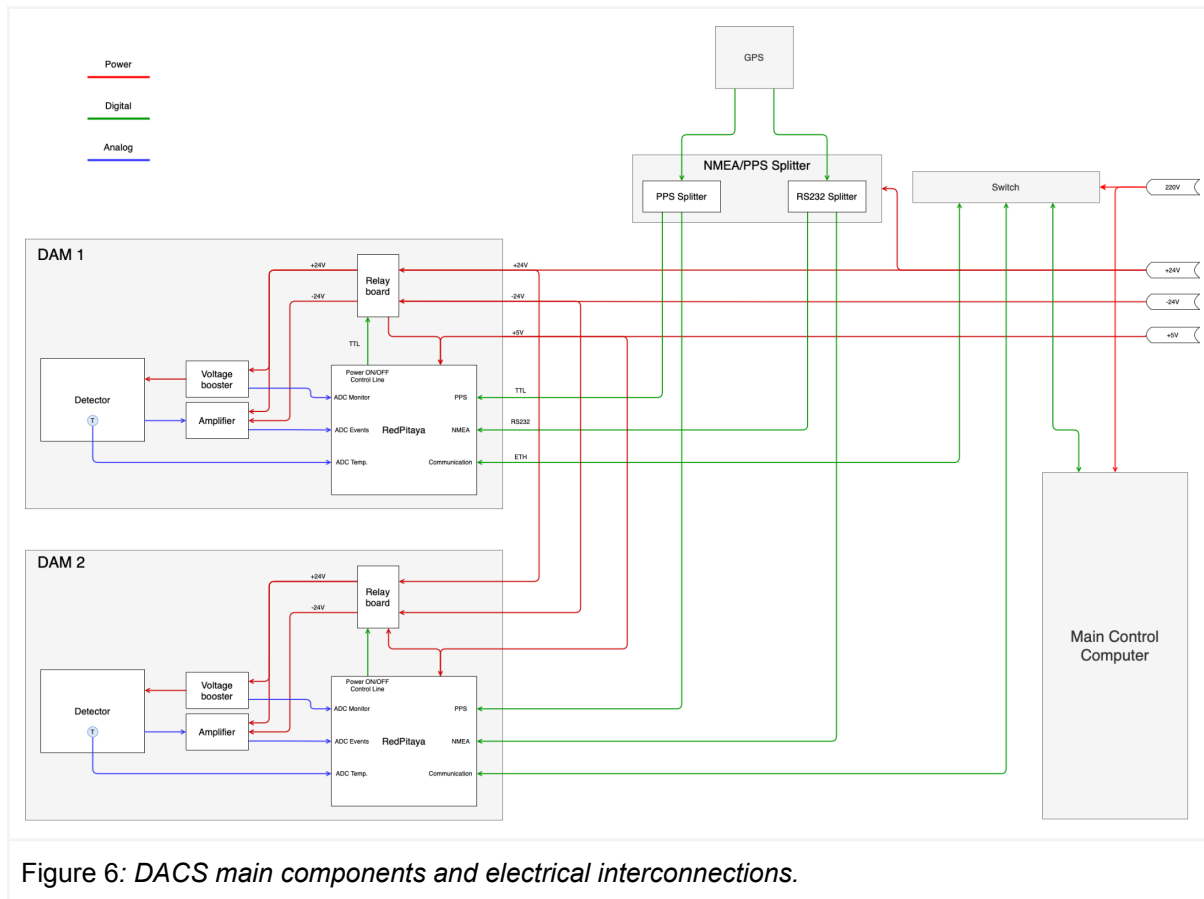
- DAM
 - OS: default Linux distribution provided by the HW supplier.

GAMMA-FLASH							
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	14/27


- FPGA FW: the FPGA FW implementing the ADC management and the trigger functions.
- ASW: controlling the data acquisition and managing the interface with the MCC.
- MCC
 - OS: default Linux distribution provided by the HW supplier.
 - ASW: managing the interfaces with each DAM and providing data processing, storage, quick look and remote monitoring and configuration.

4.2. Physical description

The main electrical connections between DACS subsystems are shown in Figure 6 and belong to three different classes: power, digital (e.g. TTL, RS232, or Ethernet connections), and analog signals used to monitor the detectors and the PE.

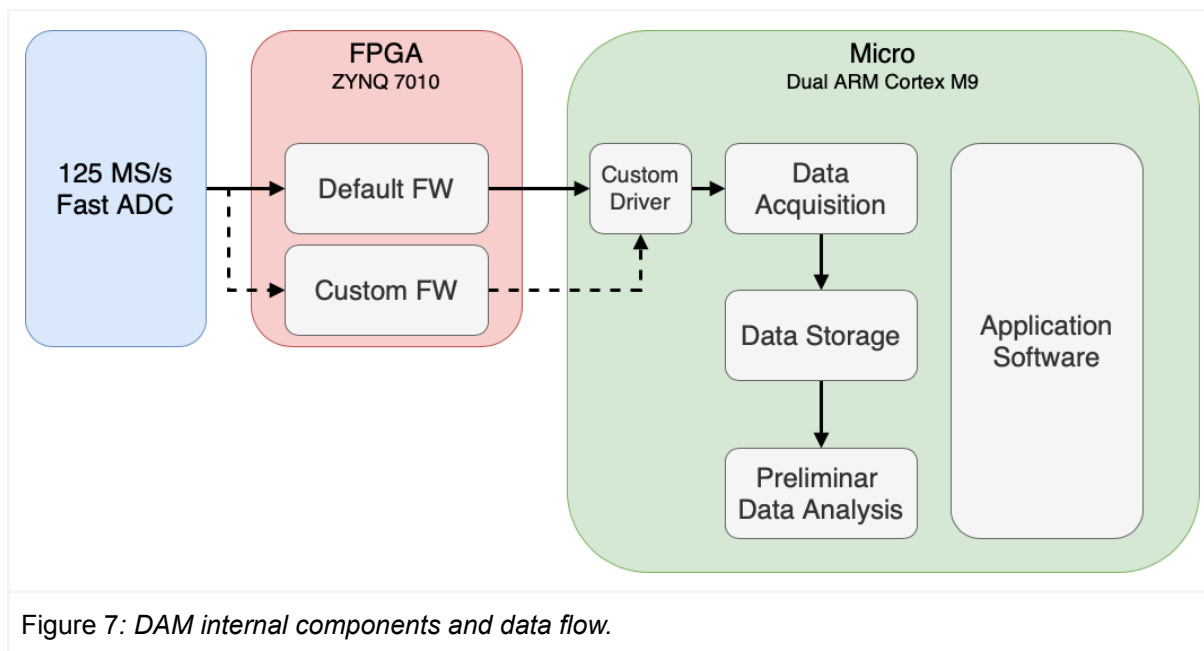


4.3. DAM FW and ASW description

GAMMA-FLASH							
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	15/27

4.3.1. RedPitaya overview

The DAM software is made up of two different parts, the FPGA FW and the ASW both running on the RedPitaya board. The RP board is equipped with a 125 MS/s fast ADC connected with a programmable FPGA. The FPGA is connected with a dual ARM Microcontroller (MC) running Linux OS (Figure 7).



The FPGA FW completely defines the functions implemented in the FPGA side of the board. Two different versions of such FW exist:

- Default FW: provided by the supplier of the board contains the DAC control logic and the event triggering logic, the logic used to handle UART and GPIOs plus several other general purpose functions (signal generation, logic analyzer ecc...). This is the version of the FW currently used for GAMMA-FLASH.
- Custom FW: this FW is experimental, it contains the DAC control logic, the event triggering logic, the logic used to handle UART and GPIOs, the GPS processing logic, the DMA data transfer logic to move the waveforms to the MC, and in the future it may include waveform processing algorithms.

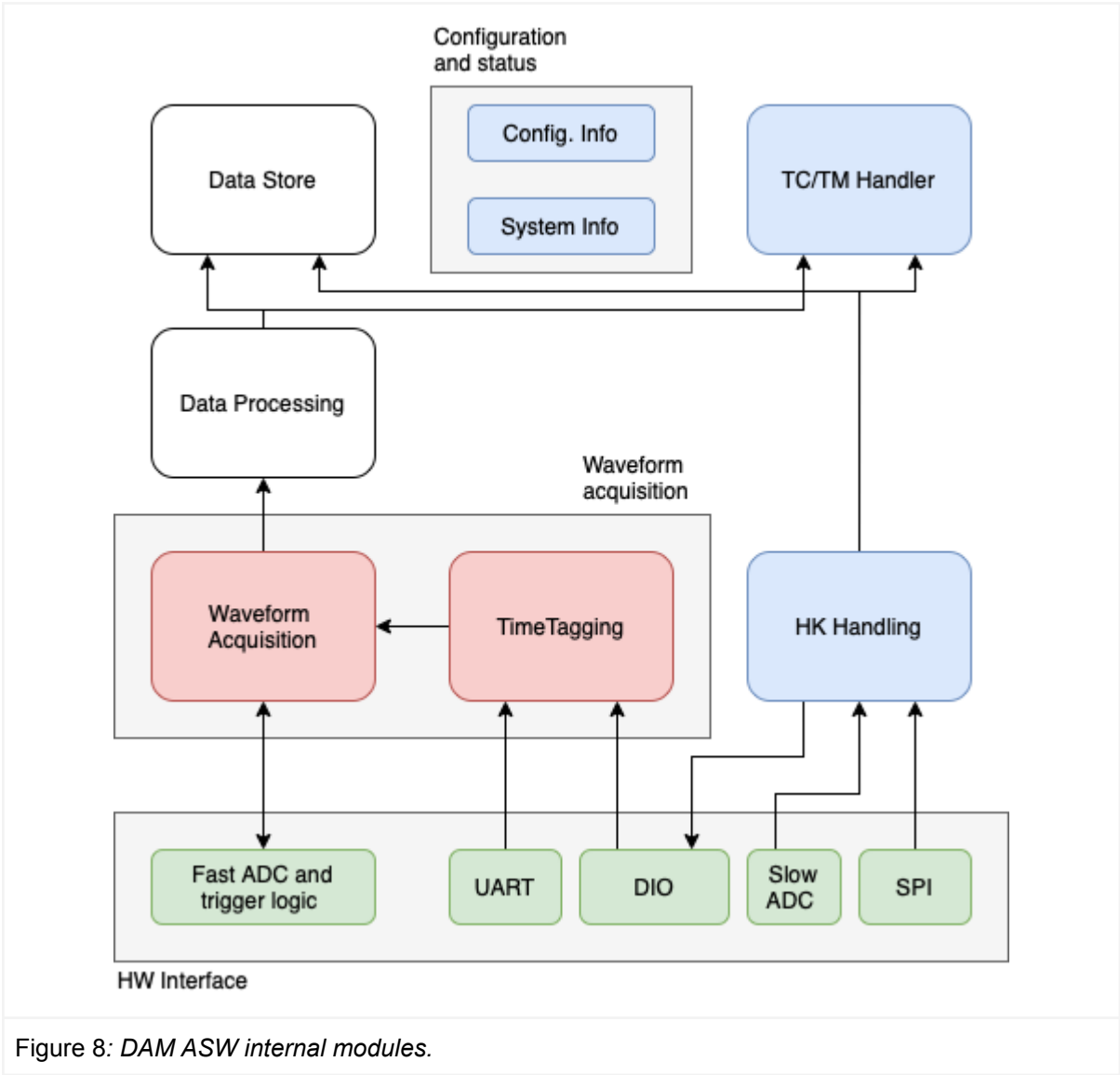
The FPGA FW can be developed in VHDL using the VIVADO or VITIS development tool and the IP cores provided by Xilinx.


The FPGA is connected with the MC by an internal bus, the communication between the MC and the FPGA is based on a custom driver developed using C language and designed to maximize the data transfer speed.

4.3.2. ASW description

4.3.2.1. Internal structure

The DAM ASW controls the detector, the waveform acquisition process and the interface with the MCC. It is a single multithread process implemented in C/C++ using POSIX API on Linux OS. The ASW implements the data flow shown in Figure 8: waveform acquisition on the FPGA side, data transfer from FPGA to the Micro, time tagging, data processing, storage and transfer to the MCC.



		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	17/27

The main modules of the ASW are shown in Figure 8. The HW interface subsystem is in charge of controlling the interface between the Micro and the FPGA, in particular for the configuration and control of the following resources:

- The trigger logic and the fast ADC used to acquire the waveforms.
The UART used to read NMEA sentences from the GPS.
- The DIO lines. One digital input is used to catch the PPS signal from the GPS and one digital output is used to control a relay to power on/off the detector voltage booster.
- The slow ADC used to monitor the output of the voltage booster.
- The SPI used to read the detector temperature.

Note that the interfaces with the fast ADC, the triggering logic, slow ADC and DIO are handled using a custom driver developed to maximize the data transfer speed. For UART and SPI the standard Linux drivers and APIs are used.

The Waveform acquisition subsystem is made up of two critical modules (red rectangles) with the corresponding threads executed with high priority. The Time Tagging module is in charge of receiving both the PPS signal and the NMEA sentences from the GPS and to correlate them with the internal clock of the board. The Waveform Acquisition module configures the trigger logic and arm the trigger waiting for an event (waveform), when an event is detected the acquired waveform is moved from the FPGA to the Micro and properly time tagged. Waveforms are then sent to the Data Processing and Data Storage modules.

Data Processing and Data Storage modules (white blocks) are currently placeholders that will be used for future implementation. In particular, the Data Processing module passes directly the waveforms to the TM/TC Handler.


The HK Handling module is a low priority thread in charge of collecting the HK data (voltage booster reference output and detector temperature) and the status of the DAM. The HK data are acquired with a fixed period and sent to the TM/TC Handler.

The TC/TM Handler receives HK data and Waveforms from the other modules, creates one or more TM packets and sends them to the MCC. TC received from the MCC are decoded by this module and executed according to the logical status of the DAM. For each TC two TMs are generated to indicate that the TC has been received and the results of the TC execution. All the incoming and outgoing packets are protected by 32 bit CRC. The TC/TM Handler implements a TCP-IP server.

The configuration and Status subsystem keeps track of the DAM configuration parameters and all the runtime parameters (such as the logical status). The Configuration Info module loads (and saves) all the configuration parameters from an external xml file. The System Info module implements the DAM state machine keeping all the runtime information such as the status of the power on/off relays, the last values of the HK info.

4.3.2.2. Boot sequence

At the system power on or each time the DAM ASW is executed the following sequence of events is performed:

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	18/27

1. The state of the DAM is set to Startup.
2. Init the Configuration Info module: all the configuration parameters are initialized with hard coded safe (default) values, then the configuration file is loaded and the parameters updated accordingly. If no configuration file is found then, a new configuration file is created containing the default parameters.
3. Init the System Info module: the runtime parameters of the DAM are initialized with hardcoded default values and the initial state of the DAM is set to SERVICE.
4. Init the Data Store module: currently it is an empty module aimed at future implementation.
5. Startup of the TCP-IP server used to handle communication with the MCC.
6. Init of the TC/TM Handler module.
7. Init of the Time Tagging module: the module starts waiting for PPS and NMEA sentences.
8. Startup of the HK Handling module: the periodic acquisition of HK data is started and HK packets are produced.

4.3.2.3. ASW Configuration

The ASW reads its configuration parameters at the startup loading a xml file. Currently, the only means of changing the DAM configuration is manually editing the configuration file before executing the DAM ASW.

Each DAM is uniquely identified by an APID number. This number is reported in each TM packet generated by the DAM.

The acquisition session and the configuration of the parameters are tracked by Session ID and Configuration ID fields.

An additional Run ID is provided by the DAM and reported in the TM packets, this is a counter incremented by the ASW at each boot and each time an acquisition is started. This value is stored in a separate file and can not be edited by the user.

The "Configuration" section of the file contains at least the monitor period in seconds used to collect HK data and sends HK TM to the MCC.

The "Oscilloscope" section of the file contains the settings used to configure the trigger logic on the FPGA side.

The following box contains an example of a DAM configuration file.

```
<?xml version="1.0"?>
<DAM version="02.01.0080" apid="01" damSessionID="00" configID="00" />
<Configuration monitorPeriodSecs="0005" />
<Oscilloscope eqLevel="01" decimation="0001" trigSource="02" trigThresh="002048" trigHyst="000050" trigDelay="010240"
trigDebounce="000010" />
```

4.3.3. DAM client

To support the development and debugging activities a client software with a graphical front-end has been developed using Python, PyQt5, Matplotlib (Figure 9)

The DAM client can be used to send TC and receive TM from the DAM, providing a quick look of the acquired waveforms as well as relevant info stored in the TM packets.

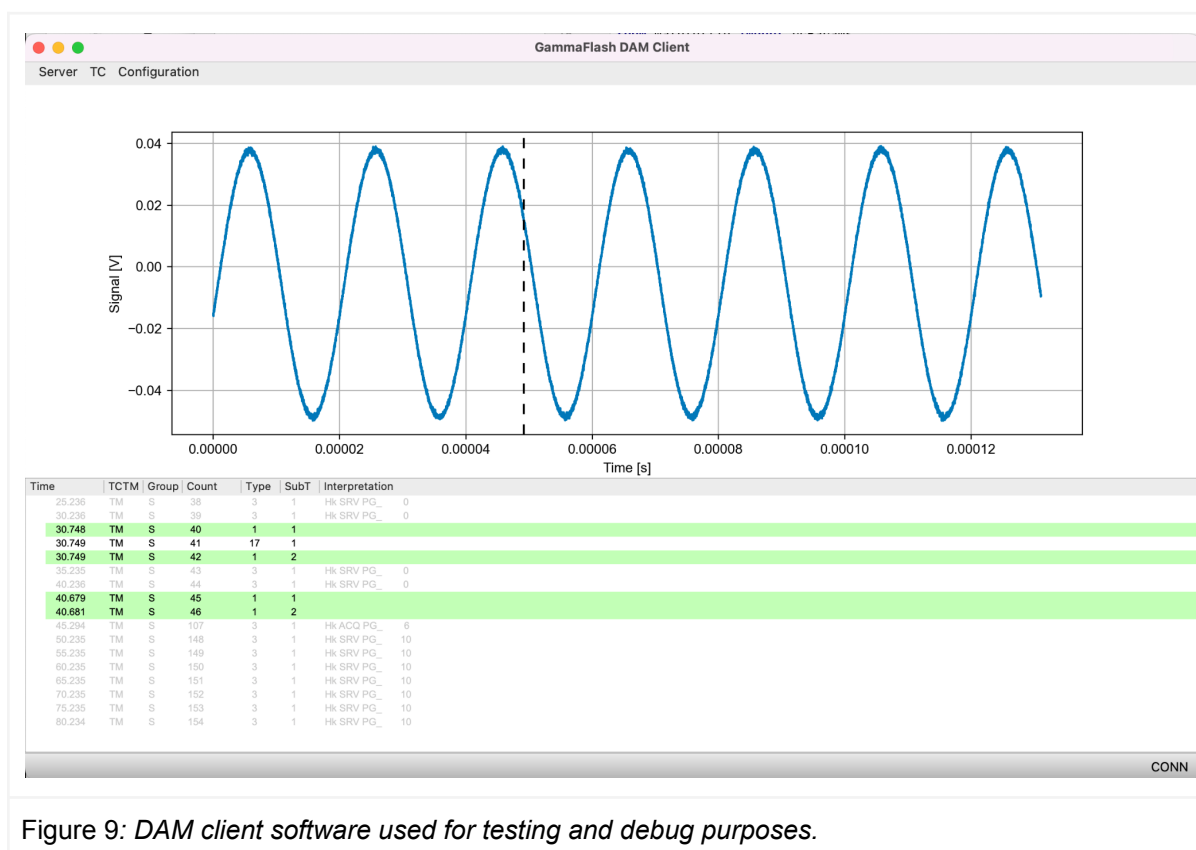



Figure 9: DAM client software used for testing and debug purposes.

4.4. MCC ASW software description

The Gamma Flash ASW data processing pipeline installed in the MCC is a Python library which efficiently processes the data collected by Redpitaya boards. The software has been designed to be scalable and configurable for supporting multiple data types. An xml configuration file is required to describe the dataflow and several parameters required at runtime.

GAMMA-FLASH							
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	20/27

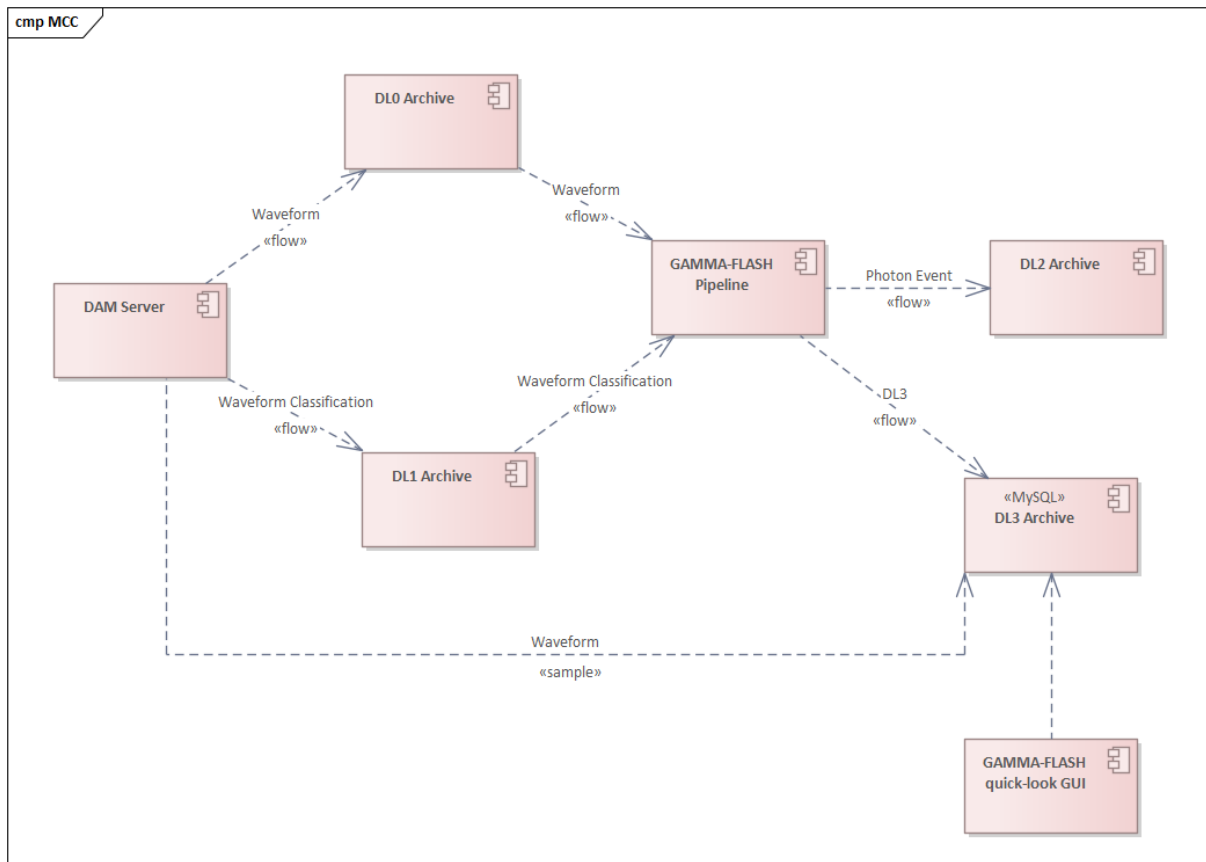


Figure 10: MCC software detailed component decomposition.

Figure 10 represents the MCC software detailed decomposition, with the following main components:


- 1) DAM Server
- 2) DL0 Archive to store DL0 data
- 3) DL1 Archive to store DL1 data
- 4) DL2 Archive to store DL2 data
- 5) DL3 Archive to store in a MySQL database DL3 data
- 6) GAMMA-FLASH pipeline
- 7) GAMMA-FLASH quick-look GUI

4.4.1. DAM Server

The DL0 files are produced by HDF5create Python class. All waveforms received are stored in a shared queue that is consumed by a parallel process when max_number_waveform value is reached.

4.4.2. GAMMA-FLASH pipeline

4.4.2.1. Configuration file

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	21/27

```
<?xml version="1.0" encoding="UTF-8"?>
<rta_dq_pipe_config>

  <logging dir="$RTADQPIPE_TEST_OUTPUT/output_logs/test_dqpipebuilder" level="DEBUG" />

  <databases>
    <database type="mysql" hostname="hostname" port="port" username="username" password="password"/>
  </databases>

  <dqpipelines>

    <dqpipeline type="reco_gammaflash" id="reco_gammaflash_max" dqchain_id="nodqchain" nthreads="0">
      <input_data type="gfh5" input_dirs="$RTADQPIPE_TEST_OUTPUT/dl0/00000/" file_pattern="*"
reading_string="/waveforms" join_with="" join_keys="" filter_column="" filter_value=""/>
      <output_data type="gfh5" output_loc="$RTADQPIPE_TEST_OUTPUT/reco_output" overwrite="false"
suffix="dl2" />
    </dqpipeline>

    <dqpipeline type="dq_analysis" id="" dqchain_id="gamma-flash-dq_pipeline" nthreads="0">
      <input_data type="h5" input_dirs="$RTADQPIPE_TEST_OUTPUT/reco_output" file_pattern="*"
reading_string="/dl2/eventlist" join_with="" join_keys="" filter_column="" filter_value=""/>
      <output_data type="pickle" output_loc="$RTADQPIPE_TEST_OUTPUT/dq_output" overwrite="false"
suffix="results" />
    </dqpipeline>


  </dqpipelines>

</rta_dq_pipe_config>
```

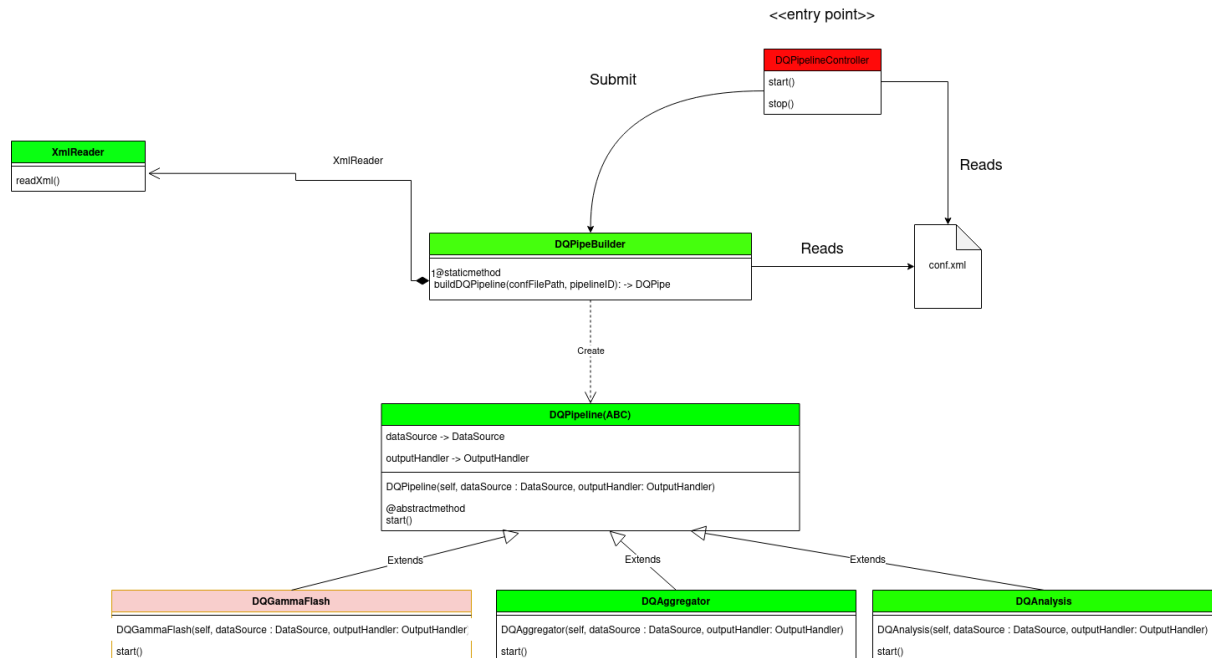
In the figure above is shown a xml file template that describes a typical configuration for one Redpitaya connected to the MCC, it contains:

- The output directory of the logging files
- The parameters of a mysql connection
- A dqpipeline containing:
 - the input data directories with reading and joining string for HDF5 tables
 - the output data directories
 - id field for algorithm selection
 - the number of threads to allow parallel processing
 - the algorithm selected to process the data

Using the configuration file It is possible to create a data processing workflow, for example to connect a dqpipeline X with dqpipeline Y it is necessary that the input directory of dqpipeline Y and the output directory of dqpipeline X must be the same.

GAMMA-FLASH						
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page: 22/27

4.4.2.2. Class Diagram: pipelines

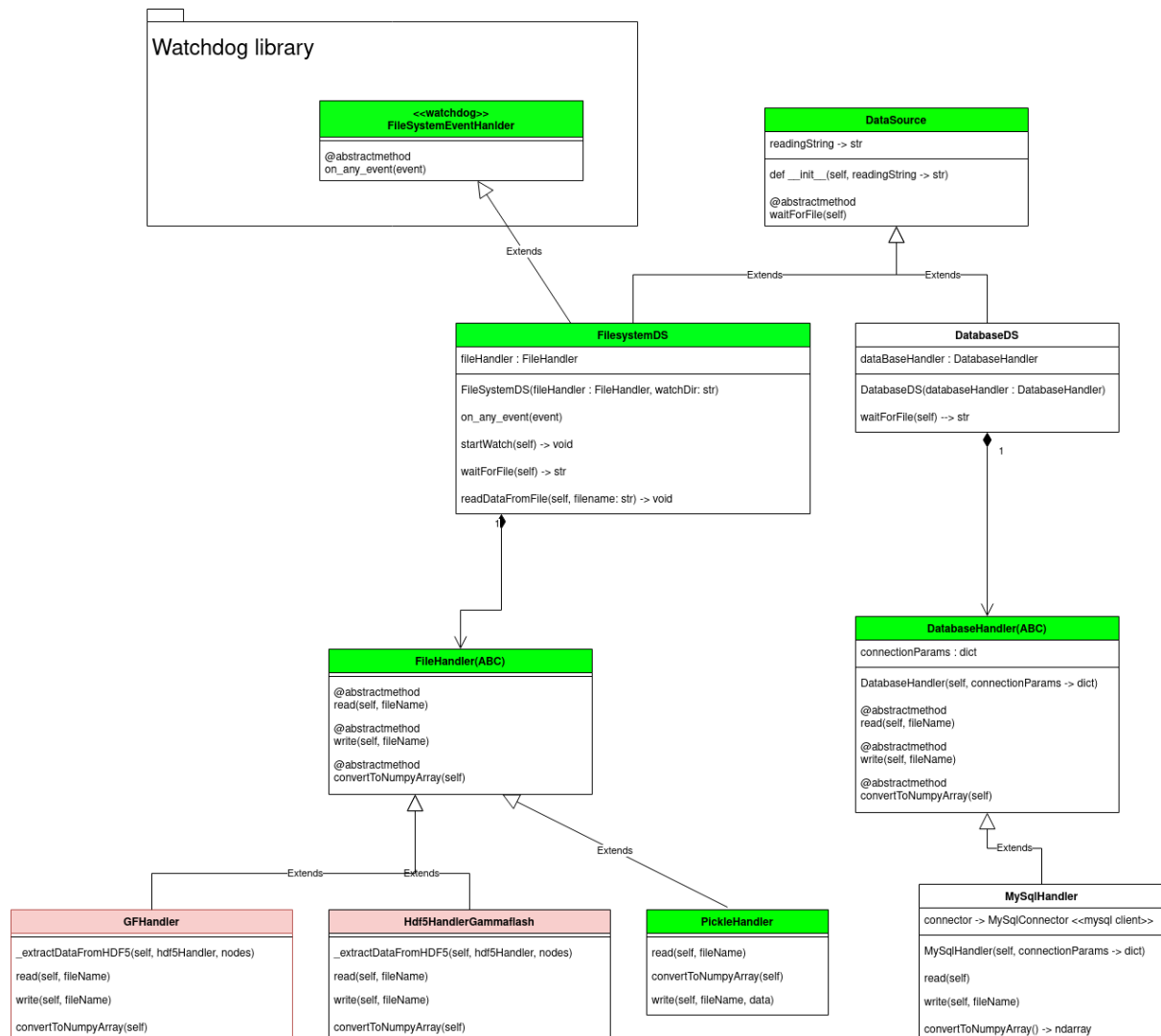


The entry point of Gamma Flash data processing pipeline is **DQPipelineController**, it instances a **DQPipeBuilder** object using the method `start`. **DQPipeBuilder** reads a xml configuration file and builds one or more pipelines defined on it.

It has been developed an abstract class that implements three pipelines with different strategies:

- **DQGammaflash** class opens DL0 file, processes the waveforms and writes a DL2 file.
- **DQAnalysis** and **DQAggregator** use DL2 for generating the cumulative energy spectrum.

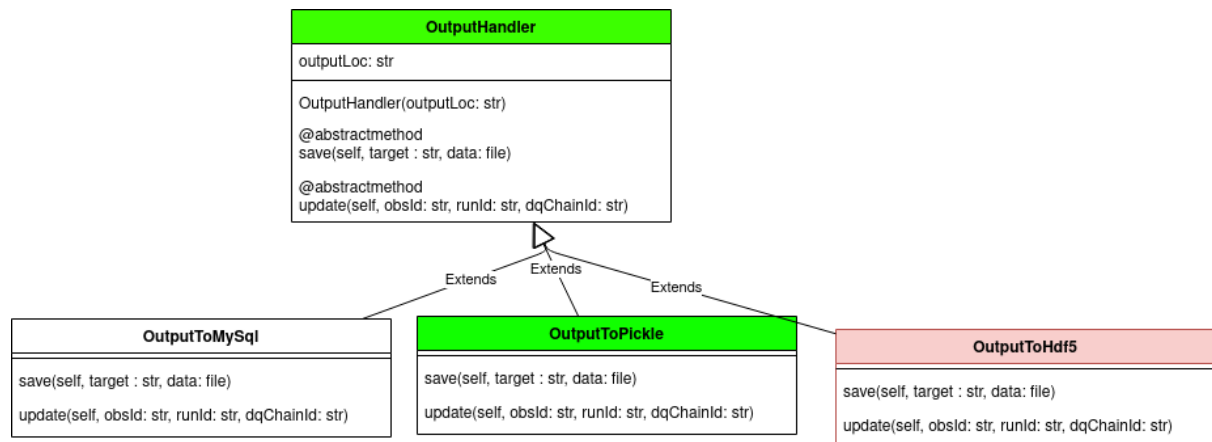
4.4.2.3. Class Diagram: datasource



The diagram shows the class structure of Datasource for handling different data types. FilesystemDS implements a watchdog library that notifies when a new file is created on a specific directory.

- Hdf5HandlerGammaflash is responsible for opening DL0 and DL1 files produced by the dam. DQGAMMAFLASH uses this class for getting the waveforms values in an array.
- GFHandler opens DL2 files. It is used by DQANALYSIS in order to get eventlist values.
- PickleHandler is used for DL3 files. It wraps the pickle python library to have a similar interface to DL0, DL1 and DL2 data layers.

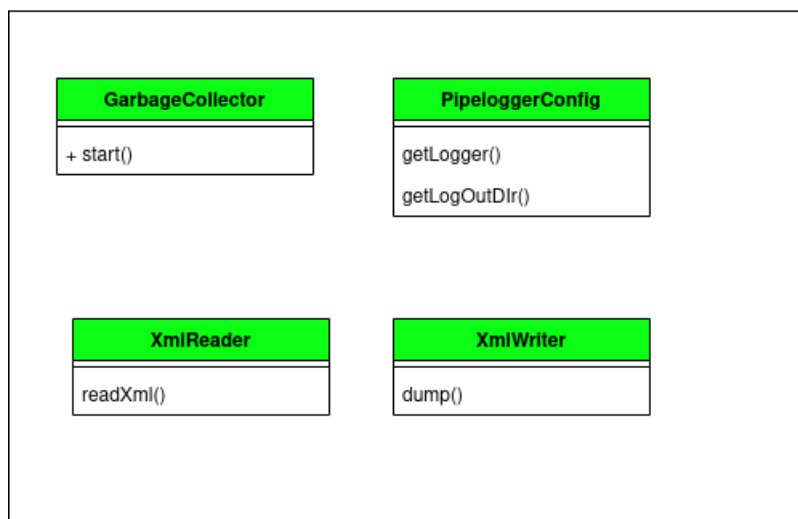
4.4.2.4. Class diagram: Output Handlers



Output handlers write the data in a specific format decided by the user on the configuration file. Pickle and HDF5 format are available for writing into the filesystem. OutputToMySQL is responsible for writing into a MySQL database, the connection parameters are located on the configuration file.


4.4.2.5. Class diagram: Utils

Utils



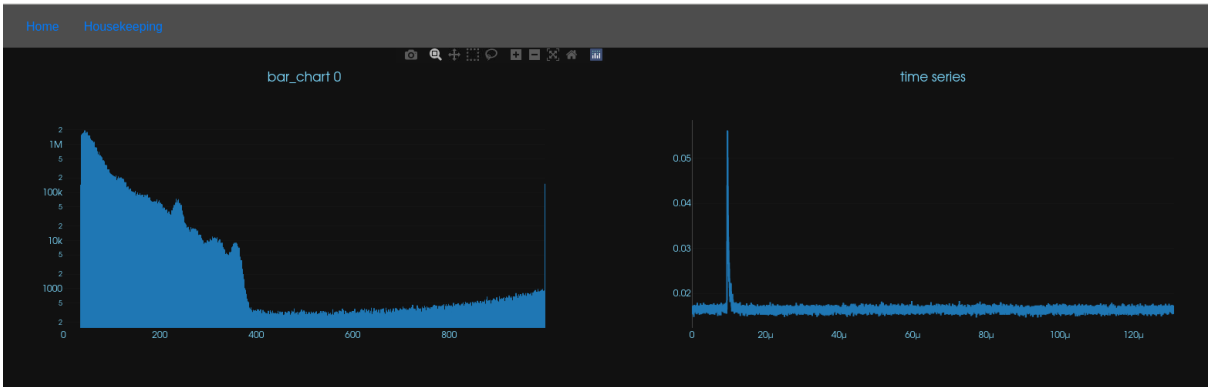
These classes are components used for utility purpose:

- PipelggerConfig is a logger implemented as Singleton
- Garbage collector deletes data files that are already processed and not required
- XmlReader for configuration file parsing
- XmlWriter for configuration file creation

		GAMMA-FLASH					
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page:	25/27

4.5. GAMMA-FLASH quick-look GUI

A quick look dashboard is used for checking the system health and monitoring the real time processing. It is connected to a MySQL server and it shows the cumulative energy spectrum of gamma radiations collected from PMTs, DL0 waveforms randomly chosen in a selected time window and the temperature curve produced by the weather station.



The figure above shows the dashboard prototype developed using Plotly Python library. For each RP are available two plots showing DL3 and DL0 to the left and to the right respectively. The refresh interval value can be easily set from the configuration file. Plotly uses a nginx web server to allow the access via browser, and can be remotely deployed on the cloud or in an external local machine.

5. Data model

The data model describes data which flow through the system.
Detailed description of the data model is reported in [AD2].

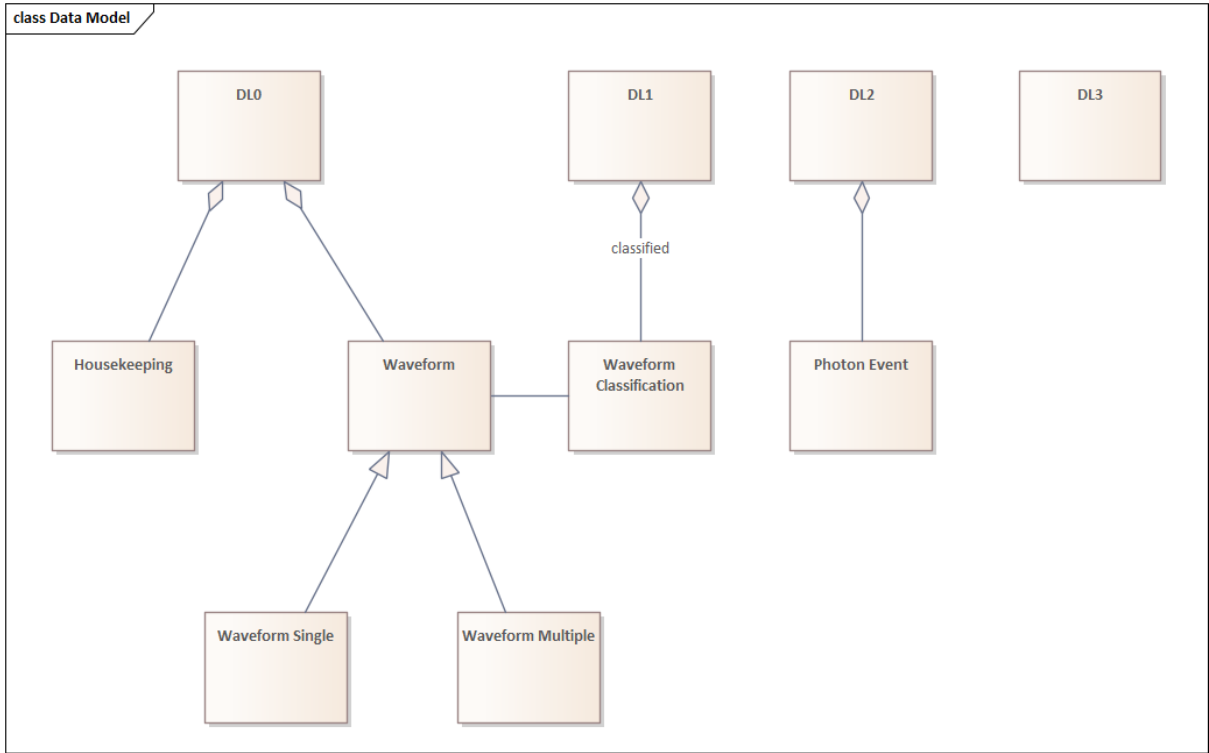



Figure 11: data model

GAMMA-FLASH						
	GAMMA-FLASH-001-SDD	Issue:	1.1	Date:	May 11, 2022	Page: 27/27

6. Deployment view

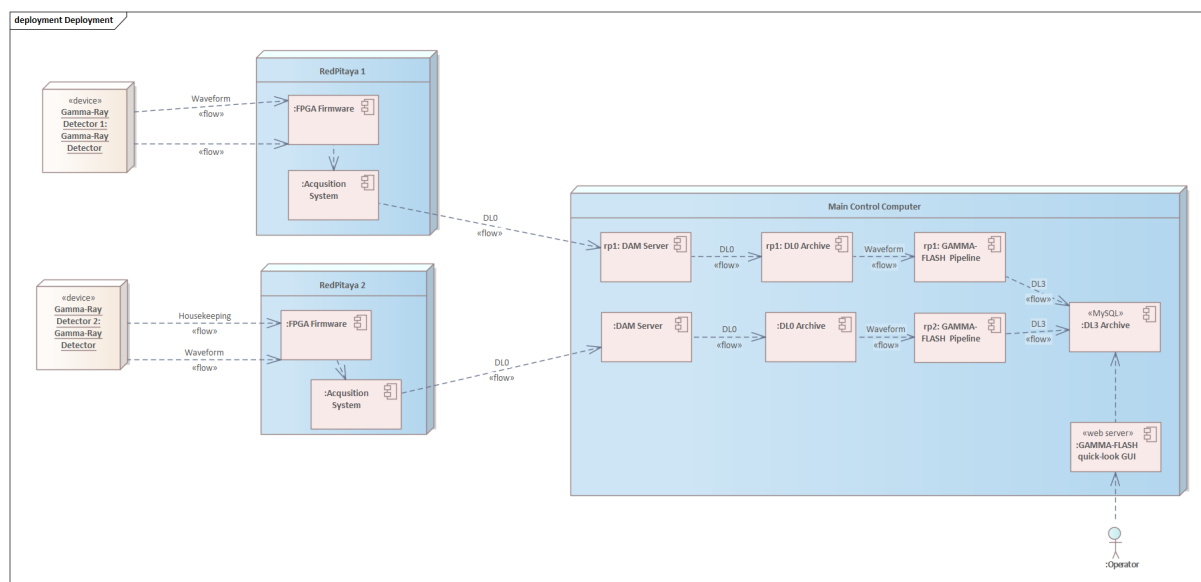


Figure 12: deployment diagram

The deployment diagram above shows the Gamma Flash logical-physical ecosystem. In the figure is shown a typical scenario:

- One or more Gamma Ray detectors connected to RedPitaya boards.
- The data flows from RedPitaya to the MCC that computes the data layers: DL0 DL1 and DL2 are saved on the filesystem, DL3 stored on a MySQL database.
- The operator views DL3 through the Gamma Flash GUI.