Simulate the DAQ for the VMM-FE-ASICs

A. Stylianidis [Undergraduate Student of Computer Science UoPeloponnese]
N. Benekos [NTUA]

European Union
European Social Fund
OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION, LIFELONG LEARNING AND RELIGIOUS AFFAIRS
MANAGING AUTHORITY
NSRF 2007-2013
EUROPEAN SOCIAL FUND
Co- financed by Greece and the European Union

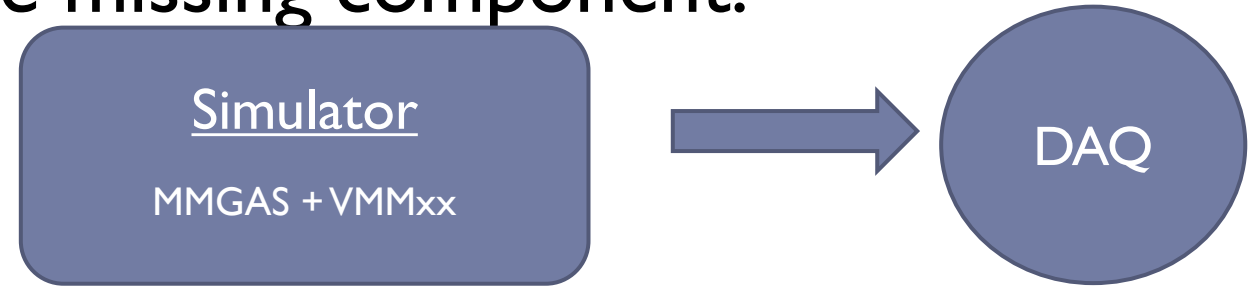
Outline of Presentation

- Introduction
  - Motivation
  - Semantics
- Program Description and Features
- Future Steps
- Summary

Introduction - Motivation

- In the framework of NSW Project update for ATLAS detector there is a need to test the DAQ system for the VMMxx electronic chip.

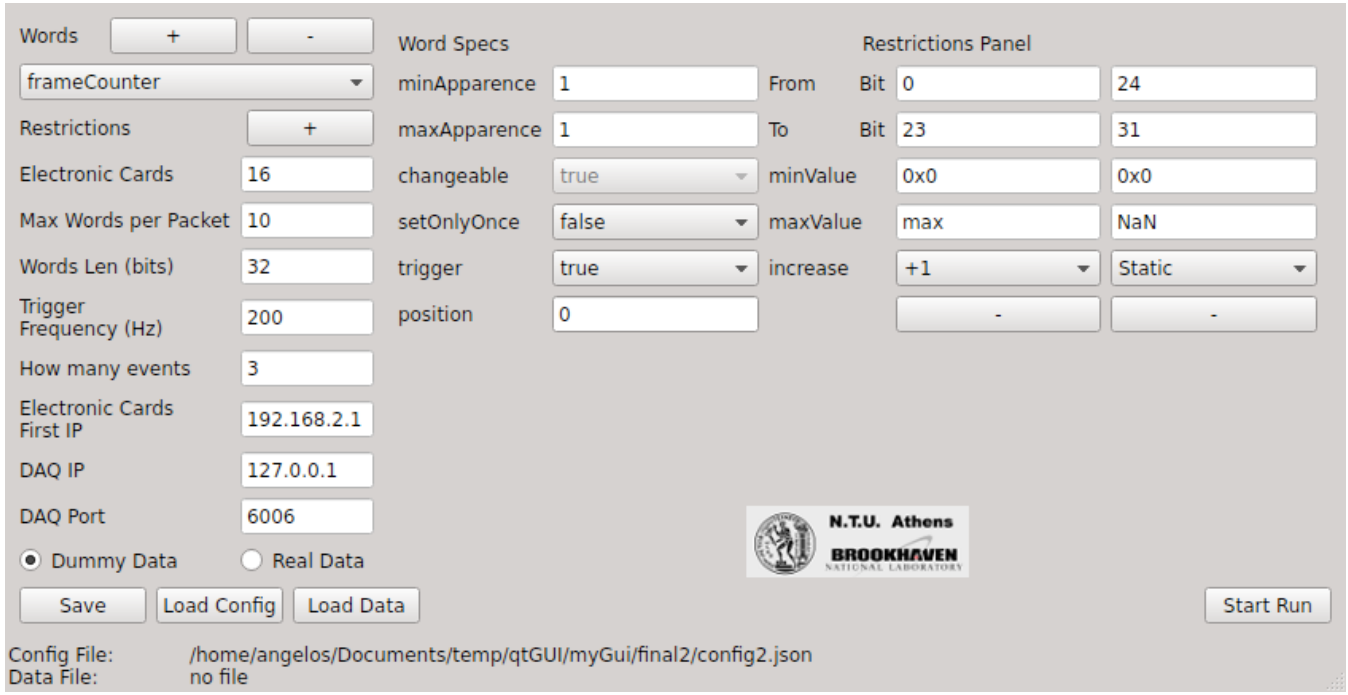
- There is ultimate need to simulate the response of VMMxx cards, since it is the missing component.



Introduction - Motivation

- The DAQ software system of the previous version of VMMxx is available.
- A computer simulator written in C++ using the QT framework was implemented, with the following features :
  - 1. Dynamic
    - Very wide flexibility in the simulated UDP packages created
  - 2. Ability to load real datasets and simulate the behavior for the real case.
  - 3. User friendly GUI for easy use by the physics users not only at CERN
- In the following slides the code characteristics and the program itself are presented in detail.

Preview of the simulator



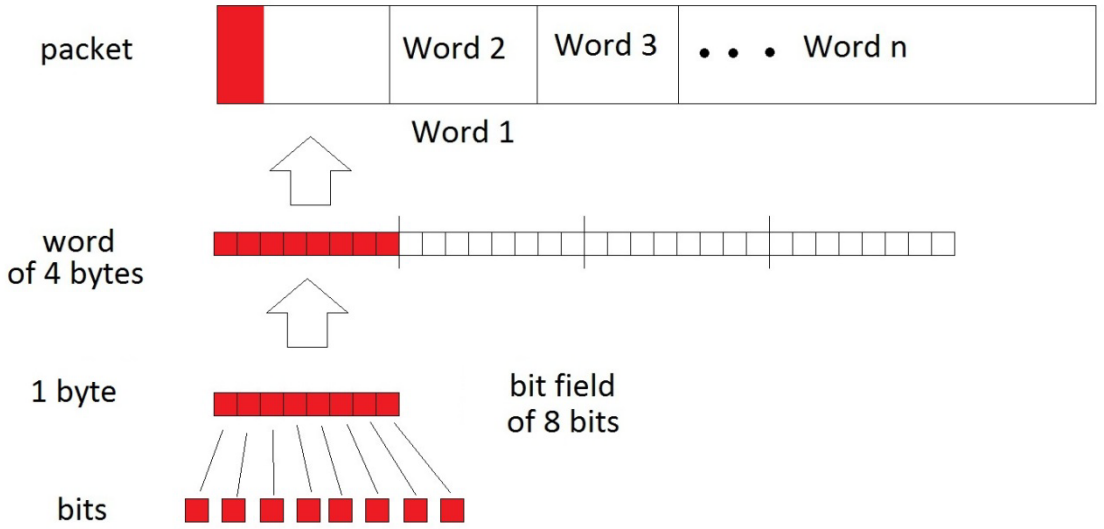
Program screen shot before any input

The window with a loaded file, with initialization information

Semantics

- Special terminology that will be used in the rest of the presentation :
  - "BIT FIELD" → A series of bits that can be treated together.
  - "WORD" → It's a bit sequence that is recognized by the current architecture as an instruction.
- "PACKET" → It's equivalent of a post-office envelope. Consists of the headers and the main information(data).

Virtual Representation



- Normally the UDP header is taken from the system, but in our case we build a custom UDP header using raw data. This is implemented to allow for the user to define custom IP address for them VMMxx for the proposes of his simulation.

Code Structure

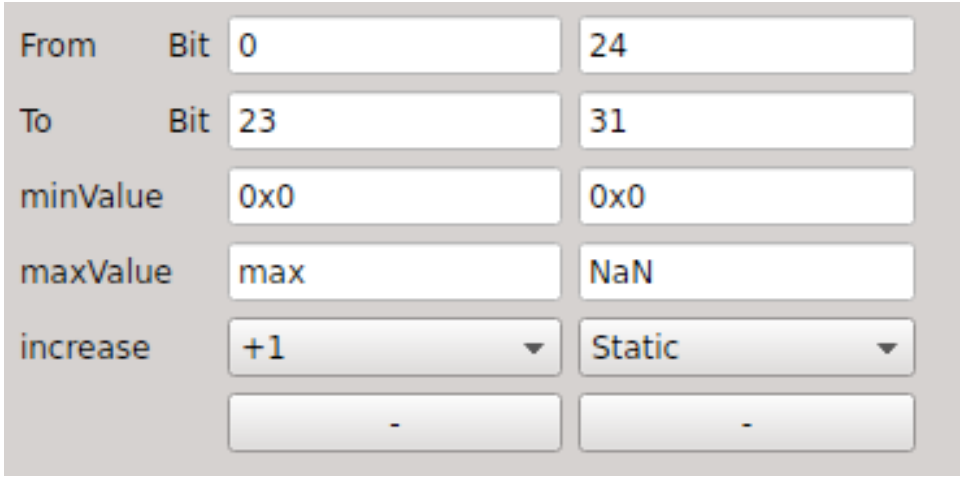
- Main purpose of the program is **to create** and **transmit** custom packages, with characteristics ("restrictions") defined by the user.

- Each "part"(=bit field) of the "word" is treated according to the restriction referring to that bit field.

- Each restriction set by the user affects the corresponding bit field

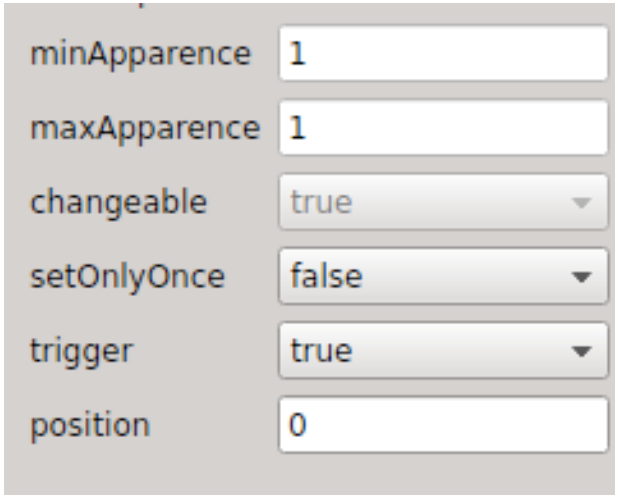
Restrictions Panel

- Each restriction defines:
  - The bit field which is referring at:
    - From Bit n_1 → To Bit n_2
  - The minimum and maximum allowed values of this bit-field
  - The way in which the bit-field handles its value.
    - Either remain static
    - Change Randomly
    - Changing by step +1



Words' Panel

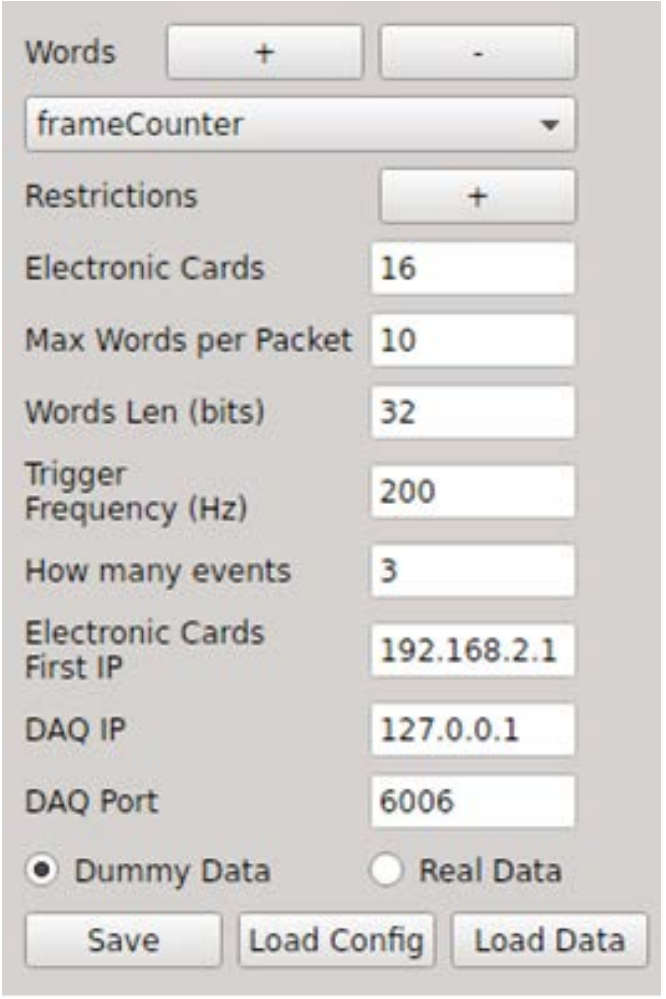
- Each "word" obeys also to a set of (user defined) restrictions :
  - Number of occurrences
  - "Changeable" and/or "Send only one"
    - This is to gain CPU power.
    - There are two case scenario where we can know the bit field value before we need it.
      - The case where we know from the beginning that when we set the bits they will never change.
        → STATIC WORD
      - And the case where a word is never going to change after the first time that the user (electronic cheap VMMxx) sets it.
        → STATIC & CHANGEABLE WORD



- Trigger
  - Trigger refers to a word that changes only when the trigger hits. When a user(VMMxx) needs it, it just takes it with the current value.
  - The only way for the value of that word to be changed is by Object of class Trigger.
- Position
  - The position of the word in packet (1st 2nd …)

Package Panel

- From this panel new words and restrictions can be created, as well as various parameters of the simulation can be regulated.
  - Electronic Cards:
    - number of VMMxx to be used
  - How many events:
    - the user can choose how many events he will run, either with dummy or real data
  - Electronic Cards First IP:
    - The user can change the sender's IP. The IP's are generated one-by-one starting from the first-one.

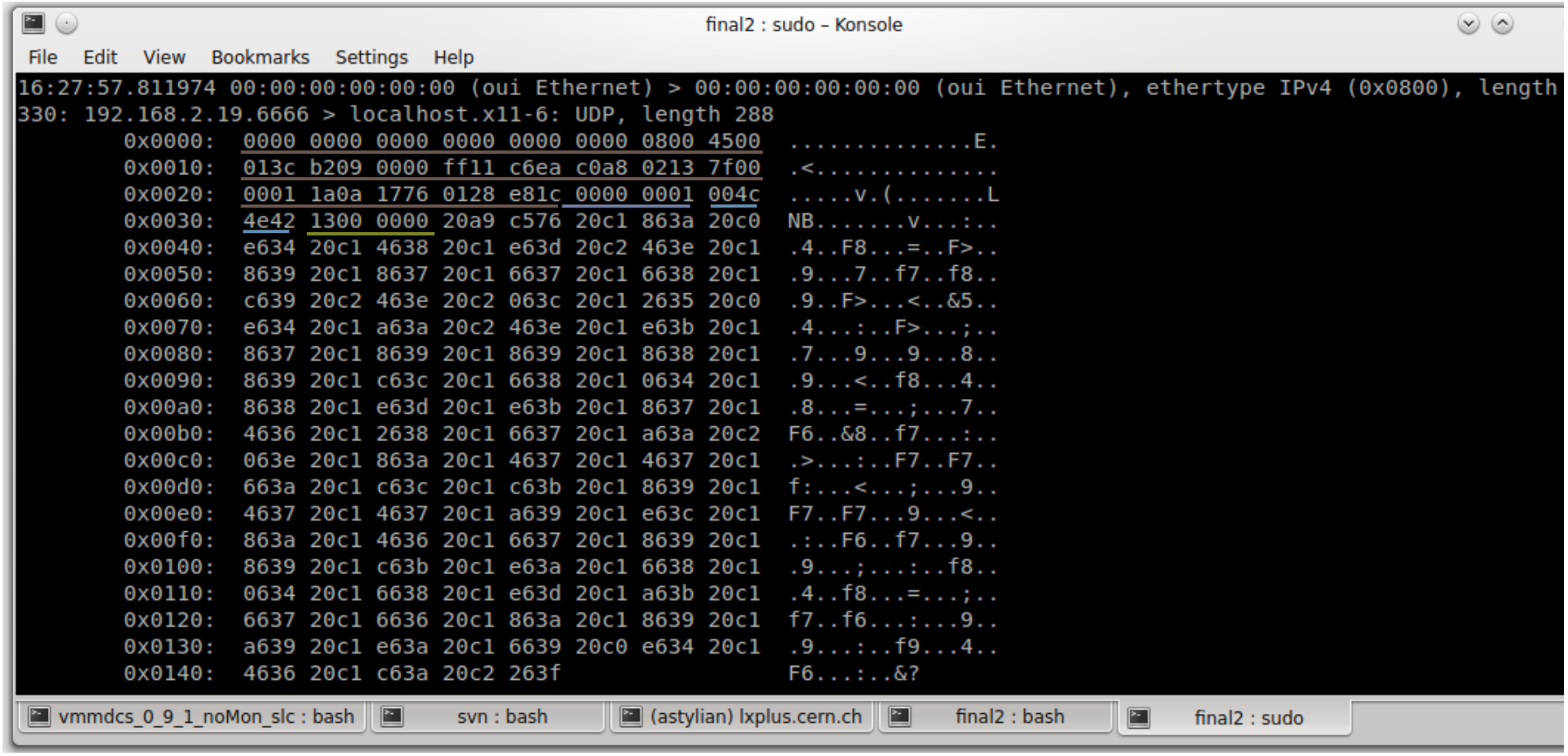


Test Run

- In the following example real data are used.
  - Set-up the description of data model
  - Load the root ntuple with the real data
  - Run the simulation and…
  - Capture the communicated packages with tcpdump sniffer.
    - tcpdump –i lo 'udp port 6006' –c 10 –e –XX

Data Model					
		byte 3	byte 2	byte 1	byte 0
frame counter		0x0	trig cnt1	trig cnt2	trig cnt3
srs data header		0x0	0x4C	0x4E	0x42
srs header info		board id	0x0	0x0	user def
data		bits 31-19 amp	bits 18-6 time		bits 5-0 ad

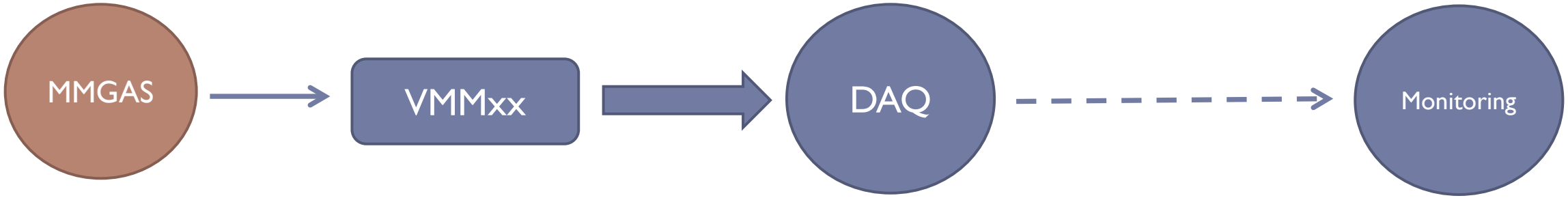
Read data with tcpdump



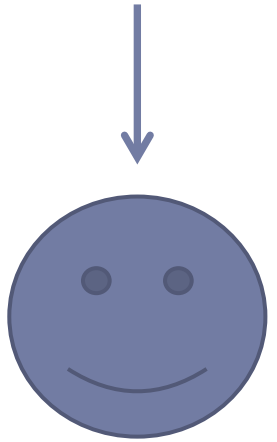
- UDP Header
- frame counter
- srsDataHeader
- srsHeaderInfo
- Data

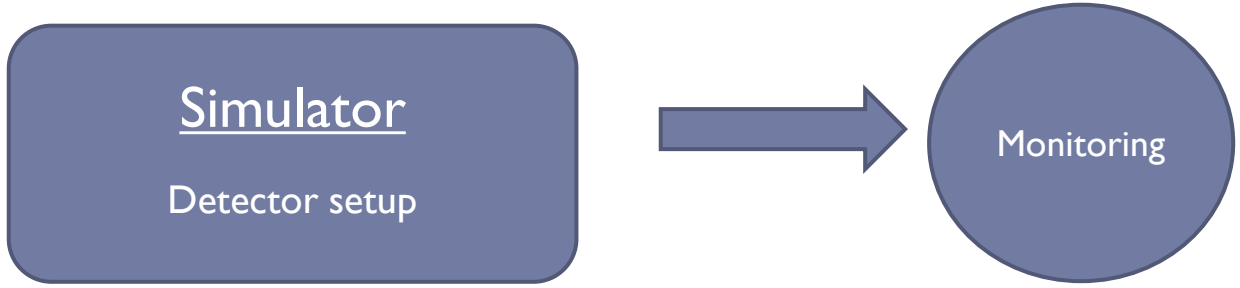
(no line)

Future Steps



- Implementation of a communication system for DAQ with monitoring
- To do that we will implement the communication, test it with an other simulator and then import it in DAQ system.





Thank you

I would like to thank the following people who gave me the opportunity to be here and also help me during my time here.

- Prof. Theodora Papadopoulou
  - For giving me the opportunity to be here and running all the background procedures
- Rector Prof. Konstantinos Masselos
  - For giving me the opportunity to be here
- Dr. Nectarios Benekos
  - For being my supervisor and for teaching me all the interesting things at test-beam and for being so patient.
- Prof. Theo Alexopoulos and all NTUA-HEP members
  - For giving me the opportunity to join the NTUA group and giving me the chance to work with them