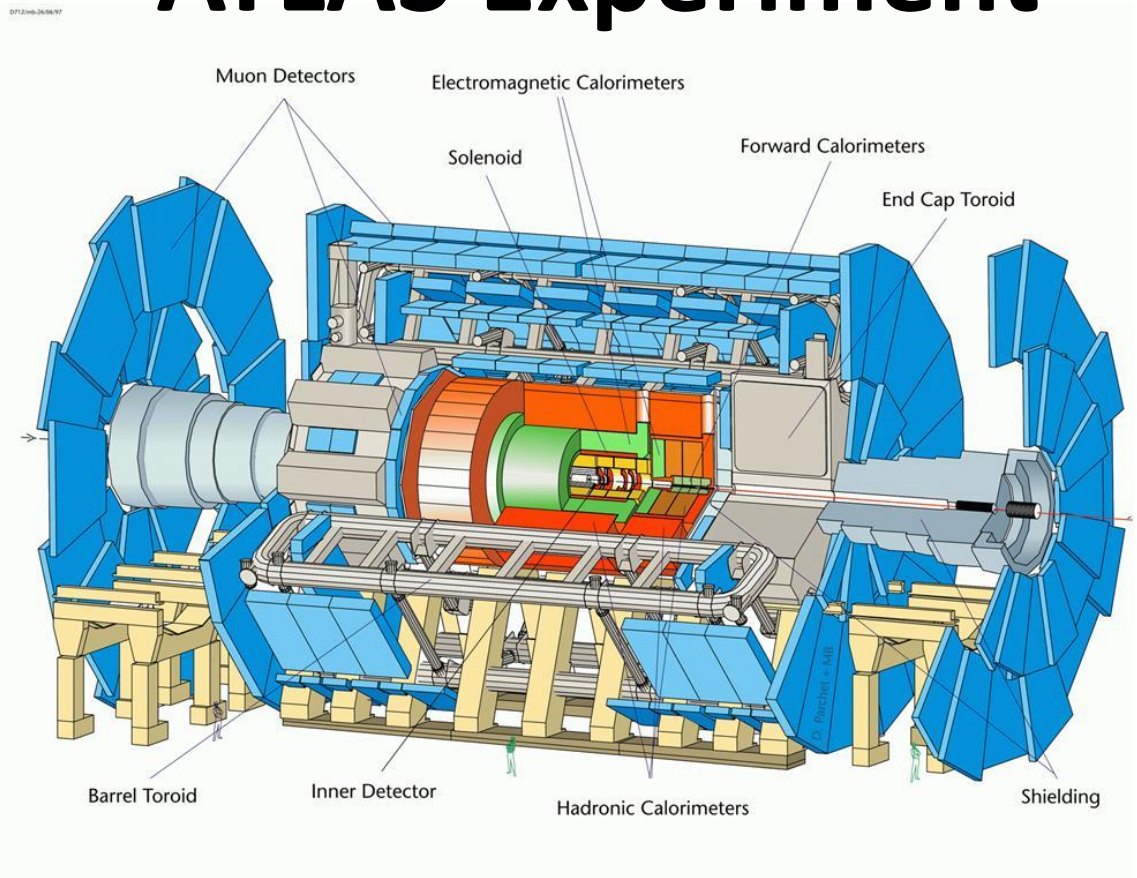
CentOS Linux for the ATLAS MUCTPI Upgrade

- Introduction
- MUCTPI Upgrade
- System-on-Chip
- Use of CentOS
- Summary

R. Spiwoks[a], A. Armbruster[a], P. Czodrowski[a], N. Ellis[a], P. Farthouat[a], S. Haas[a], A. Kulinska[a,b], A. Marzin[a], P. Papageorgiou[a,c], T. Pauly[a], S. Perrella[a], M. Saimpert[a], P. Vichoudis[a], T. Wengler[a]

a) CERN, Switzerland
b) AGH University of Technology and Science, Krakow, Poland
c) National Technical University of Athens, Greece

ATLAS Experiment



General-purpose experiment at the Large Hadron Collider (LHC) at CERN: proton-proton collisions at a centre-of-mass energy of ~14 TeV, with bunch crossing (BC) every 25 ns (40 MHz) with up to 80 collisions (pile-up) expected for the next run (2022)

⇒ More than 10^9 interactions per second potentially interesting to physics: need trigger system in order to select events which are interesting to physics and which can be recorded at a reasonable rate

ATLAS TDAQ System

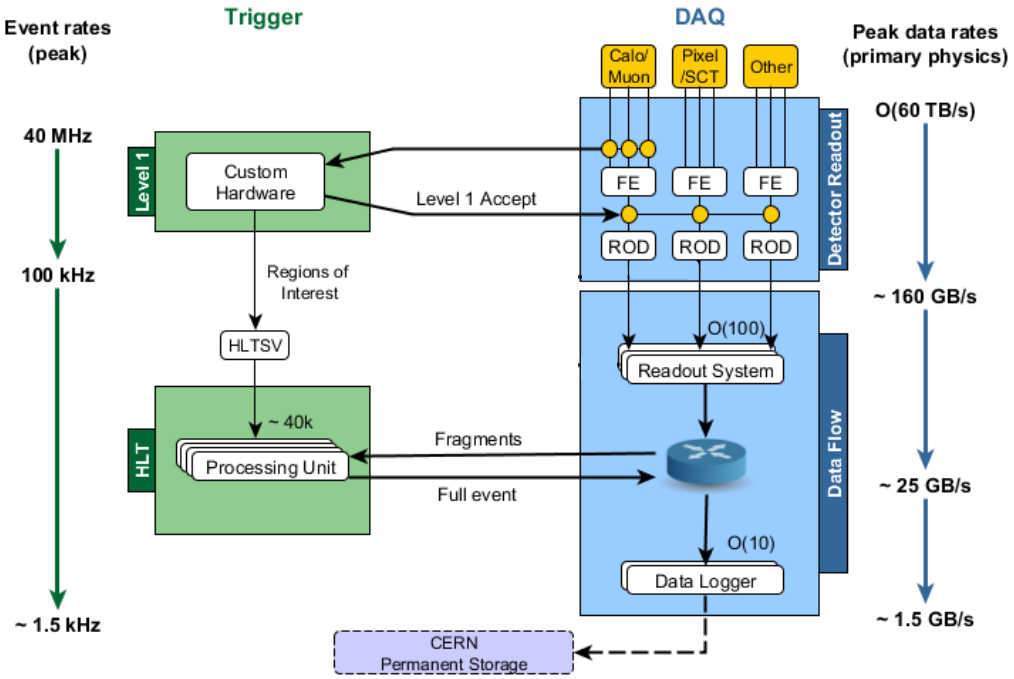


Diagram from https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ

Two-level trigger system:

- **Level-1 Trigger: based on custom electronics and firmware – reduction to 100 kHz event rate**
- **High-Level Trigger: commercial off-the-shelf computers, network and software – reduction to ~1.5 kHz (peak) event rate and ~1.5 GByte/s data rate**

ATLAS Level-1 Trigger System

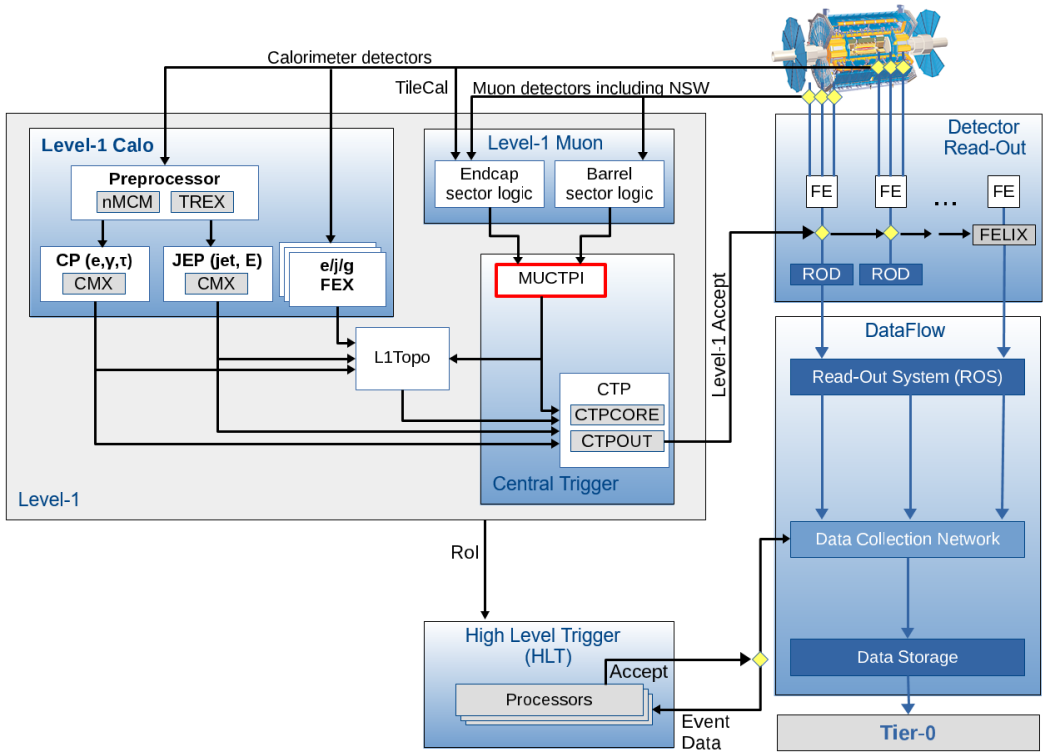
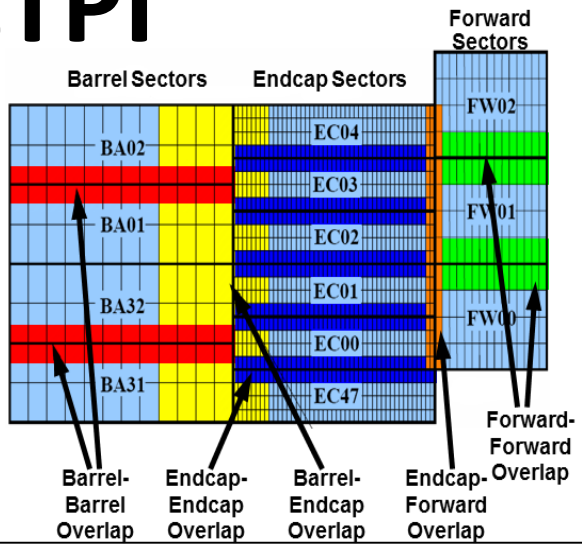


Diagram from https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ApprovedPlotsDAQ

- Based on muon (dedicated detectors) and calorimeter (reduced granularity) information
- MUCTPI = Muon to Central Trigger Processor (CTP) Interface:
  Combines trigger information from barrel (resistive plate chambers) and end-cap and forward (thin-gap chambers) muon detectors
- Topological Trigger Processor combines muon and calorimeter information
- Central Trigger Processor (CTP) takes final Level-1 Accept decision

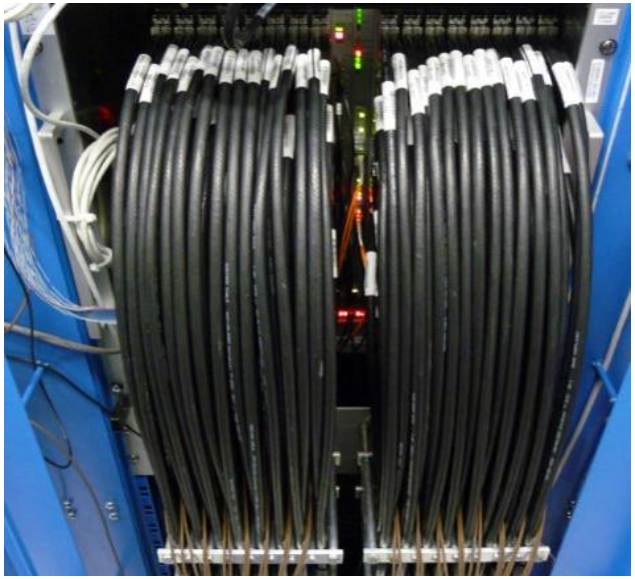
ATLAS MUCTPI

- Combines information from 208 muon sector logics
- Avoids potential double counting of muon candidates (overlap handling)
- Sends highest-p_T candidates to the topological trigger processor
- Sends multiplicities to CTP



Geometrical coverage of typical overlap of barrel, endcap and forward sectors

Up to Run 2 (2018): VME based system, with electrical cables from muon sector logics to MUCTPI, and 18 different modules



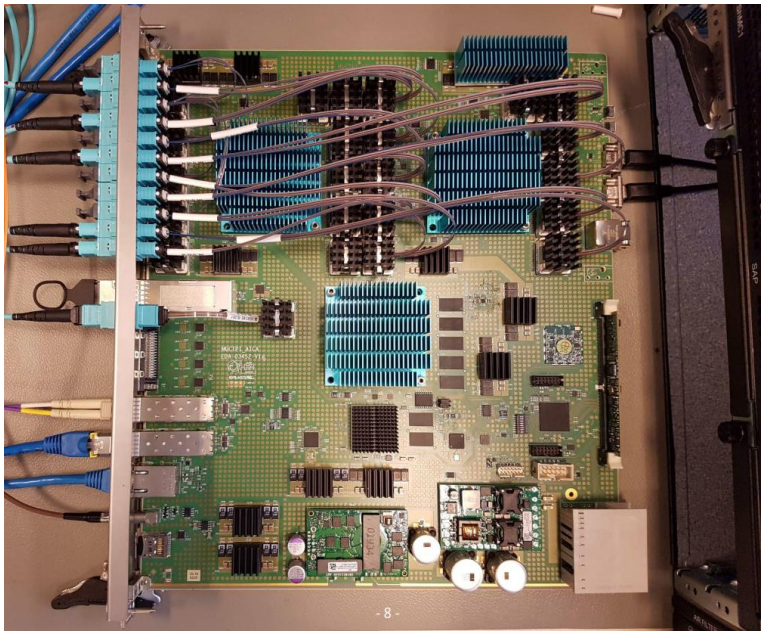
Upgrade of the ATLAS MUCTPI

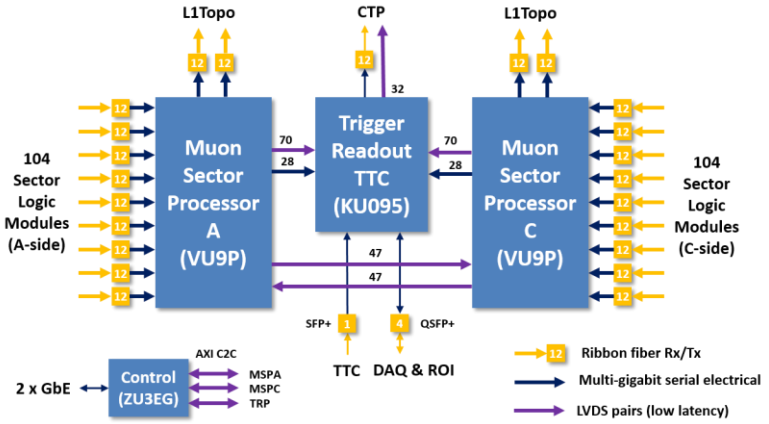
For Run 3 (2022) we need to enhance the trigger selectivity at higher luminosity:

- **More muon candidates per sector and more information per candidate ⇒ need higher bandwidth**
- **Provide improved overlap handling ⇒ need more processing**
- **Provide possibility for muon-only topological trigger processing in MUCTPI**
- **More muon candidates with full granularity to topological trigger processor**
- **Make it future-proof for Run 4**

ATCA based system, with optical links between muon sectors and to MUCTPI, which is a single module

Use state-of-the-art FPGAs (Xilinx Virtex/Kintex Ultrascale+) for the data transfer and processing



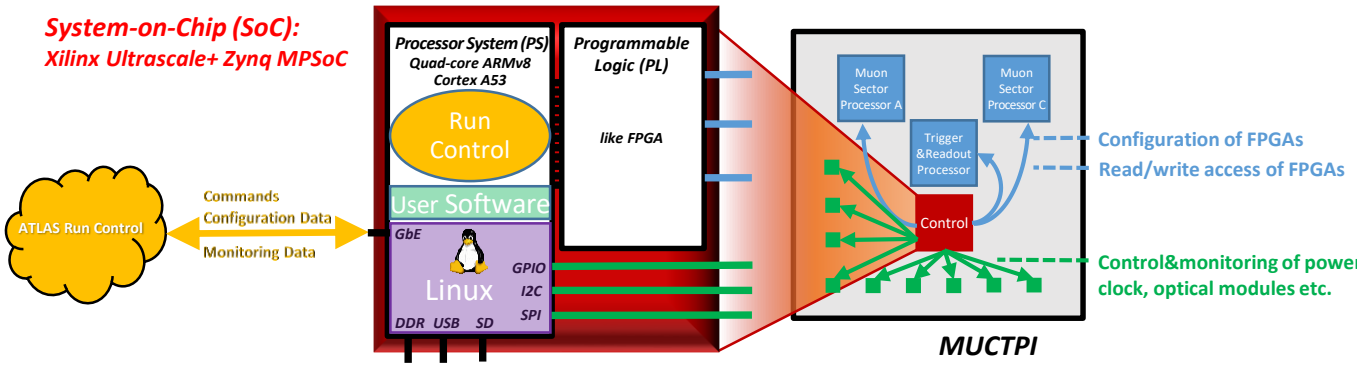


System-on-Chip

How to control the MUCTPI?

Previously, we had a Single-Board Computer (SBC) in the VME crate

SBC was running Scientific Linux CERN (SLC), the user application software was fully integrated into the ATLAS TDAQ system



⇒ **System-on-Chip = CPU + FPGA** *(note, that this is a slightly more restrictive definition than usual)*

- **Processor system (PS) = like CPU:**
  - **Based on multi-core ARM processors**
  - **Has memory and peripherals: GbE for communication with ATLAS run control; I2C, SPI, GPIO, etc. for control of hardware**
  - **Runs software: "bare-metal" application or operating system, e.g. Linux**
- **Programmable logic (PL) = like FPGA:**
  - **Has logic cells, memory blocks, I/O links, and MGTs**
  - **Implements interfaces to the other processing FPGAs, can implement real-time logic or additional peripherals, e.g. 10GbE**

Xilinx Zynq/ZynqMP

MUCTPI uses Xilinx FPGAs, so it was natural to use a Xilinx SoC:

Xilinx Zynq Ultrascale+ MPSoC ZU3eg, quad ARM Cortex A53 (armv8/aarch64, 64-bit), 1.2 GHz, 4 GByte DDR4

What operating system to use?

Note: Xilinx/Vivado and Xilinx/SDK are used to prepare other boot files:

- **Bitstream file: for programmable logic**
- **First-Stage Boot Loader (FSBL): software to initialize hardware and to load bitstream file and bootloader**
- **Device tree files: used by the Linux kernel**

PetaLinux (Xilinx) or Yocto/OpenEmbedded with Xilinx meta layers:

- **Linux kernel and (Xilinx) drivers**
- **Linux root file system**
- **U-Boot for loading kernel, device tree, and root file system**

This is not a CERN certified OS and will not be allowed to run on the ATLAS Technical Control Network (ATCN) – needs to be isolated behind a gateway PC

Requires effort for updating continuously … ATLAS system administrators or CERN-IT will not provide support for it

CentOS (1)

Operating system is key to the operation of the SoC and its integration into the ATCN and the ATLAS TDAQ software!

⇒ Use CERN CentOS (http://www.centos.org and https://linux.web.cern.ch/centos):

- **Widely accepted**
- **Available for aarch64**
- **Continuously updated**

Cross install on a host PC:

- **Example: *dnf … --forcearch=aarch64 … groupinstall 'Minimal Install'***
- **Uses qemu and Linux' binfmt_misc capability**
- **Full recipe on the twiki page https://twiki.cern.ch/twiki/bin/view/SystemOnChip/CentOSForZynqMP**

This is a result from the SoC Interest Group, system-on-chip@cern.ch:

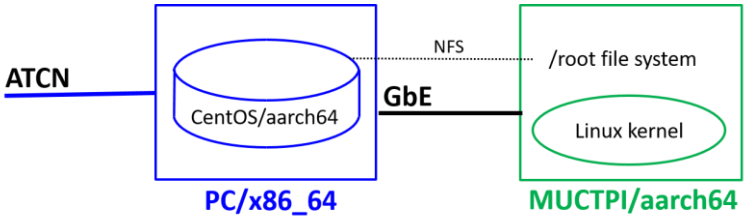
- **Workshop in 2019: https://indico.cern.ch/event/799275/**
- **Interest group meetings: https://indico.cern.ch/category/11883/**
- **Twiki page: https://twiki.cern.ch/twiki/bin/view/SystemOnChip/WebHome**
- **Organising committee following up on common issues:**
  e.g discussions with CERN-IT to provide CentOS/aarch64 in the same way as CentOS/x86_64 …

Note: this is not agreed upon for Run 3, and will only come for Run 4,
in the meantime ATLAS MUCTPI will continue using CentOS, and run behind a gateway PC

CentOS (2)

1) Use kernel (Xilinx-specific drivers) and U-Boot from Yocto (alternatively PetaLinux)

2) Mount cross-installed CentOS rootfs via NFS (could also be on SD card)



⇒ Full operating system, provides:

- ssh for login
- bash, python, tcl, expect, etc. for scripting
- NFS mount of other file systems, NTP for date and time
- Linux tools for network: ip, ifconfig, DHCP client, …
- iptables for network security
- Many other packages can easily be installed

⇒ looks like any other CERN Linux system:

```
[spiwoks@pcphl1ct07] ~> ssh l1ct-muctpi
spiwoks@l1ct-muctpi's password:
[spiwoks@l1ct-muctpi ~]$ uname -a
Linux l1ct-muctpi 4.19.0-xilinx-v2019.1 #1 SMP Mon Oct 28
12:26:20 UTC 2019 aarch64 aarch64 aarch64 GNU/Linux
```

But how to add user application software for access to the hardware features?

How to add ATLAS TDAQ run control software?

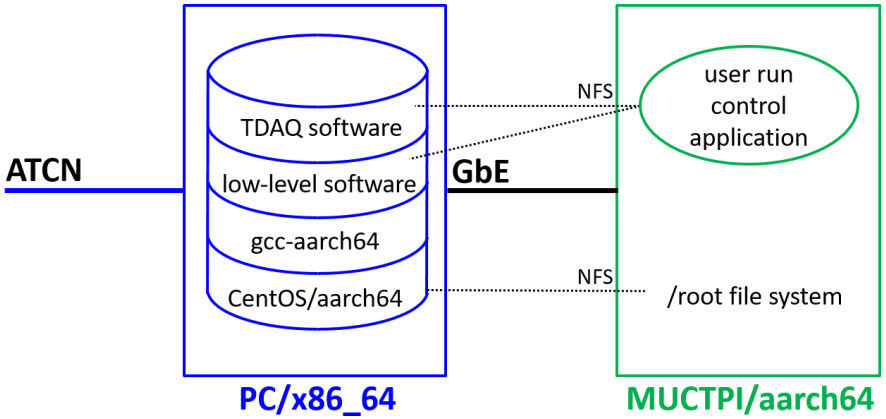
Software Development

Use cross compilation:

gcc for aarch64 from CentOS repository or developer.arm.com, or build from source (ggc.gnu.org)

Re-use work flow from previous generation:

- **Describe all registers of the MUCTPI with the bit fields, memories and FIFOs in XML file**
- **We developed software to generate from the XML file VHDL code for firmware and C++ code for software (firmware/software co-development)**
- **Registers and memories are mapped (AXI) into processor system of SoC using /dev/uio**
- **C++ code accesses /dev/uio using read/write functions (+ we developed a kernel module for DMA)**



Cross compilation works with CMake (build tool chosen by ATLAS TDAQ and for MUCTPI)

The ATLAS TDAQ software can be cross compiled in the same way:

⇒ **Build a run control application to be run on the MUCTPI**

Worldwide LHC Computing Grid (WLCG) provides builds for aarch64:

⇒ **e.g. use ROOT for histograms, graphs, etc. for monitoring**

We have a single script for setup on the host PC (development) or on the MUCTPI (deployment)

⇒ **Developer/user does not notice the "cross" environment**

Results

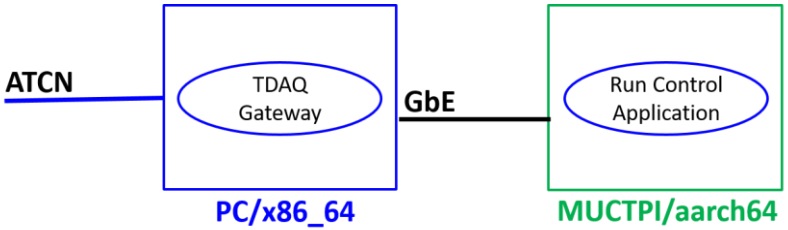
The MUCTPI C++ low-level software to access the MUCTPI hardware runs directly on the SoC, e.g. menu/test programs

Python wrappers for the software allow one to use it for interactive debugging, e.g. in test scripts

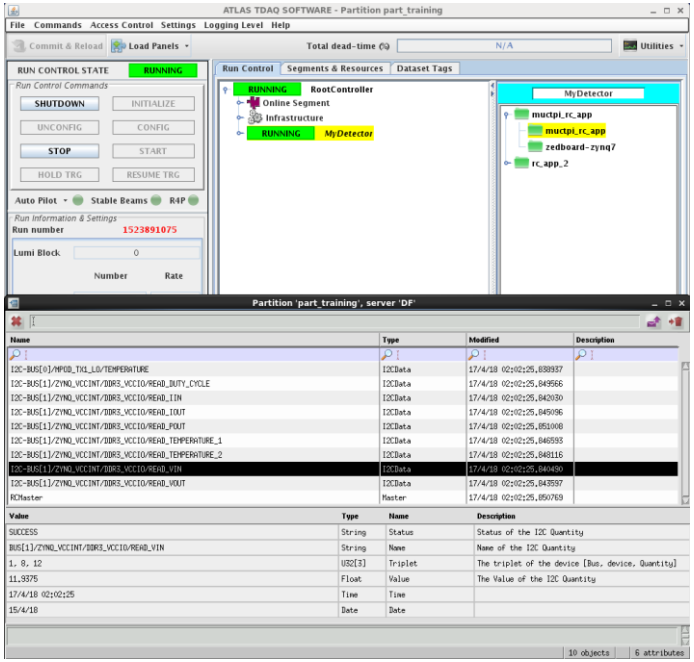
Sample python code

```
[spiwoks@l1ct-muctpi ~]$ python
Python 2.7.5 (default, Apr  9 2019, 14:35:40) [GCC 4.8.5
20150623 (Red Hat 4.8.5-36)] on linux2 Type "help",
"copyright", "credits" or "license" for more information.
>>> import Muctpi
>>> muctpi = Muctpi.Muctpi(1)
------------------------------------------------------------
---------------------
MUCTPI: INFO - MUCTPI "MUCTPI_V3_#1" opened
------------------------------------------------------------
---------------------
>>> muctpi.LEDStatusRead(0)
[0, False, False, True]
```

The MUCTPI C++ software is used by a TDAQ run control application for integration into the ATLAS TDAQ system:



Note: the TDAQ gateway running on the host PC is necessary until the MUCTPI will be allowed on the ATCN



ATLAS TDAQ GUI

Summary

- **We have successfully built a new MUCTPI using the ATCA standard and a System-on-Chip (SoC)**
- **When running CentOS on the SoC, and using cross-compilation, the software work flow is exactly as before when using VME and a Single-Board Computer**
- **There are a lot of projects for the High-Luminosity LHC (2027) in ATLAS and CMS: thousands of such systems using ATCA and SoCs**
- **SoC Interest Group is looking into common solutions: https://twiki.cern.ch/twiki/bin/view/SystemOnChip/WebHome**