EPFL

Control systems technology for a Tokamak Plant

Cristian Galperti

Swiss Plasma Center

09.02.2023

EPFL

Summary

- General overview of tokamak control engineering
- Technologies
- TCV examples

Tokamak machines from the control point of view

Galperti Cristian

- Tokamaks are part of the so called “**big-physics**” devices that encompass many technologies
- They run in a “**plant**” that takes 24/7 care of systems like temperature, vacuum, security etc.
- They are still mostly “**pulsed**” devices that have physics operations for often (far) less than 10% of wall-clock time
- During not operation periods control usually is delegated to slow unmanned plant control systems
- During these pulses, **heightened** activity is common and real-time feedback control is mandatory to control and sustain the plasma and to achieve scientific results

Slow control system architecture

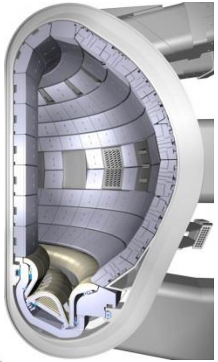
Galperti Cristian

Keep ALL systems necessary for running all components of the plant (including water flows, air temperatures, servicing requirements all the way to machine shot preparation)

HMIs

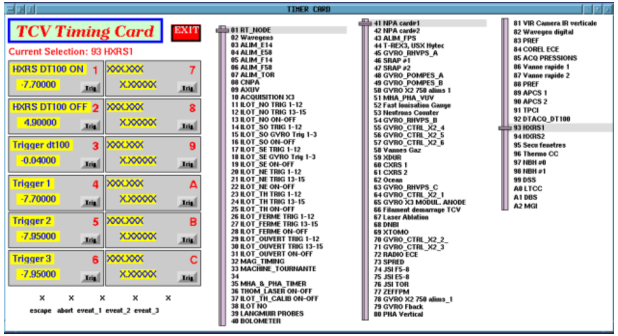
Plant "live" database

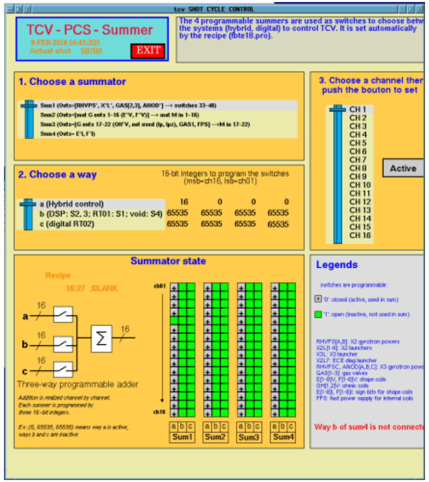
Usual update rate < 10 Hz



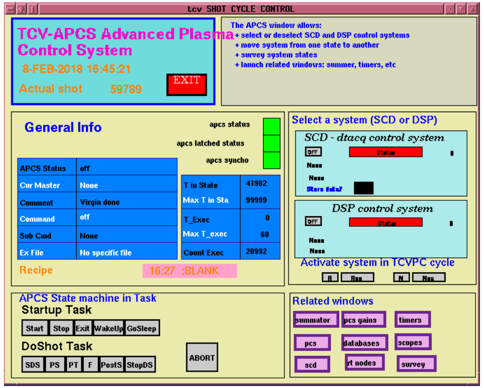
"24/24H" PLANT

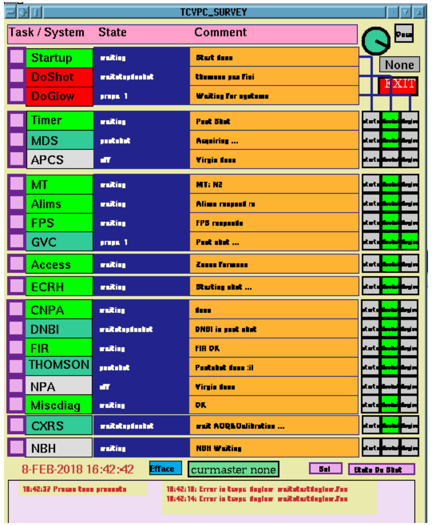
TCV example





VISTA “live” database





Swiss Plasma Center

Galperti Cristian

OPCUA protocol

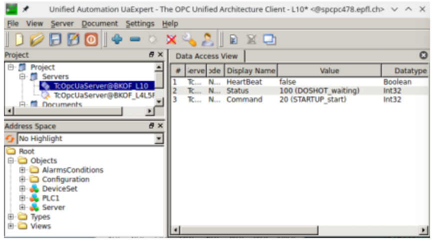
Galperti Cristian

Production
live DB client

OPCUA
connection

OPCUA
server
running on a
local control
system

OPCUA
connection



GUI based
debug client

OPCUA
connection

```
from opcua import Client
from opcua import ua

def sendcommandloop(server):
    client = Client(server)
    try:
        client.connect()
        # Command dictionary from the
        #cmddict=slaveint.get_commanc
        # direct nodeid call method (
        Command=client.get_node("ns=4
        usercmd = 0;
```

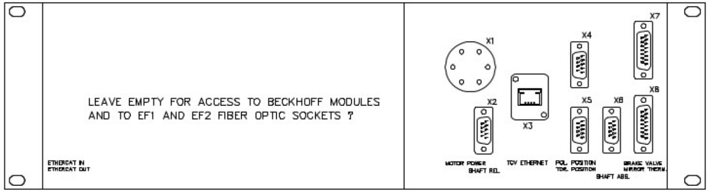
Script based
debug client

- Vendor and platform independent
- Connection based (server/client model)
- Secured (encryption and authentication)
- Available from many control equipment manufacturer as well as open source developers

https://opcfoundation.org/about/what-is-opc/

A modern integrated subsystem

Galperti Cristian

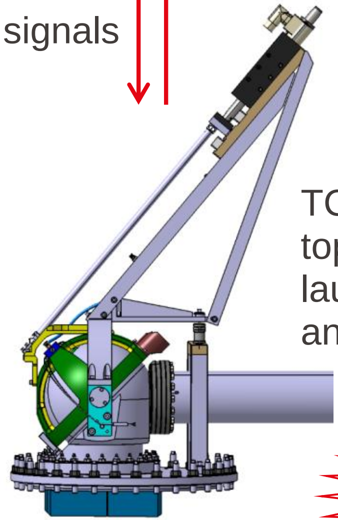


Launcher local controller

Commands and status exchange via OPCUA protocol and OPCUA/Vista bridge

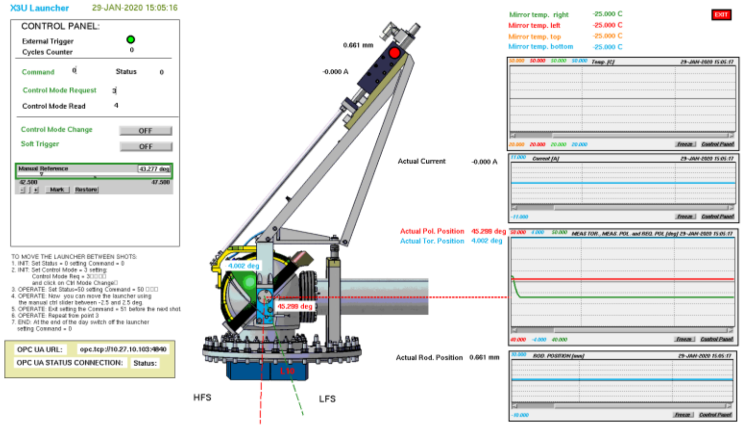
Vista DB

Field signals



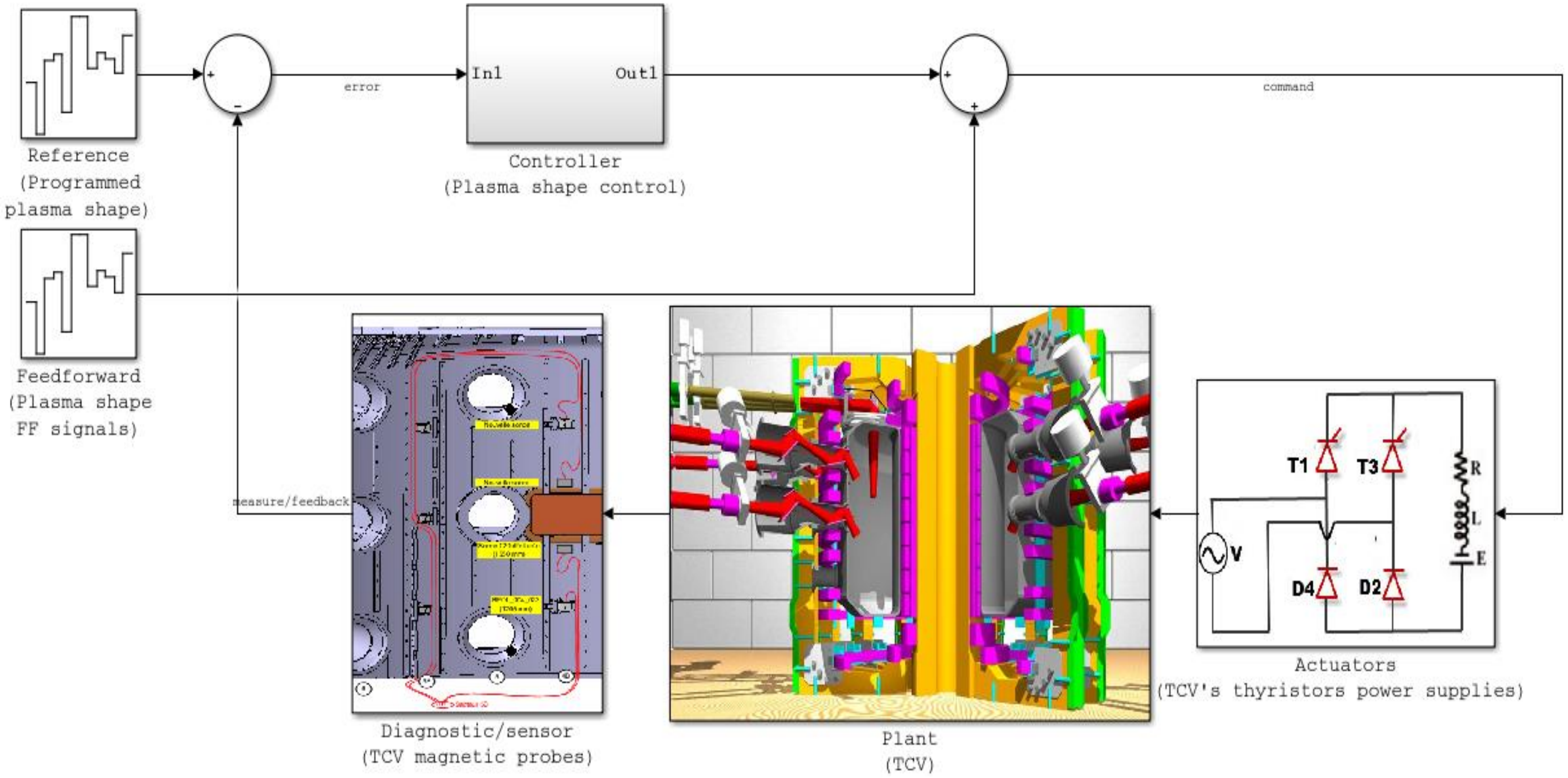
TCV ECRH top fast-launchers L10 and L11

Launcher control panel

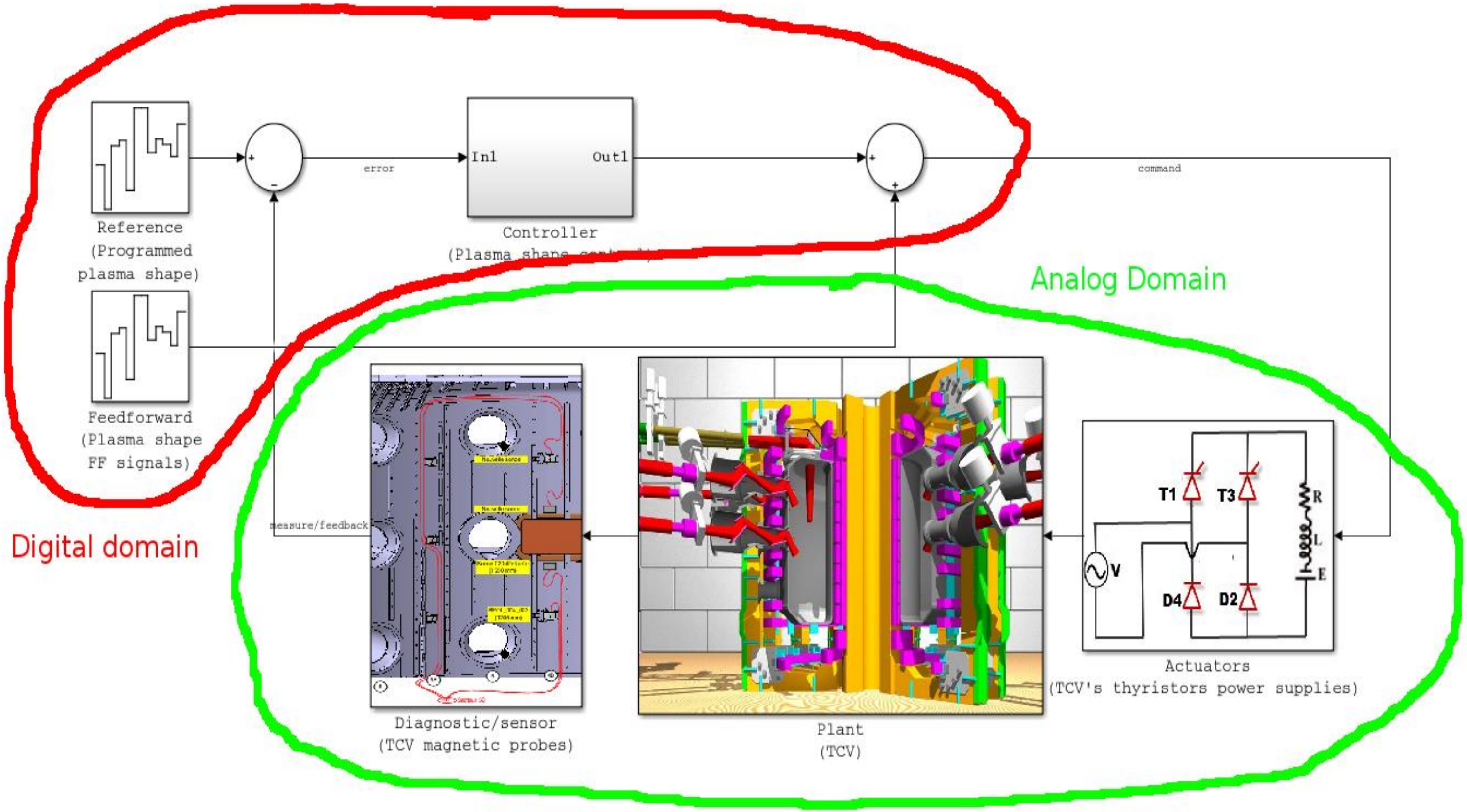


TOUR Available

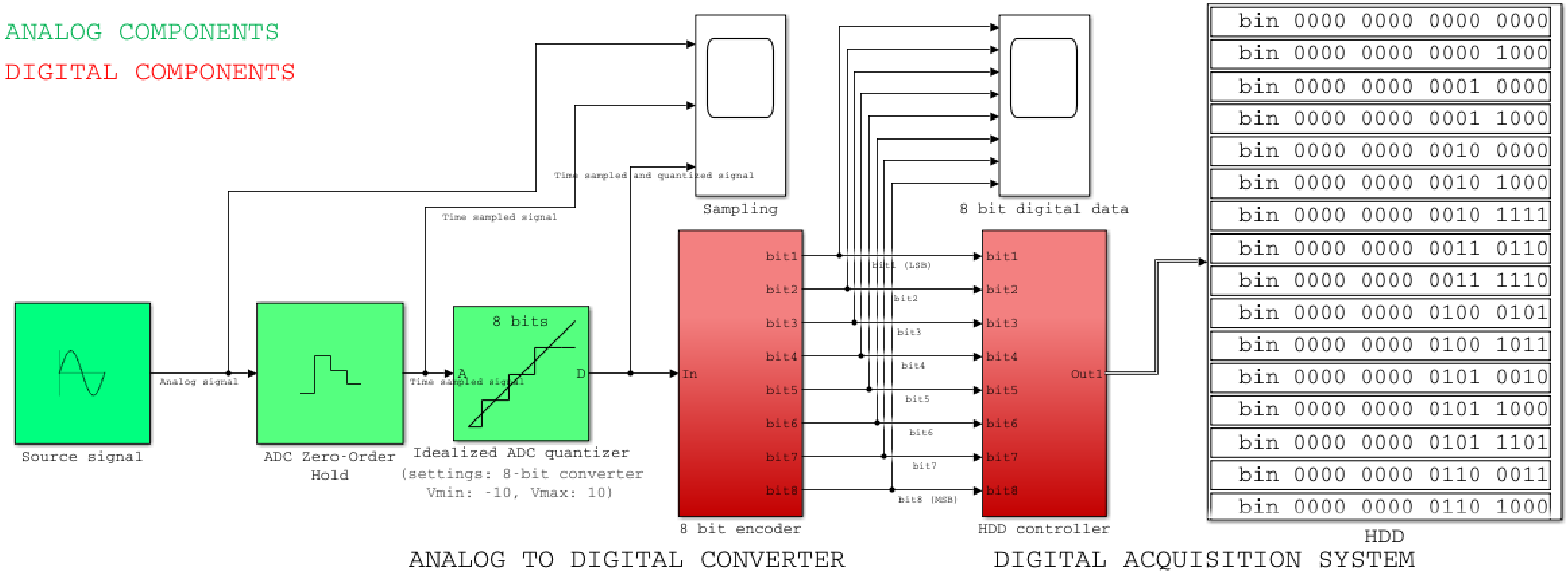
Digital real-time control of tokamaks



Digital real-time control of tokamaks



Analog to digital conversion (1)



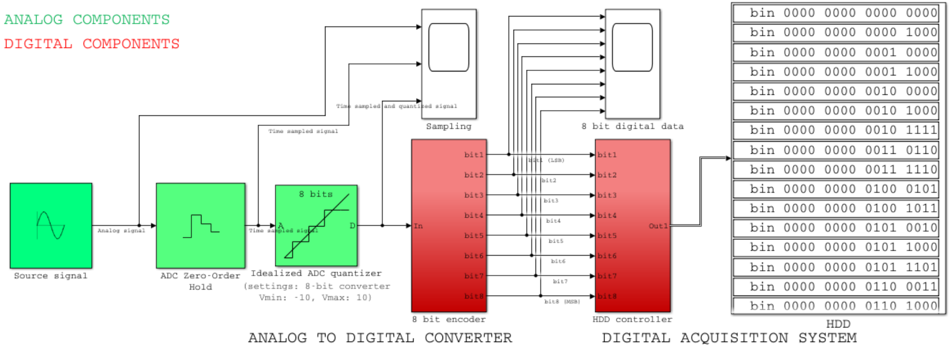
Swiss Plasma Center

EPFL

Analog to digital conversion (2)

Common problems and pitfalls

- Wrong input signal level
  - too low: quantization noise -> open loop
  - too high saturation and signal clip -> open loop
  - Need adjustable input amplifiers
- Wrong input signal speed
  - Alias effect and signal and noise spectrum pileup -> poor controller performances
  - Need correct input analog filtering



- Wrong sampler timing
  - Loss of synchronicity with the rest of the plant
  - Wrong storage timebase -> control system resimulation almost impossible

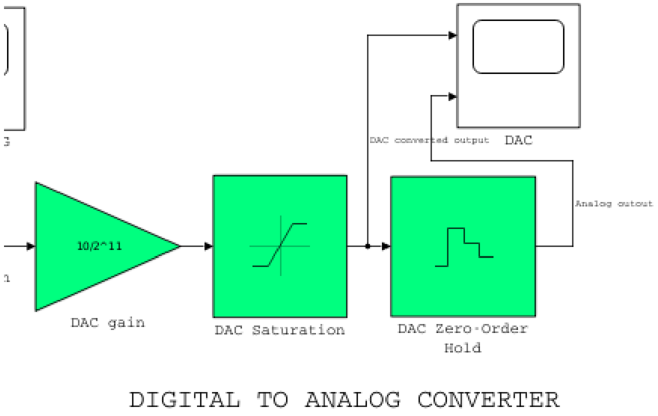
Swiss Plasma Center

EPFL

Digital to analog conversion

Common problems and pitfalls

- Wrong output signal level
  - too low: quantization noise -> open loop
  - too high saturation and signal clip -> open loop
  - Need well designed control math and DAC – actuator matching
- Too poor resolution / too high offset
  - Sometimes poorly designed control-actuator chains may lead to too little control resolution / too high offset

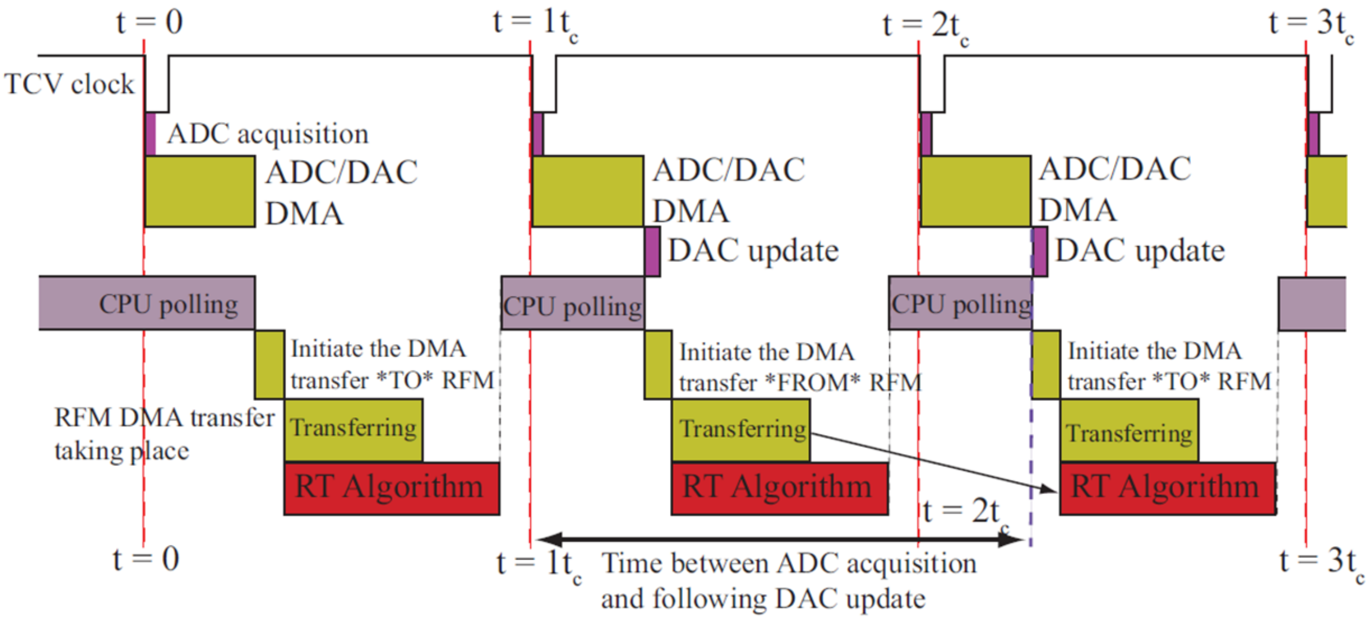


Suggested common solution: use direct digital control – actuator interfaces whenever it is possible

Swiss Plasma Center

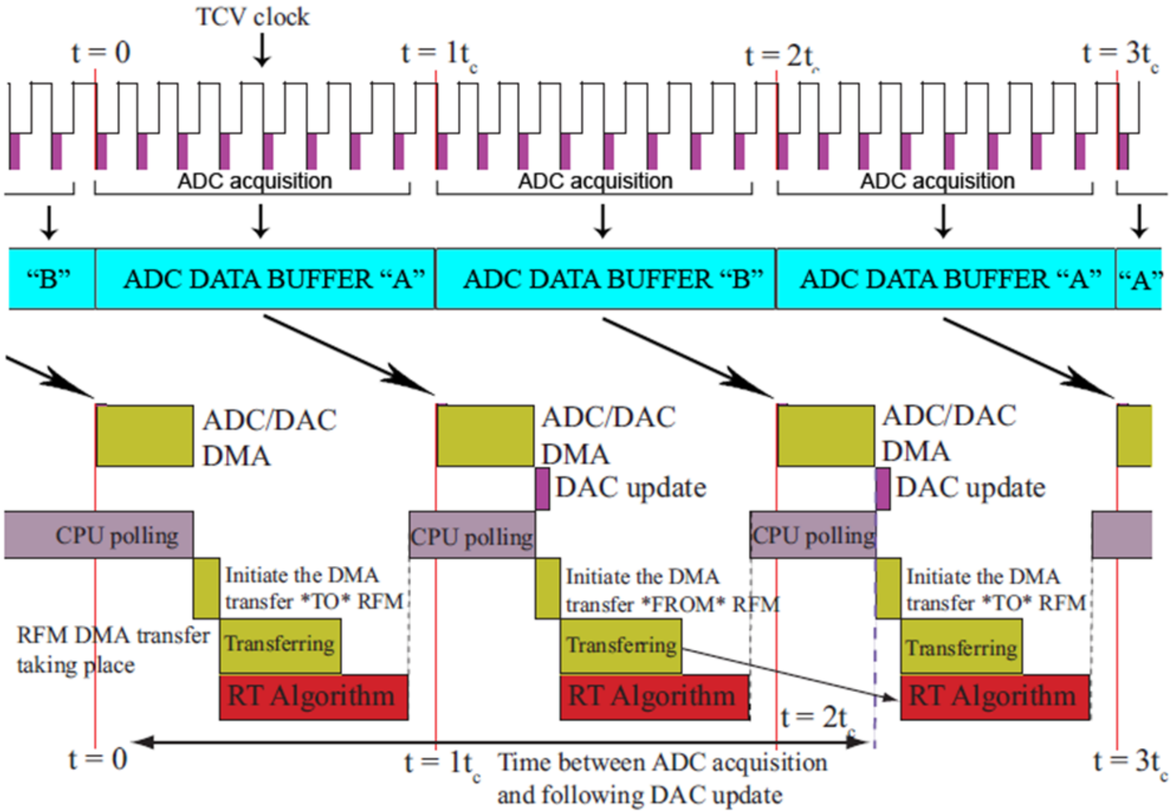
Time chart of a real-time computer

Typically, a real-time digital control hardware cyclically executes the control code triggered by a timing system. Usually the trigger is common with the ADCs (but not mandatory). Multiple concurrent processing and/or data transfers may happen (especially with multi cores CPUs)



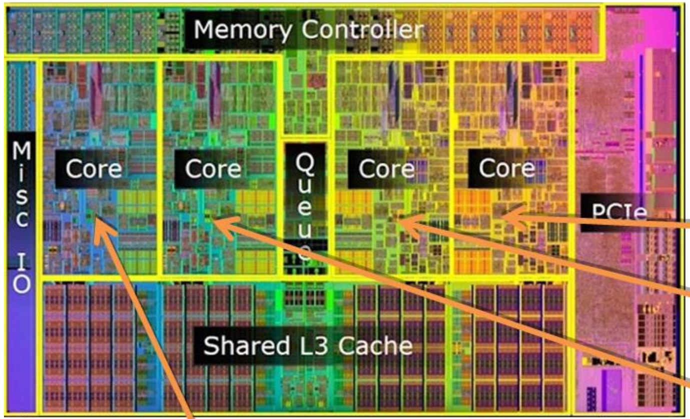
From J. I. Paley et al. Real Time Conference (RT), 2010 17th IEEE-NPSS

Time chart of a real-time oversampling computer

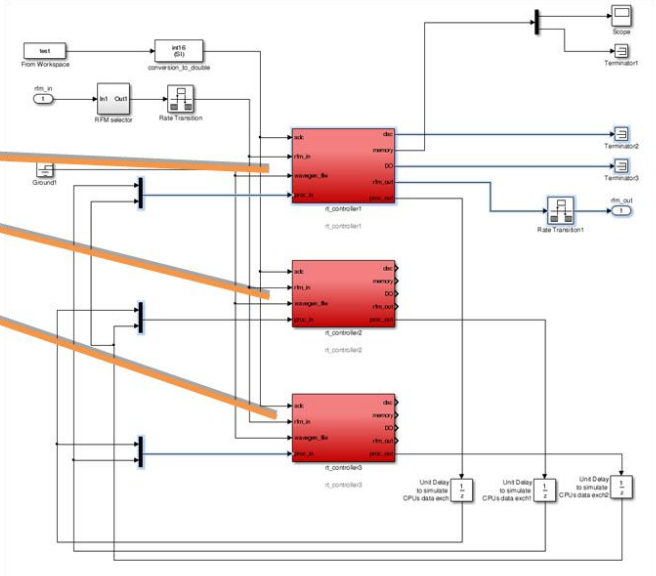


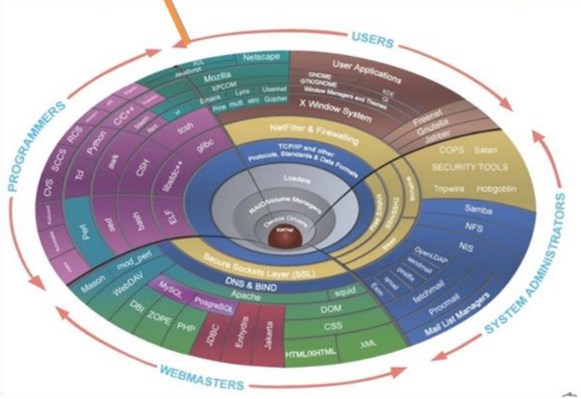
- Basically a (double) buffer is inserted between the ADC digital outputs and the DMA source endpoint.
- ADC data are now tranferred in bursts into the host PC memory. This frees the sampling frequency upper bound.

Central processing units (CPUs)



Intel I7 architecture



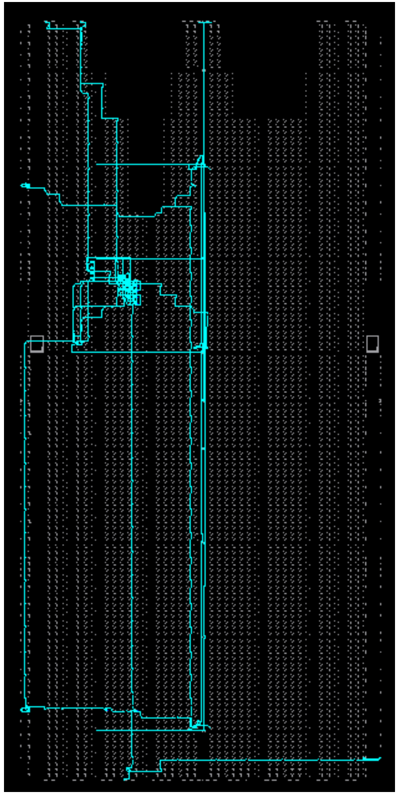


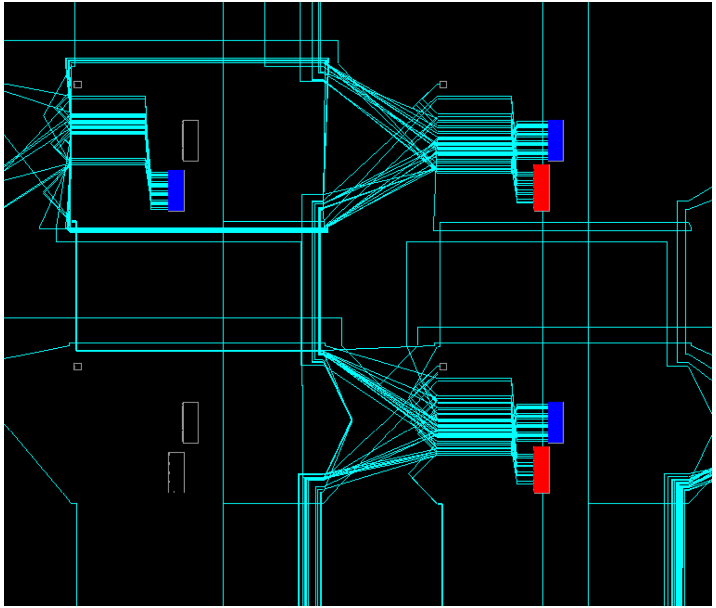
All complex control systems nowadays run on multi-core CPU architectures, a lot of Linux based RT computer tend to put the kernel on the first core, leaving the other free for RT computation (but highly s.o. dependent)

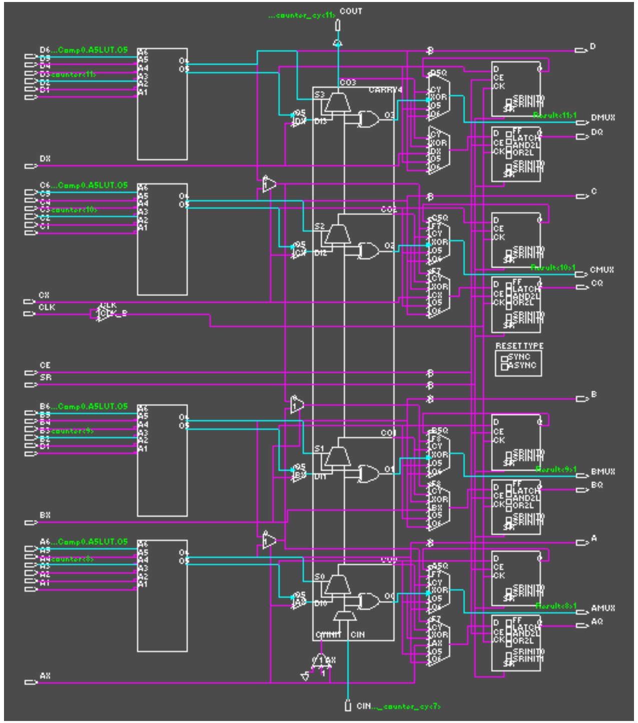
Galperti Cristian

Swiss Plasma Center

FPGAs





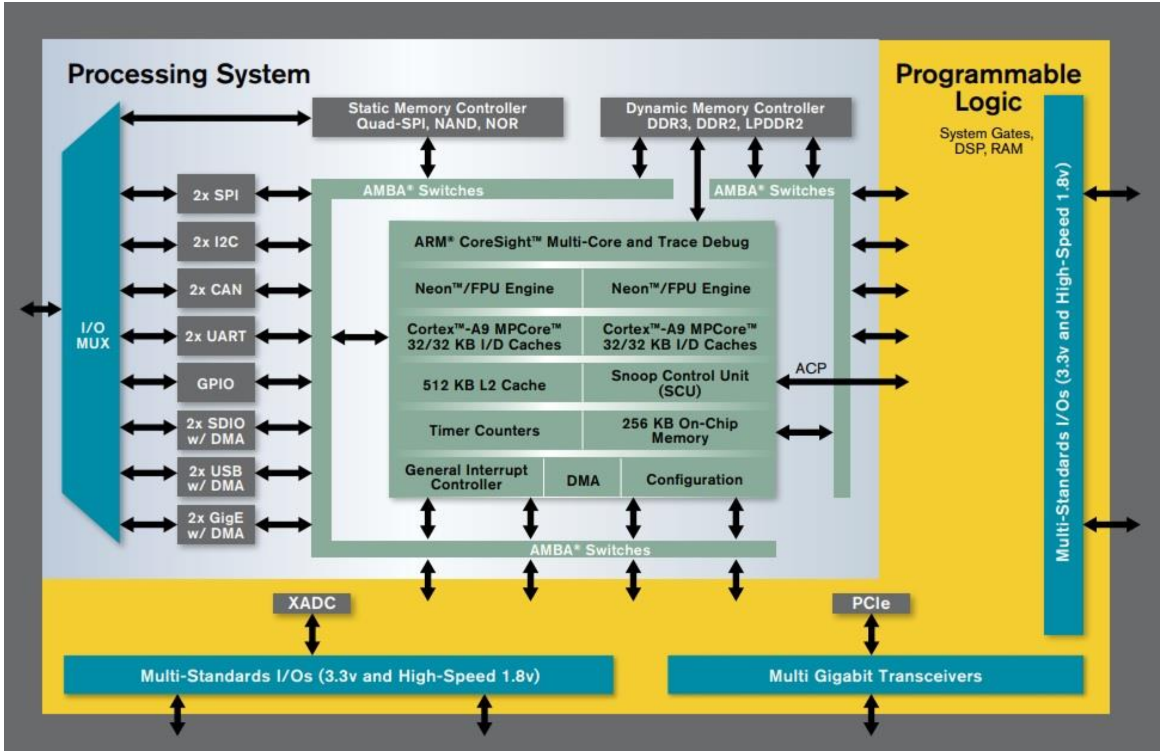


FPGAs (Field Programmable Gate Arrays) chips allow an incredible degree of hardware design flexibility at viable costs. But they are usually quite difficult to program/commission. Here an example of a routed clock DPLL for TCV.

Galperti Cristian

Swiss Plasma Center

SoC (System on chip)

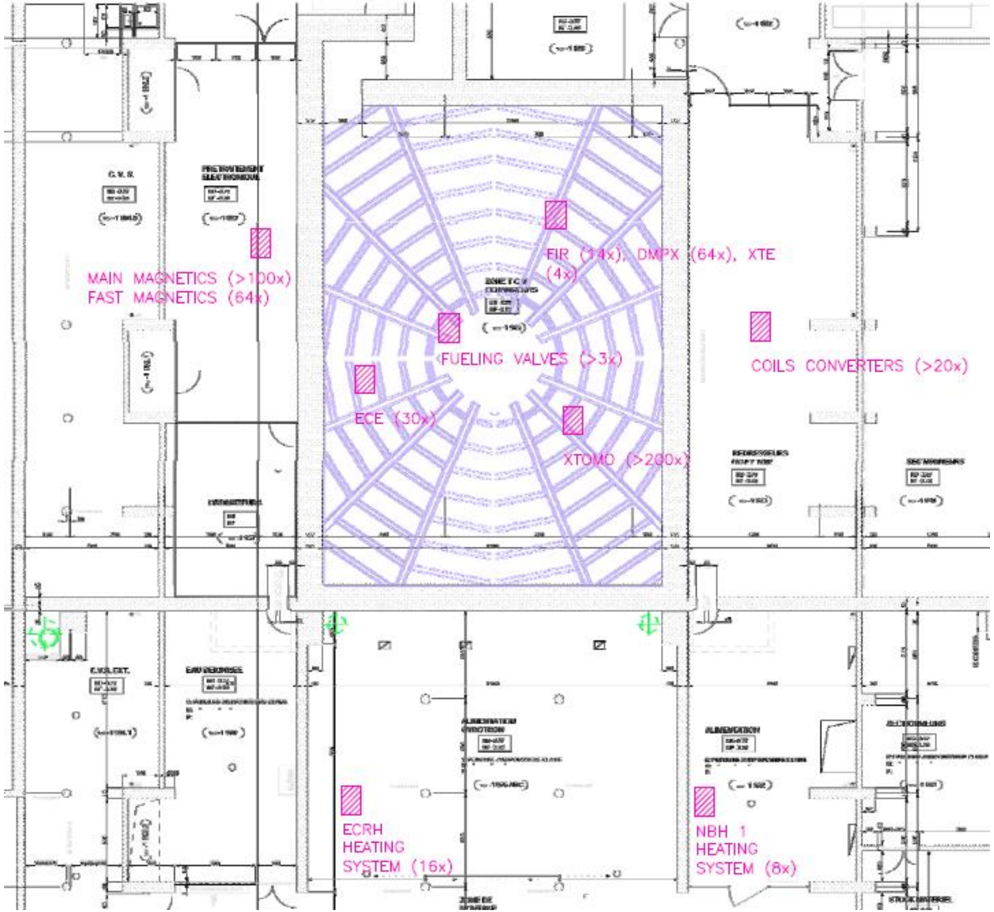


System on chip combine support for complex code execution (e.g. kernels) given by CPUs to custom circuitry flexibility granted by FPGA on the same chip. The may also embed specific standard I/O modules and/or mainstream bus endpoints like PCIe. They are the primary choice nowadays for high performance control embedded systems, even in fusion.

Xilinx ZINC7000 SoC
https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

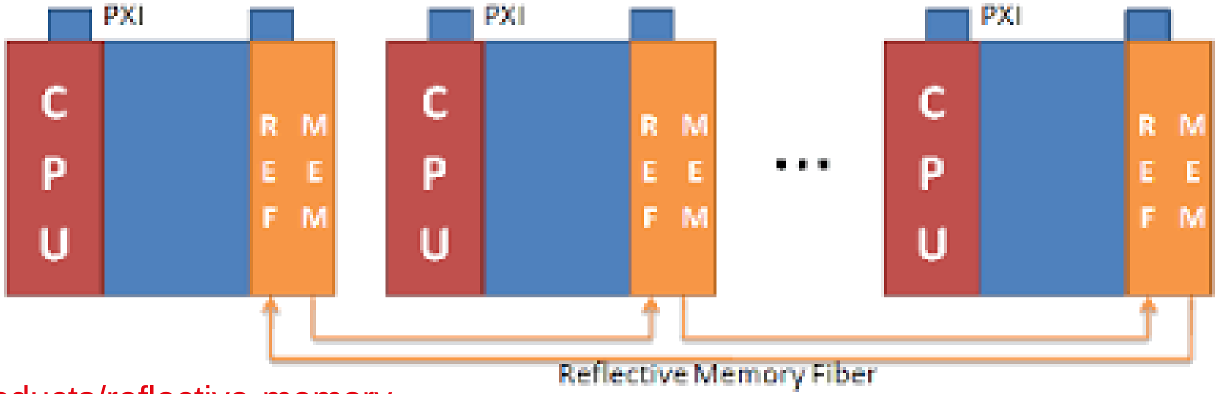
TCV's control system layout, demand of real-time networks

- Tokamaks are spatially distributed experiments, hence they require inherently distributed control systems
- For data distribution among systems -> need for realtime data networks
- For time distribution among systems -> need for distributed timing systems



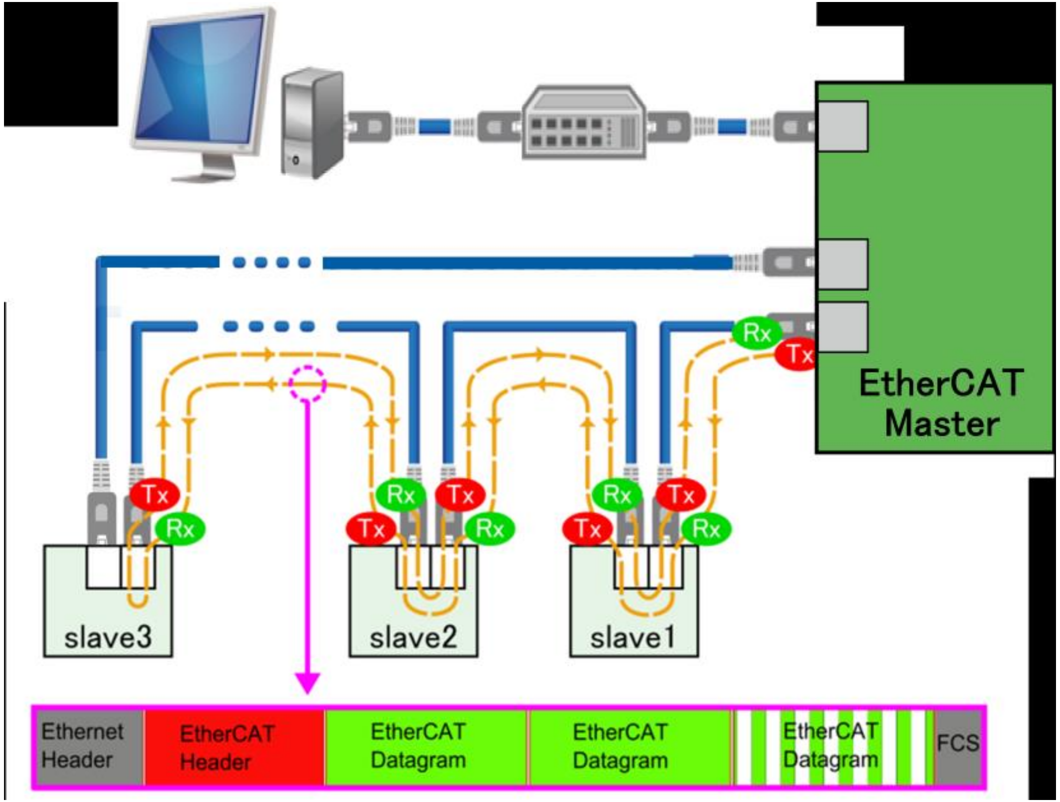
Reflective memory networks

Serial communication channels have to guarantee the minimum latency time to transfer data from one point to the next. RFM is a hardware/software agnostic fast (>2gbps) fiber optic link which automatically synchronizes memories between several hosts.



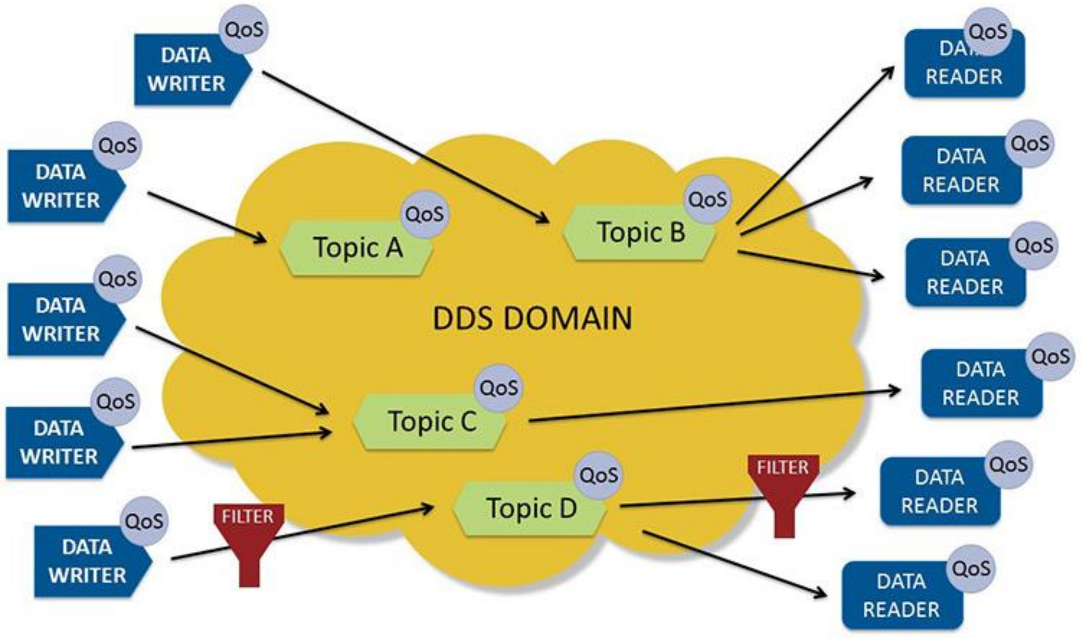
https://www.abaco.com/products/reflective-memory

EtherCAT networks



EtherCAT (Ethernet for Automation and Control Technology) is an Ethernet based control network. The goal during development of EtherCAT was to apply Ethernet for automation applications requiring short data update times (also called cycle times; ≤ 100 μs) with low communication jitter (for precise synchronization purposes; ≤ 1 μs) and reduced hardware costs. (https://en.wikipedia.org/wiki/EtherCAT)

DDS networks (publish / subscribe concept)

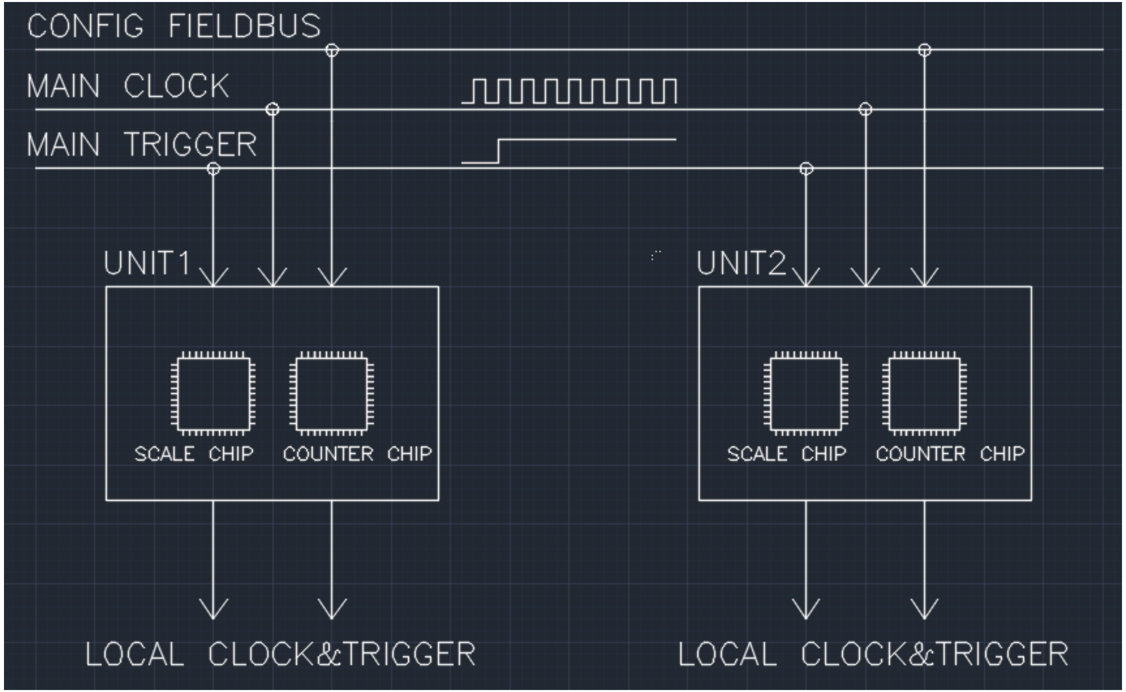


DDS (Data Distribution Service) is a middleware protocol and API standard for data-centric connectivity.

- Data is published by data-writers and subscribed to by data readers.
- Highly scalable, only data dependent.

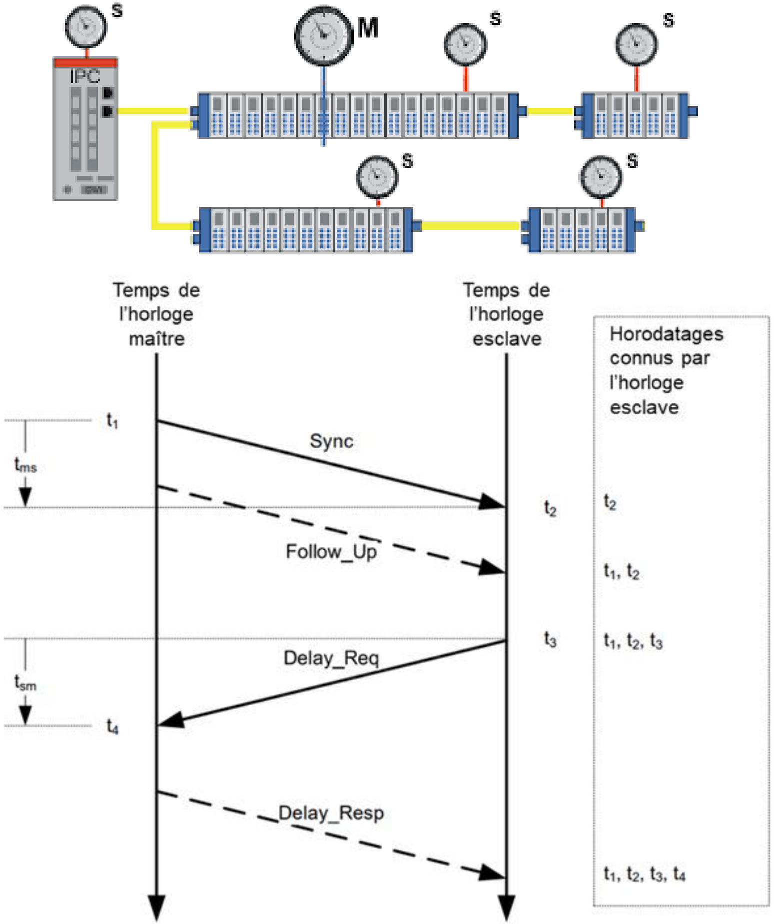
www.dds-foundation.org

Distributed timing systems, hardwired



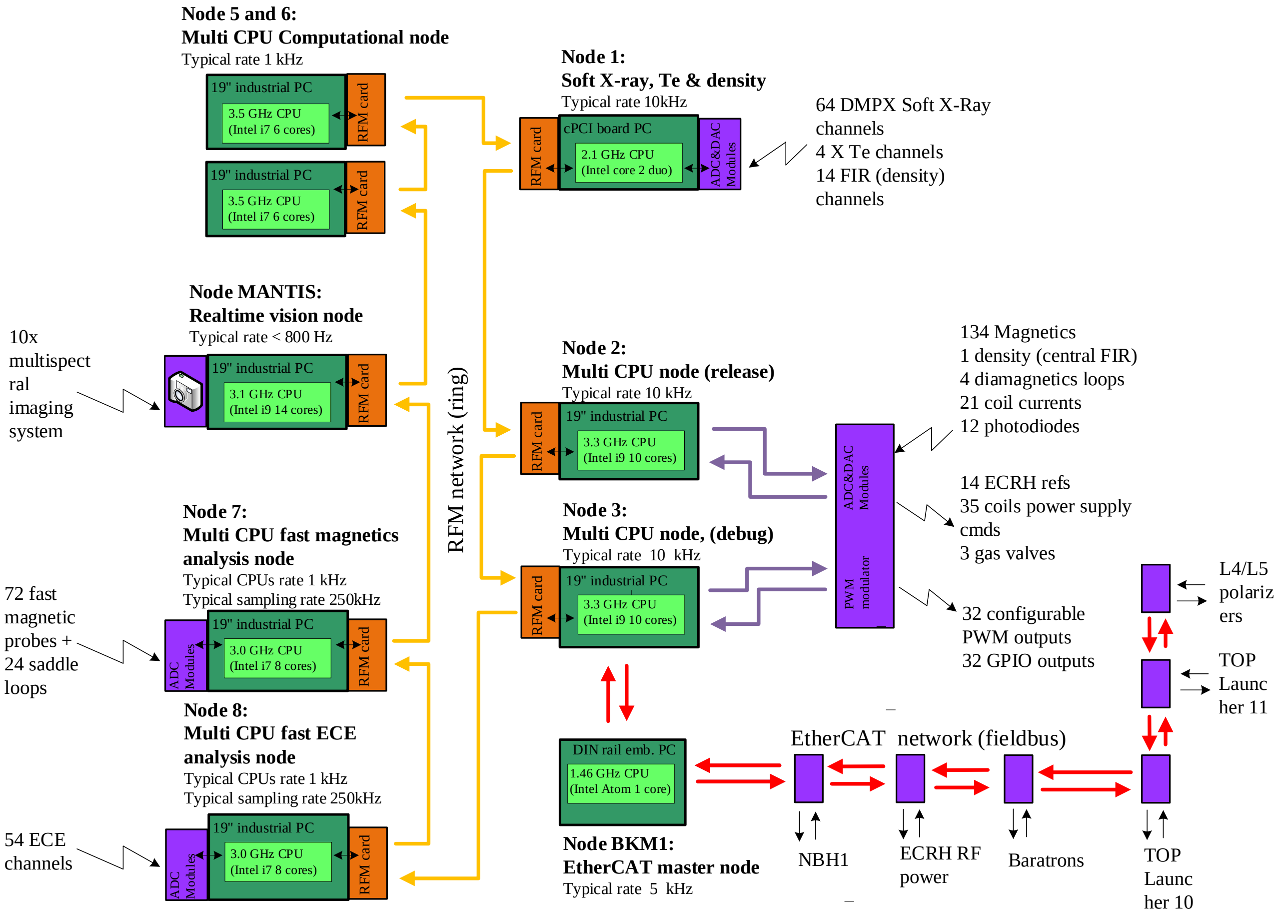
- First sync concept deployed on tokamaks
- Pros:
  - architecturally simple.
  - maintainable and upgradeable for a long time
- Cons:
  - Best sync accuracy: 1 us
  - No 24h/24 sync (only pulsed sync)
  - No sync groups
  - No industrial support

Distributed timing systems, network based



- Current industry & research standard
- Pros:
  - Best sync accuracy: 1 ns (research, WhiteRabbit project), 100 ns (industry, PTP and EtherCAT DC).
  - 24h/24 sync and sync groups out of the box
  - Strong industrial and research support
  - Easy deployable and debuggable (it is like a computer network)
- Cons:
  - Still low widespread availability of control equipment based on them.
- http://white-rabbit.web.cern.ch/
- https://en.wikipedia.org/wiki/Precision_Time_Protocol

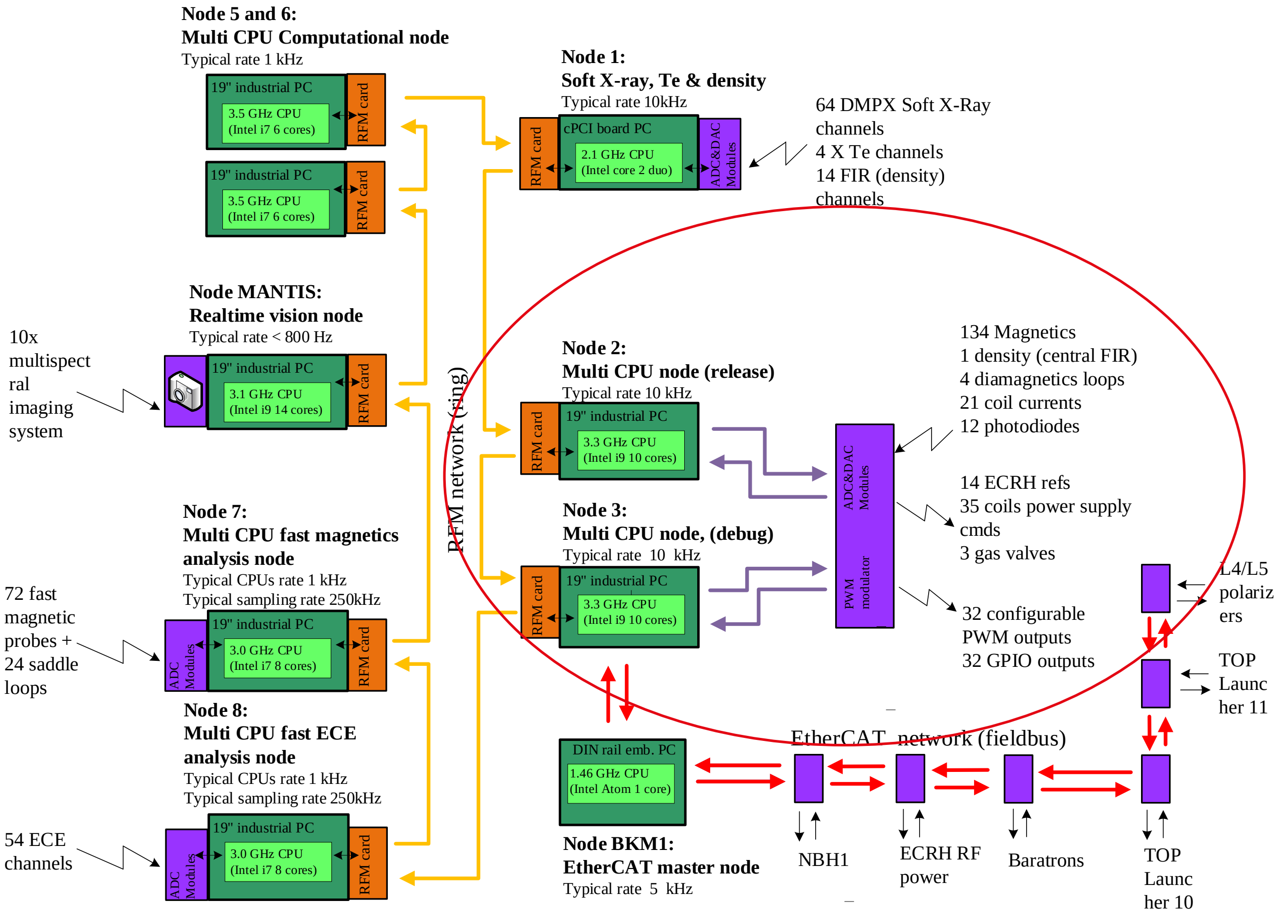
TCV digital distributed control system



- Various types of interconnected control subsystems:
  - Low latency systems for hard real time control at fastest rate
  - Oversampling systems for fast diagnostics acquisition followed by complex real-time analysis algorithms
  - Multi-core computational systems for CPU hungry codes
  - Real-time vision nodes for vision in the loop systems
- EtherCAT for fast, fexible, distributed and cost effective I/O interconnection

Swiss Plasma Center

TCV digital distributed control system



- Central main plasma control system duplicated on two identical machines

Galperti Cristian

Swiss Plasma Center

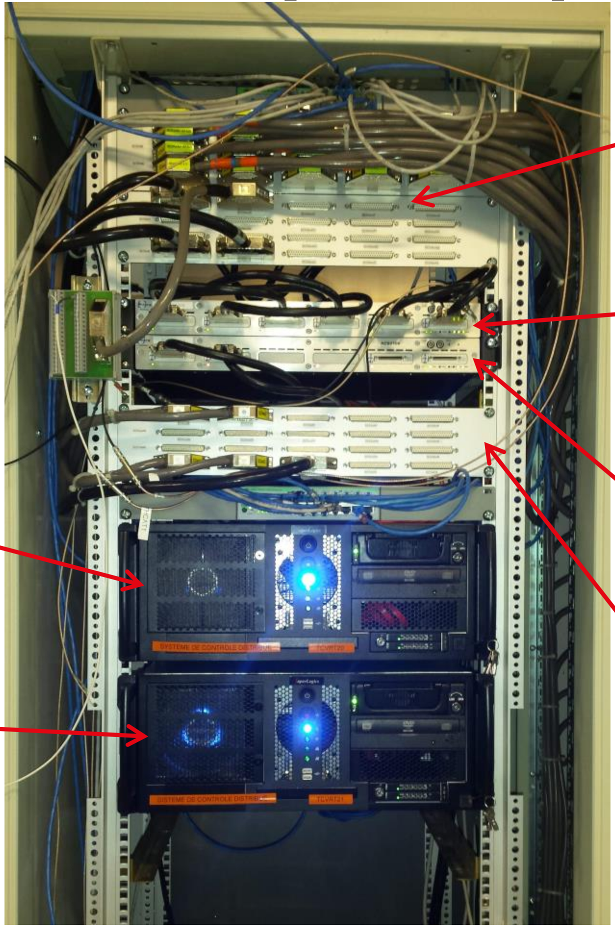
TCV main control node (node 02)

Galperti Cristian

TOUR
Available

Node 02 A
Industrial PC
(tcvrt20.crpp.tcv)

Node 02 B
Industrial PC
(tcvrt21.crpp.tcv)



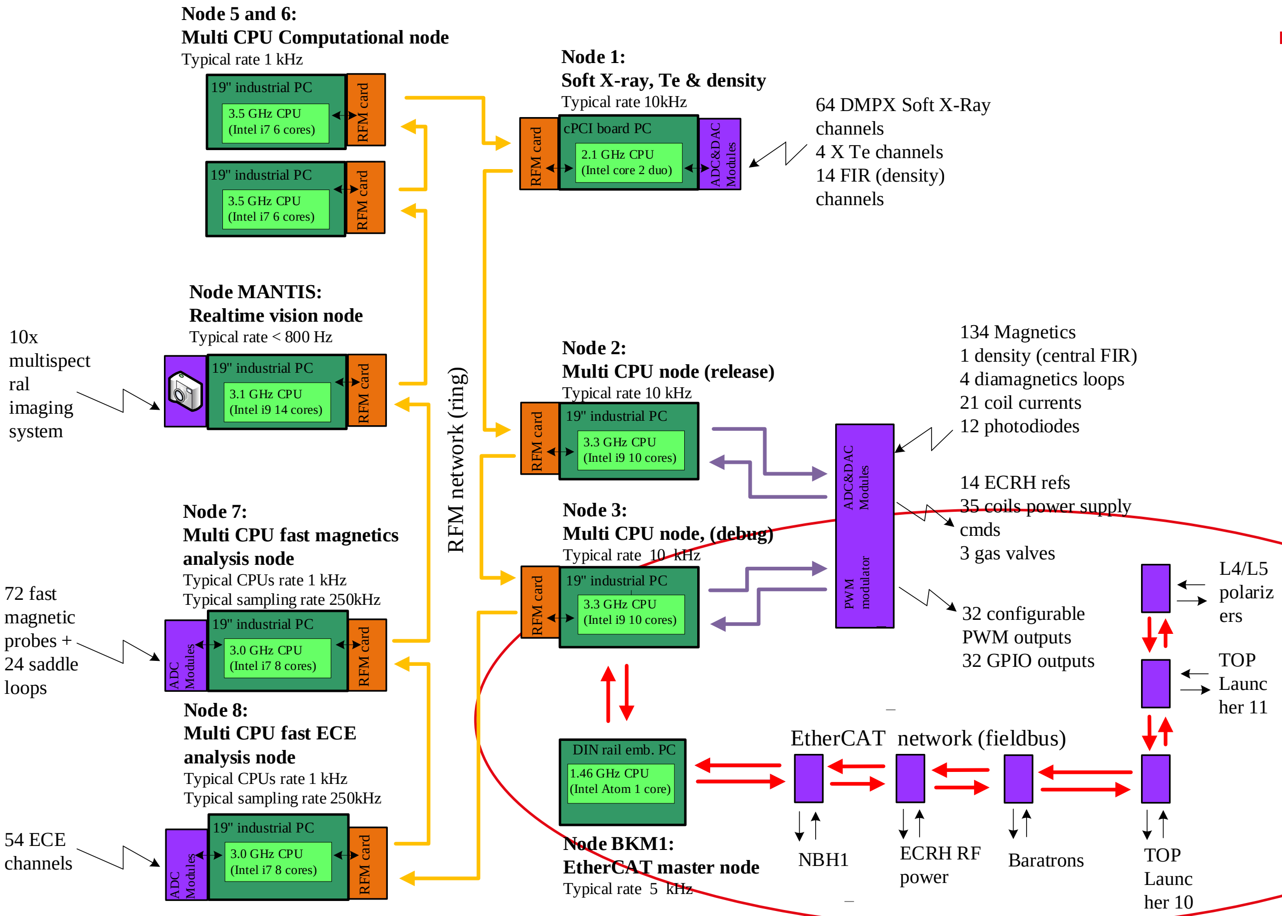
Inputs patch panels

ADC system (acq2106_076 192 channels up to 1 MHz))

DAC + PWM system (acq2106_079, up to 128 channels and 32 PWMs)

Outputs patch panels

TCV digital distributed control system



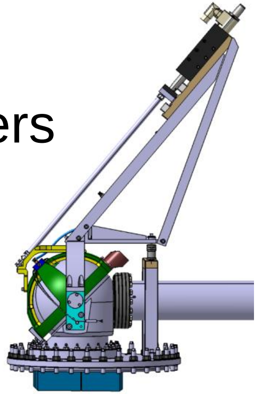
- Central main plasma control system is interfaced with a realtime EtherCAT network for flexible I/O interconnections, and direct digital drive of actuators

Direct digital drive of actuators, examples

- Linux EtherCAT master stack
- All digital realtime I/O with actuators
  - ECRH_L10 mirror angle command'
  - ECRH_L10 mirror angle measure
  - F8 coils voltage command
  - F8 coil voltage measure
- https://esd.eu/en/products/ethercat-master

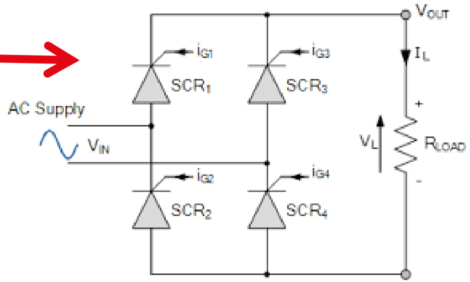
ECRH waves top launchers

https://www.beckhoff.com/en-en/



Main coils controlled power supplies (TBC)

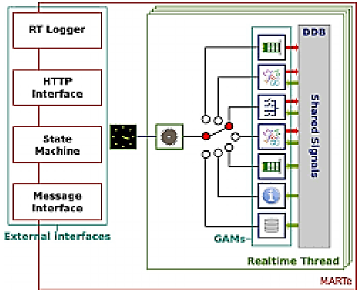
https://imperix.com



TCV realtime digital control system, the three pillars model

SIM – EXEC path

MATLAB/Simulink + SCDDS framework (Control code development and simulation)



MARTe2 (Real time control framework)

SIM – DB path

EXEC – DB path

MDSplus (shot parametrization and data archiver)

EPFL

MDSplus data archiving system

Galperti Cristian

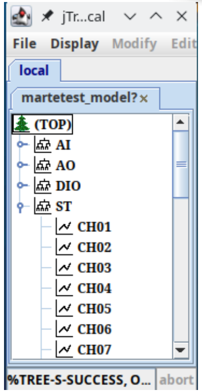
Model DB

Pulse DB creation before the experiment

Live pulse DB

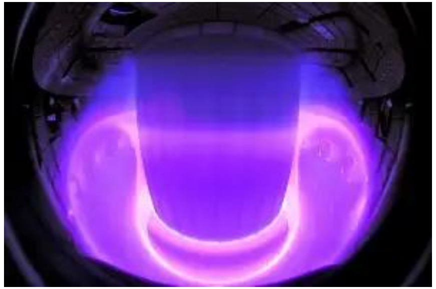
Pulse DB archiving after the experiment

Archived pulse DB

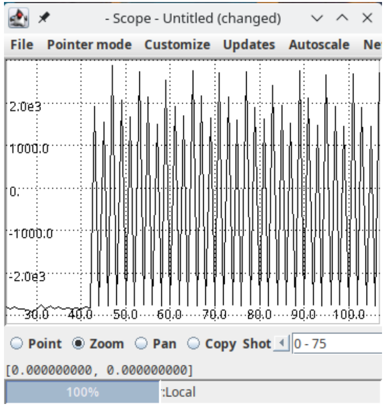


Pulse config

Pulse execution



Offline analysis



Swiss Plasma Center

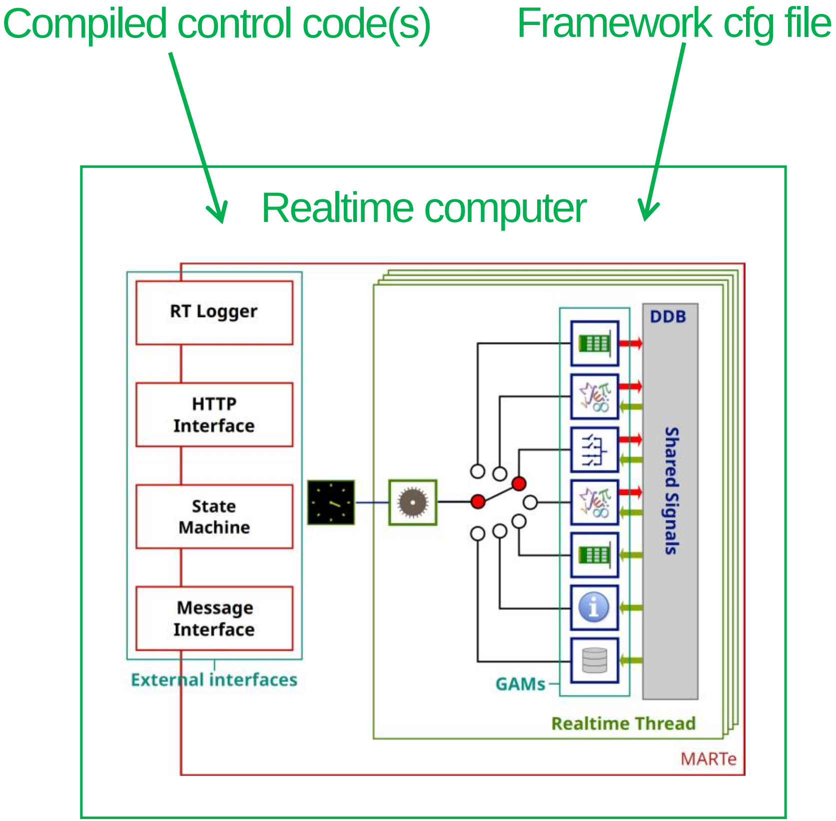
www.mdsplus.org

MARTe2 framework

Galperti Cristian

MARTe2 is a C++ software framework conceived to help building and running realtime applications. TCV entirely switched to it from last year.

- Standardization and modularization of realtime control applications
- Multithread/multicore applications natively supported
- Component to load Simulink generated code, with introspection
- Components to read/write MDSplus entries
- State machine / messages interface
- Extensive logging



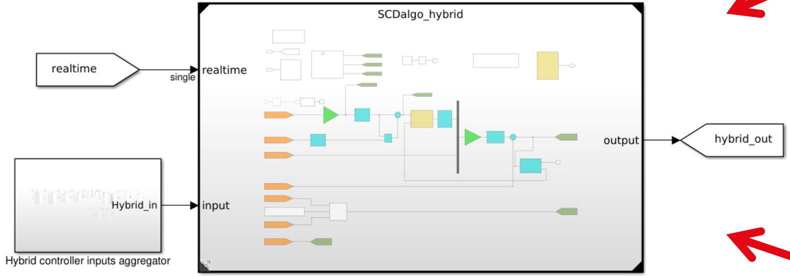
https://vcis.f4e.europa.eu/marte2-docs/master/html/

Simulink (and its paths)

Galperti Cristian

MATLAB/Simulink®

Simulink modeled control system part (e.g. algorithm)

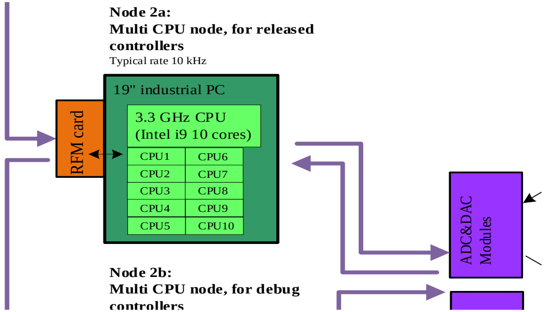


SIM – EXEC path

SIM – DB path

Real-time control systems

PLANT



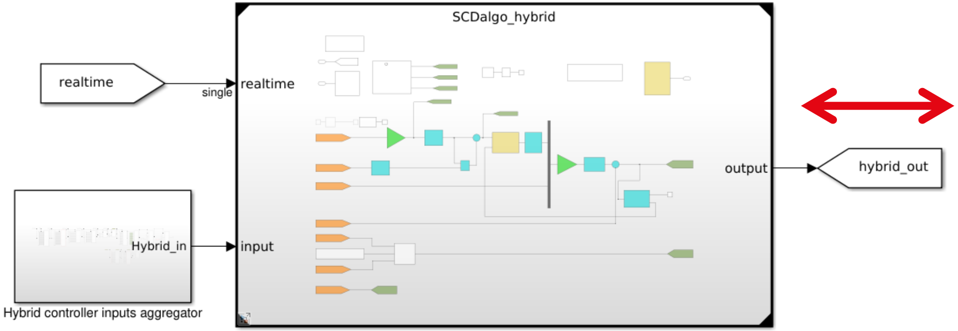
Database systems

EPFL

The SCDDS framework

Simulink Control Development and Deployment Suite

Simulink modeled control system part (e.g. algorithm)



SCDDS framework, a MATLAB object oriented framework to handle and interface control code

```
classdef (Abstract) SCDDSclass_algo
    %SCD algorithm handling object
    %   The class holds all information and
    %   methods for handling a Simulink
    %   algorithm

    properties (SetAccess = private, Hidden=false)
        modelname            % Name of the model
        modelslx             % slx model file name
        folder               % folder containing algorithm
        datadictionary       % Name of the used data dictionary
    end
```

```
%%
classdef SCD_algo < SCDDSclass_algo

end
```

```
function [obj] = SCDalgoobj_hybrid()

%% Hybrid controller core algorithm
obj=SCD_algo('SCDalgo_hybrid');

%% Timing of the algorithm
obj=obj.settiming(-4.5,1e-4,3);

%% Tunable parameters structure name
obj=obj.addtunparamstruct('SCDalgo_hybrid_tp');

%% Tunable parameters
obj=obj.addparameter(SCDclass_mdsparmatrix
obj=obj.addparameter(SCDclass_mdsparmatrix
```

Swiss Plasma Center

gitlab.epfl.ch/spc/scdds
LGPL open source license

EPFL

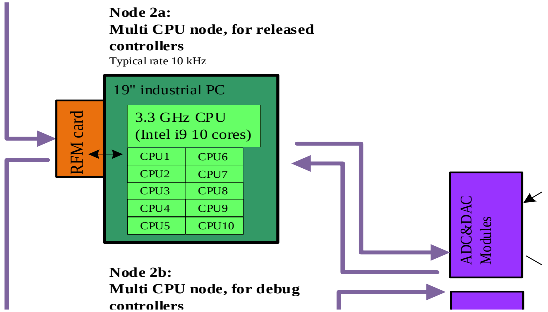
SCDDS framework, overview

Set of instantiated objects in MATLAB representing the control system

```
function [obj] = SCDalgoobj_hybrid()

%% Hybrid controller core algorithm
obj=SCD_algo('SCDalgo_hybrid');

%% Timing of the algorithm
obj=obj.settiming(-4.5,1e-4,3);

%% Tunable parameters structure name
obj=obj.addtunparamstruct('SCDalgo_hybrid_tp');

%% Tunable parameters
obj=obj.addparameter(SCDclass_mdsparmatrix
obj=obj.addparameter(SCDclass_mdsparmatrix
```

- Code generation and deployment
- MARTe2 cfg files generation and deployment

Real-time control systems (MARTe2 based)



- Parameters and waveforms retrieval for code simulation
- Parameters deployment for shot execution

Database systems (MDSplus based)

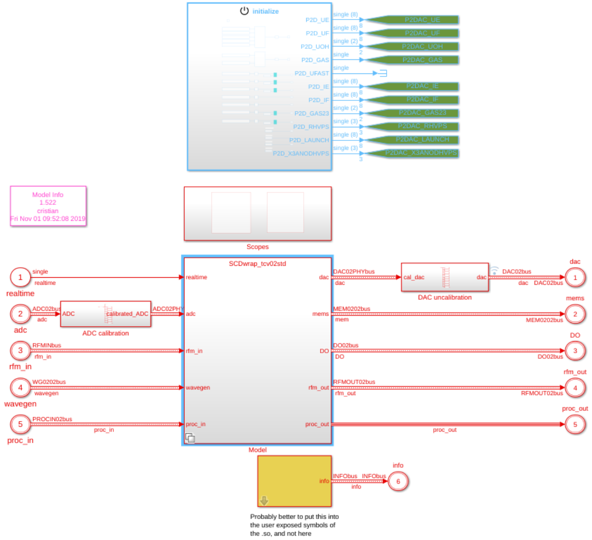
Swiss Plasma Center

EPFL

SCDDS, code generation from Simulink

Galperti Cristian

Simulink algorithm with tunable parameters



Simulink coder with enabled CAPI + C compiler (gcc or icc)

Linux shared library

Introspective interface for I/O ports

Introspective interface for tunable parameters

Introspective interface for internal states

Introspective interface for internal signals

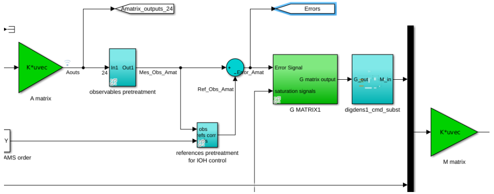
SCDDS simulation and MARTe2 execution workflow

Galperti Cristian

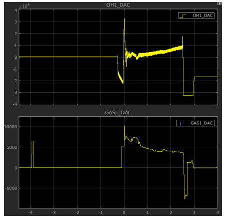
MDS+ parameters and waveforms DB

SCDDS (MATLAB) parameters and waveforms loaders

Simulink model under SCDDS framework



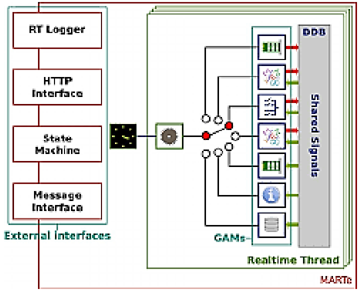
Simulink simulation



Automatic code generation

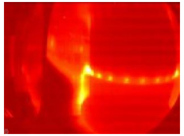
=

MARTe2 (C++) parameters and waveforms loaders



MARTe2 control system

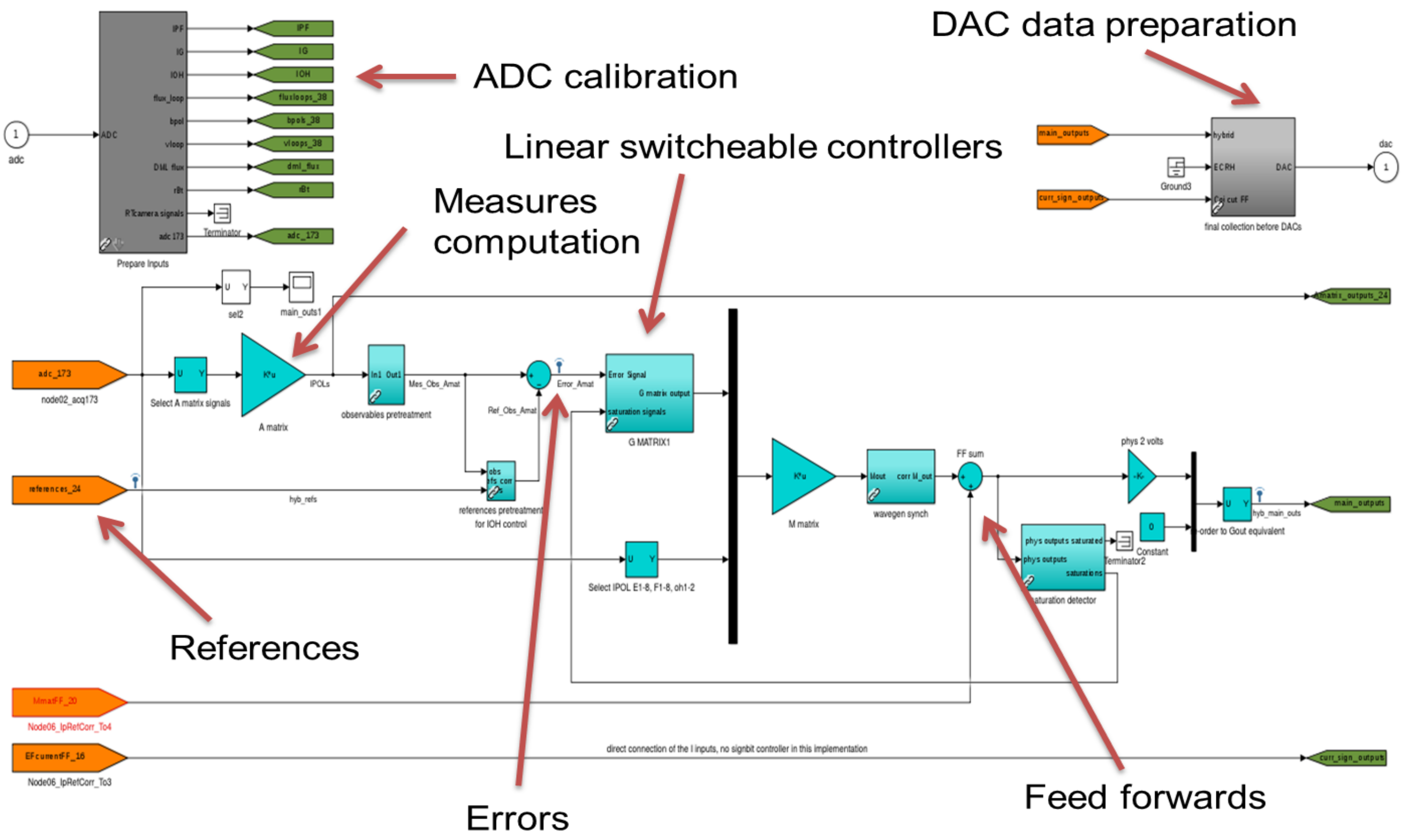
RT execution



MDS+ shot data

Swiss Plasma Center

TCV standard plasma control code



TCV Simulink plasma control simulation example

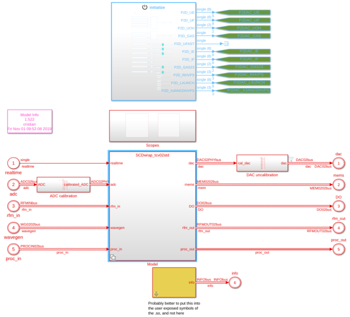
Galperti Cristian

SCDDS loader

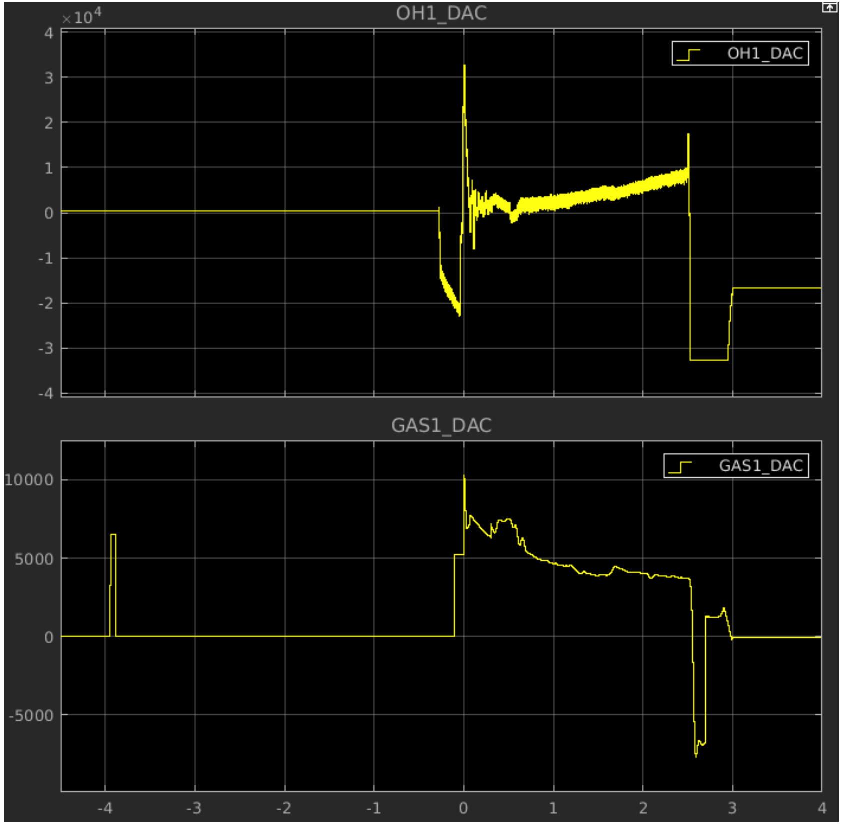
Shot parameters From MDS+

Shot ADC data From MDS+

SCDDS loader



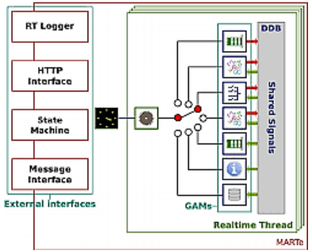
Shot simulation



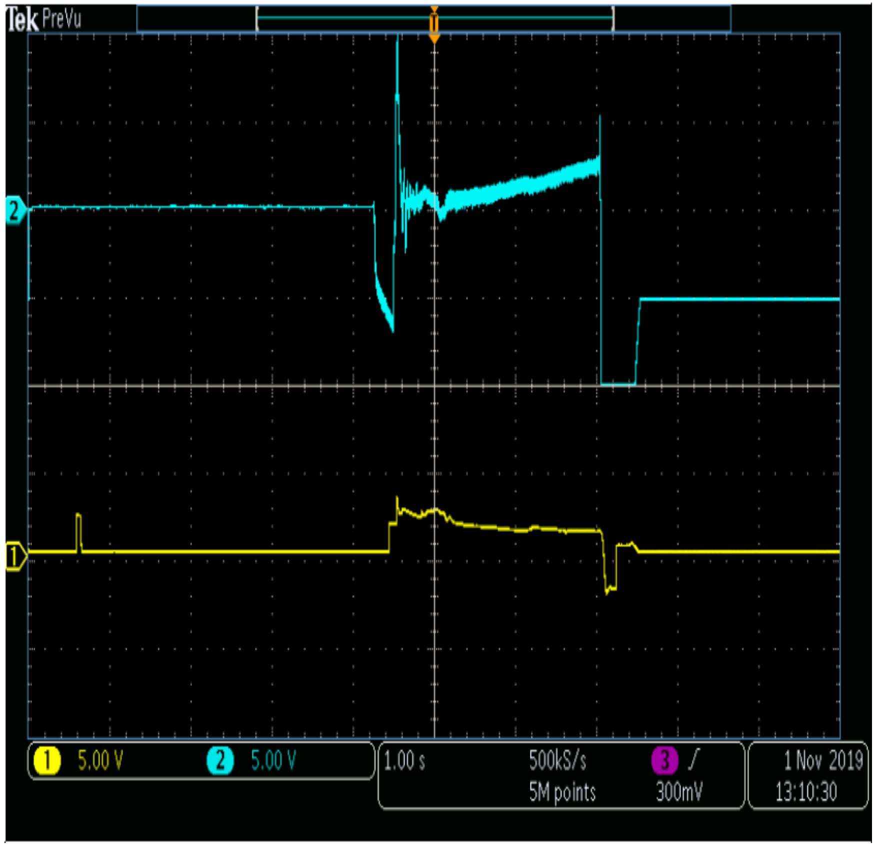
TCV real-time shot reprocessing with MARTe2 example

MARTe2
loader

Shot parameters
From MDS+



RT reprocess run



Shot ADC data
From MDS+

MARTe2
loader

TOUR
Available

Multi CPU control layout on TCV

CPU#2

- Main linear magnetic controller

CPU#3

- MEQ solver (RTLIUQE)
- Vertical growth rate estimator
- Disruption proximity estimator

CPU#4

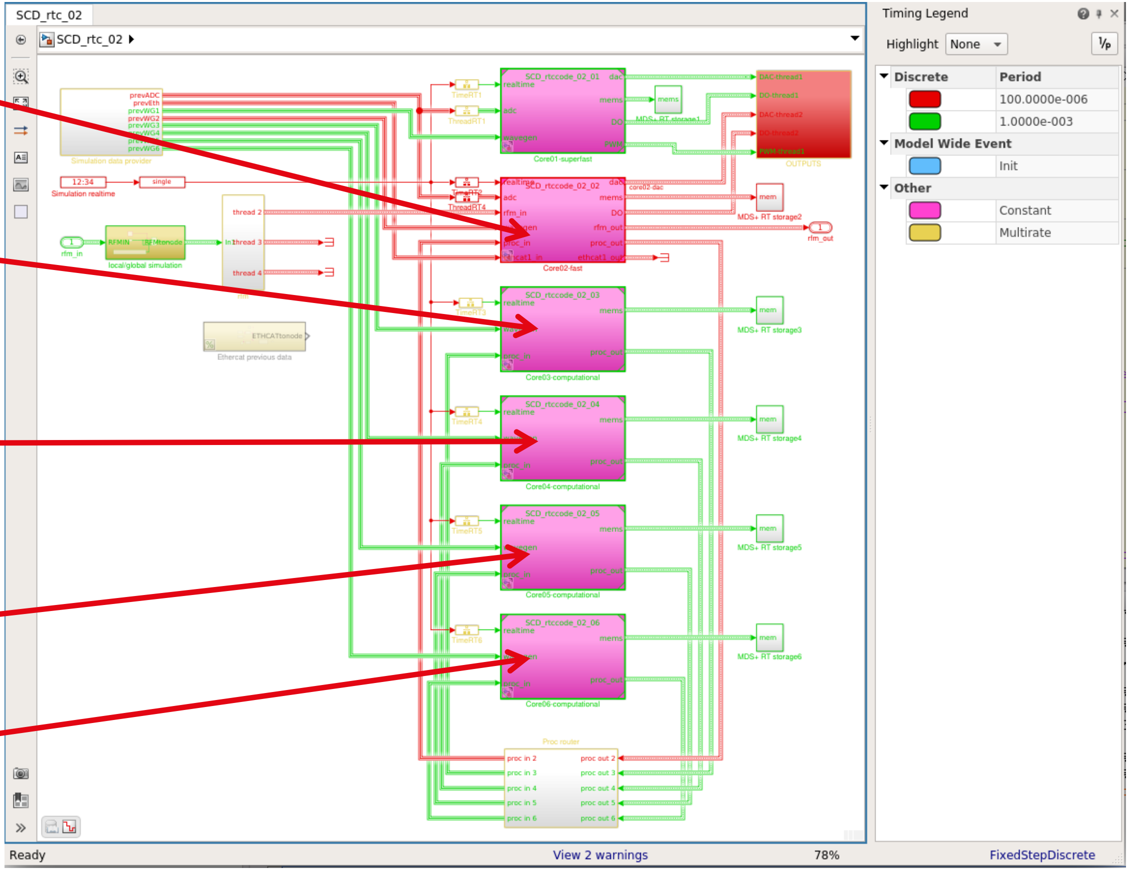
- Actuator manager and pulse scheduler (SAMONE)

CPU#5

- RAPTOR (TBC)

CPU#6

- RAPDENS (TBC)

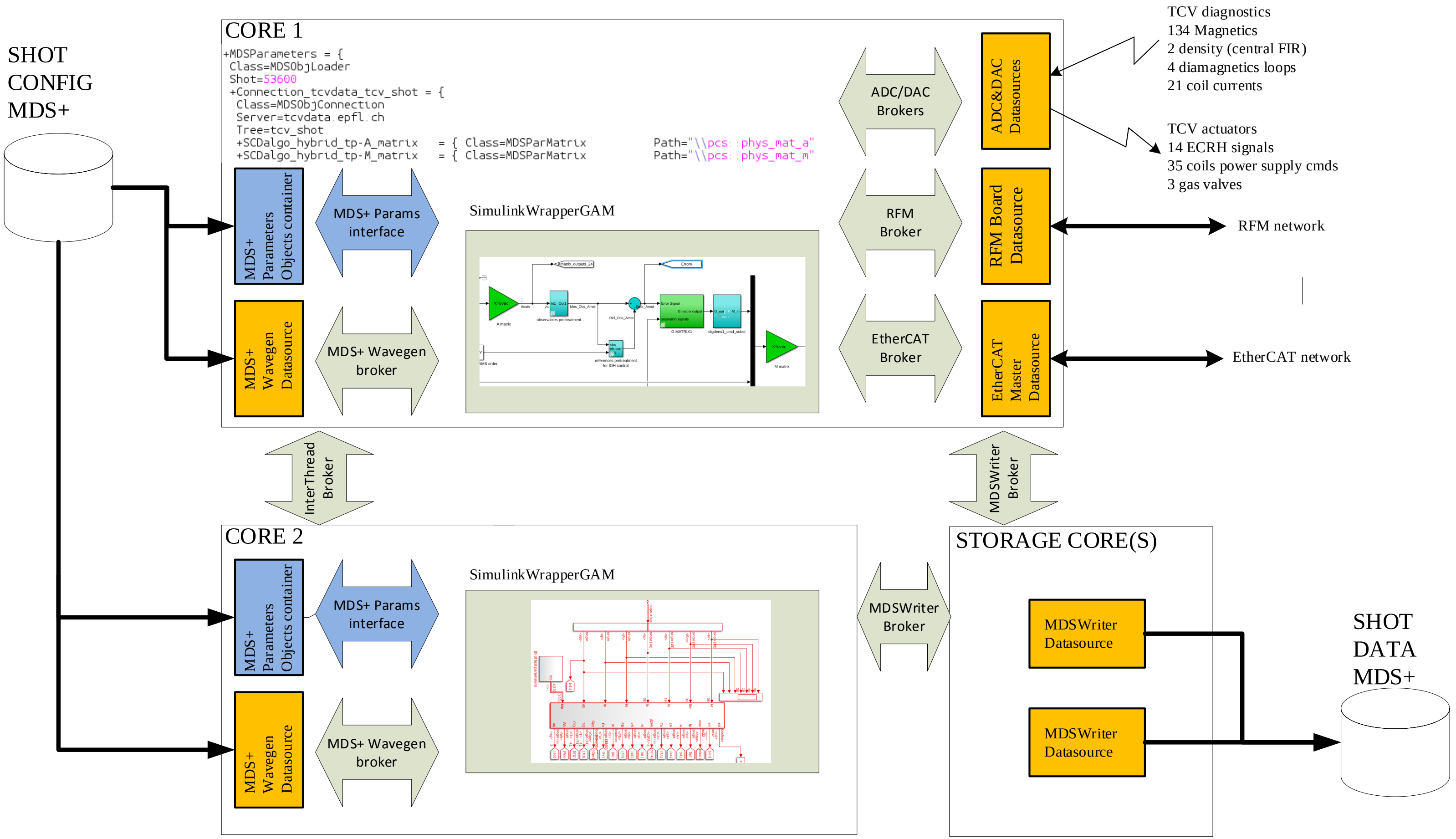


Galperti Cristian

Swiss Plasma Center

EPFL

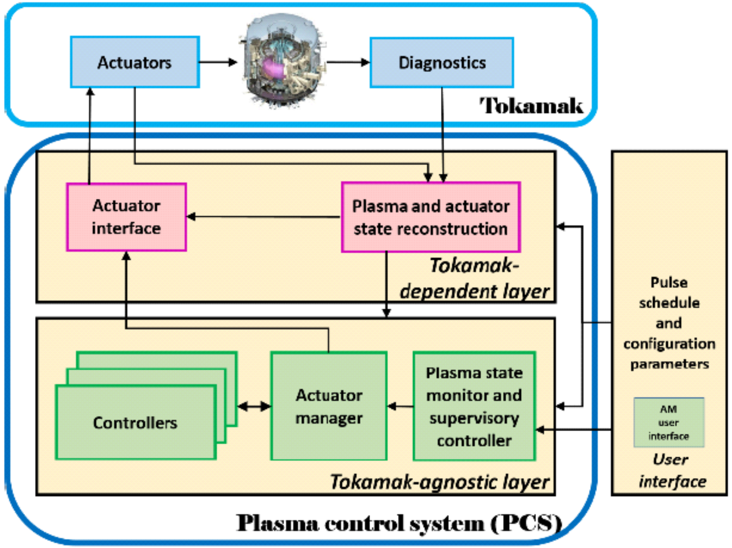
And its deployment with MARTe2

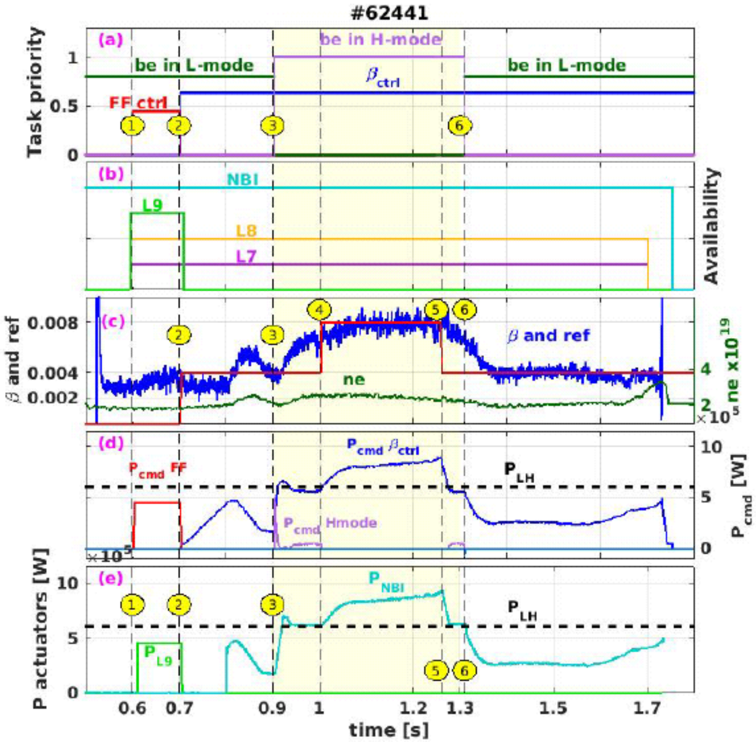


Swiss Plasma Center

Supervisory actuator manager and off normal event handling (SAMONE)

Galperti Cristian





- Complete configurable plasma supervisory, actuator manager and off normal events handler to deal with multiple tasks and few actuators

- Beta real-time control togther with confinement status (L/H) control via NBH 1

REF: [T. Vu et al. IEEE TNS 2021]
REF: [F. Felici et al. IAEA 2021]
REF: [C. Galperti et al IAEA TM on Plasma Control Systems 2021]

Conclusions, outlooks, take home points

- Tokamaks control software and hardware is continuously evolving, and this is good since control research on them does the same.
- Keeping control systems updated is a key point for a tokamak plant, as they can accommodate newer ideas (e.g. machine learning) or more complex codes (e.g. transport simulators, microwave and neutral beams raytracers).
- There will be more and more need for globally, plant wide interconnected control systems as more and more complex control tasks will be put in their hands (disruption avoidance is a primary example).
- Advanced, interconnected control system will be of primary importance for existing and next future long pulse tokamak, ITER at first.

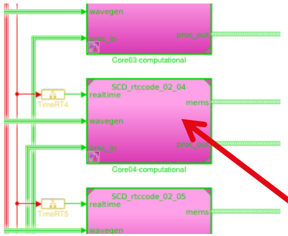
Thank you

cristian.galperti@epfl.ch

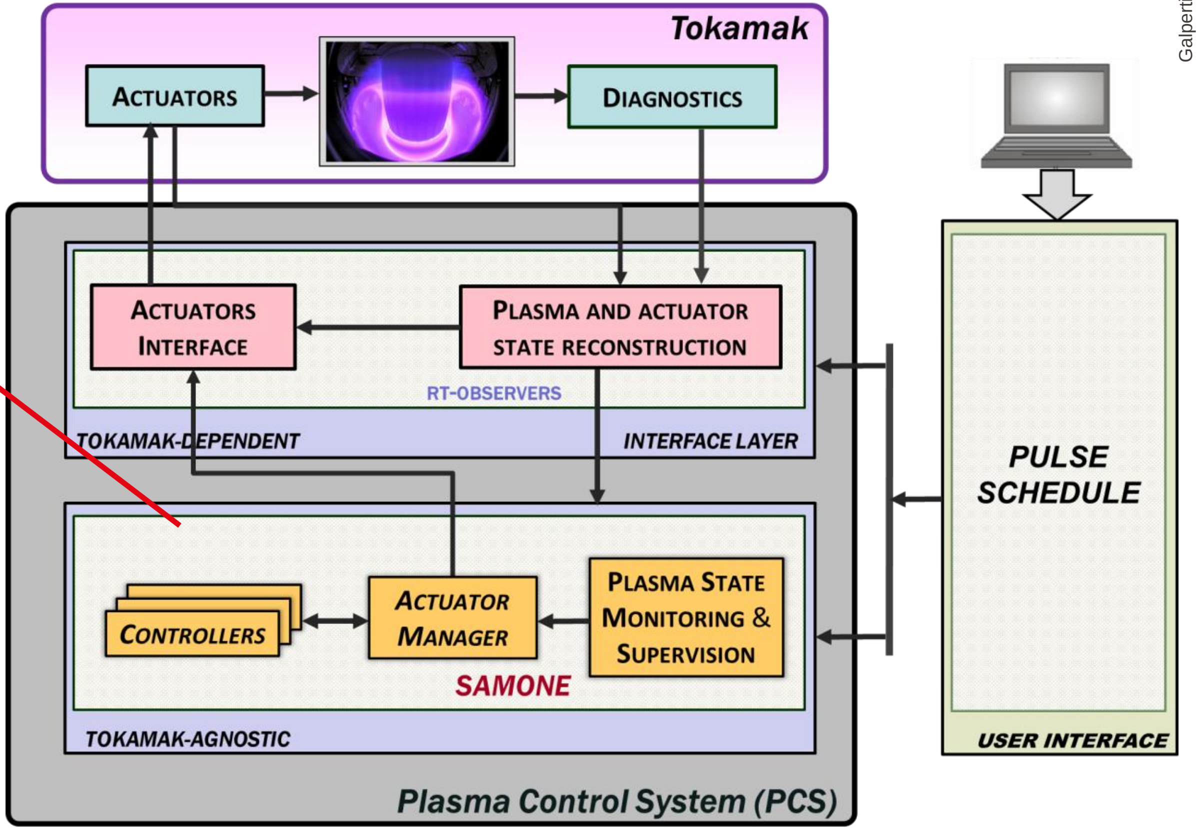
Swiss
Plasma
Center

BACKUP

CPU #4 SAMONE (supervisory actu



- Separation of tokamak **dependent** and **agnostic** layers
- **Generic** implementation
- **Flexible** framework allowing easy **maintenance** and **upgrade**
- Concepts of **integration** and **portability**

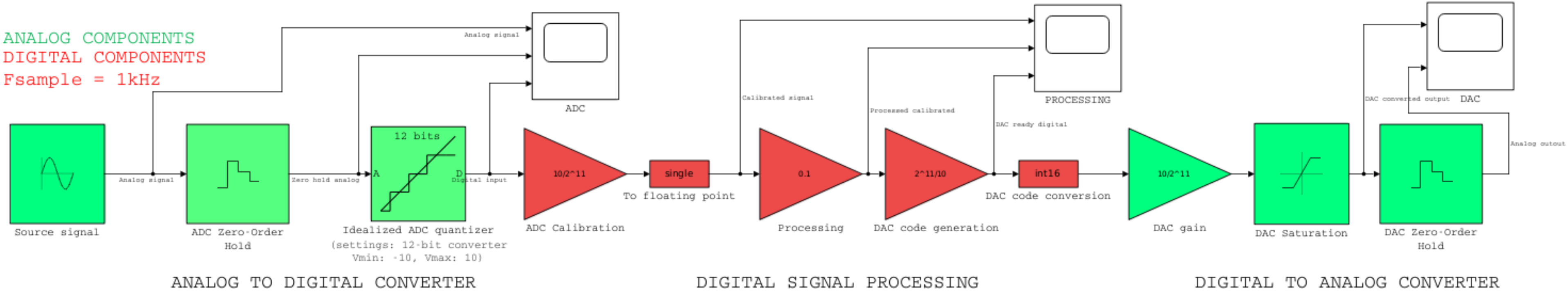


REF: [T. Vu et al. IEEE TNS 2021]
REF: [F. Felici et al. IAEA 2021]
REF: [C. Galperti et al IAEA TM on Plasma Control Systems 2021]

Common problems

- Quantizers dynamic range and speed -> analog pretreatment
- Output under-overflow

A complete control chain

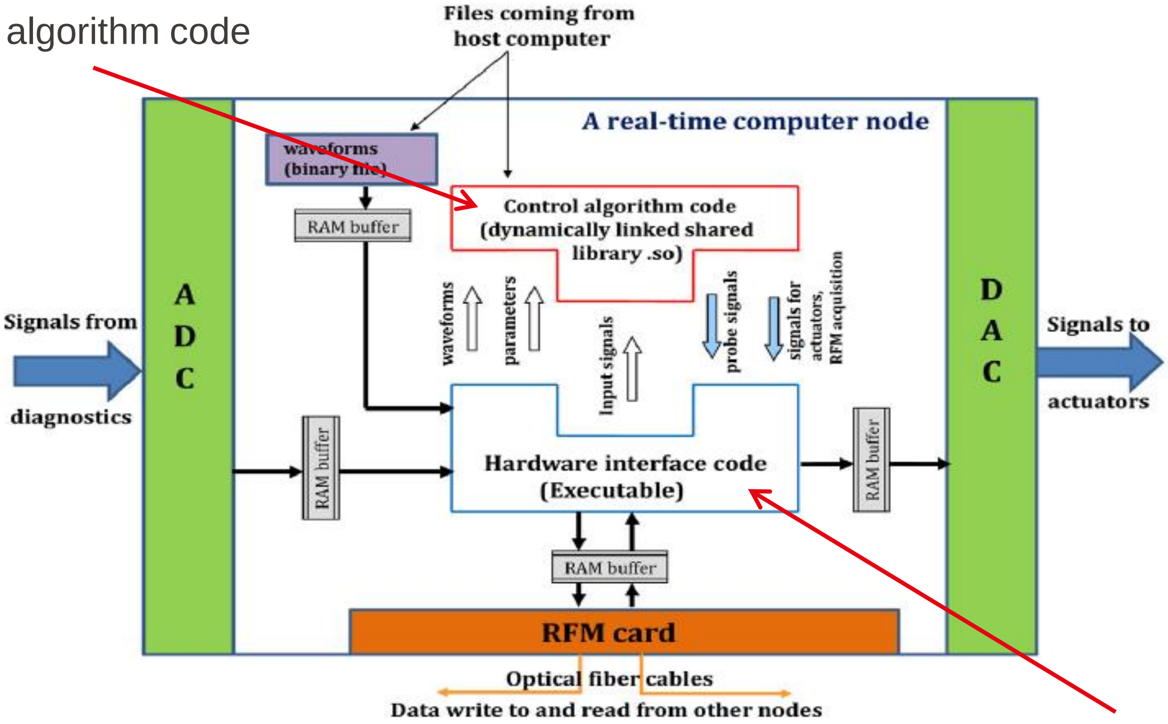


Swiss Plasma Center

General control software architecture TODO: UPDATE

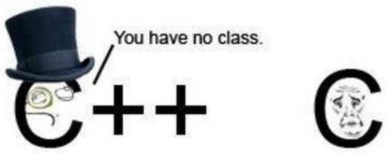
Galperti Cristian

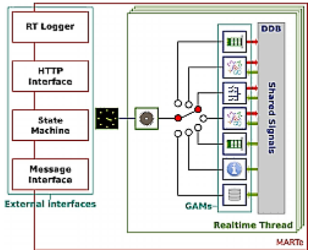
Control algorithm code



Hardware interface code (HIC from now on)

Swiss Plasma Center





HIC control code at the turning point, THIS CAN BE REMOVED IN FAVOR OF 1 OR 2 MARTE2 SLIDES

In house developed C++ project

Pros:
- Full control on every part of the code.
- Fine tuning of RT performances (we deal with 100us cycle time machines).
- Code complexity can be kept low.
- Code functionalities can be tailored specifically for TCV.

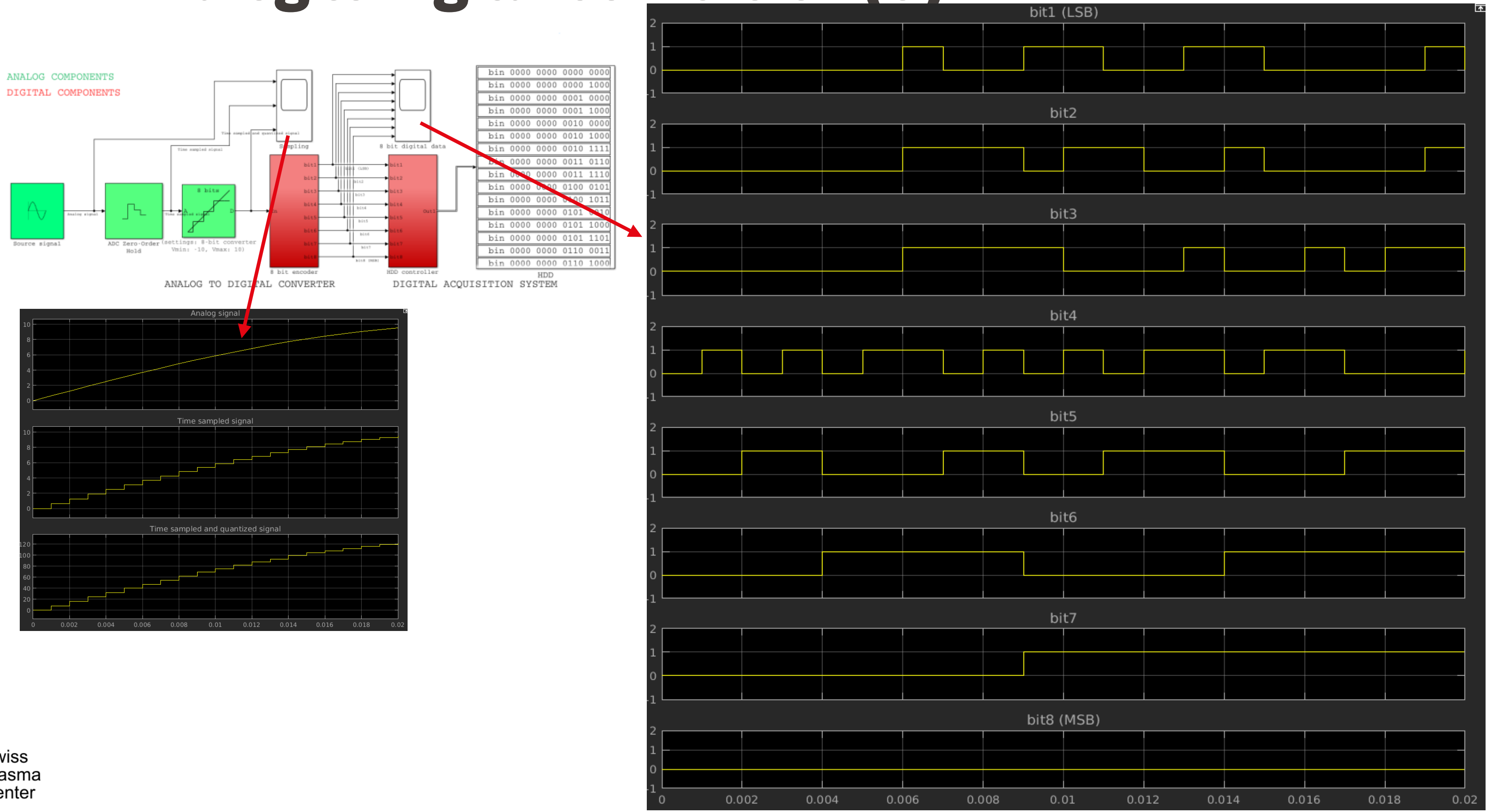
Cons:
- Maintainable code architecture, skills and time needed.
- Used only here, no support form any community.
- **MDS+ integration must be developed on purpose**
- **Data driven application required for avoiding just in time code compilation**

F4E / MARTe2 framework

Pros:
- Comprehensive multi CPU control systems framework with a number of already developed functionalities: web interface, UDP logger, state machine, signal collection, **MDS+ read/write interface**
- Support from a community, **standardization of control nodes enforced.**
- **Data driven application natively supported**

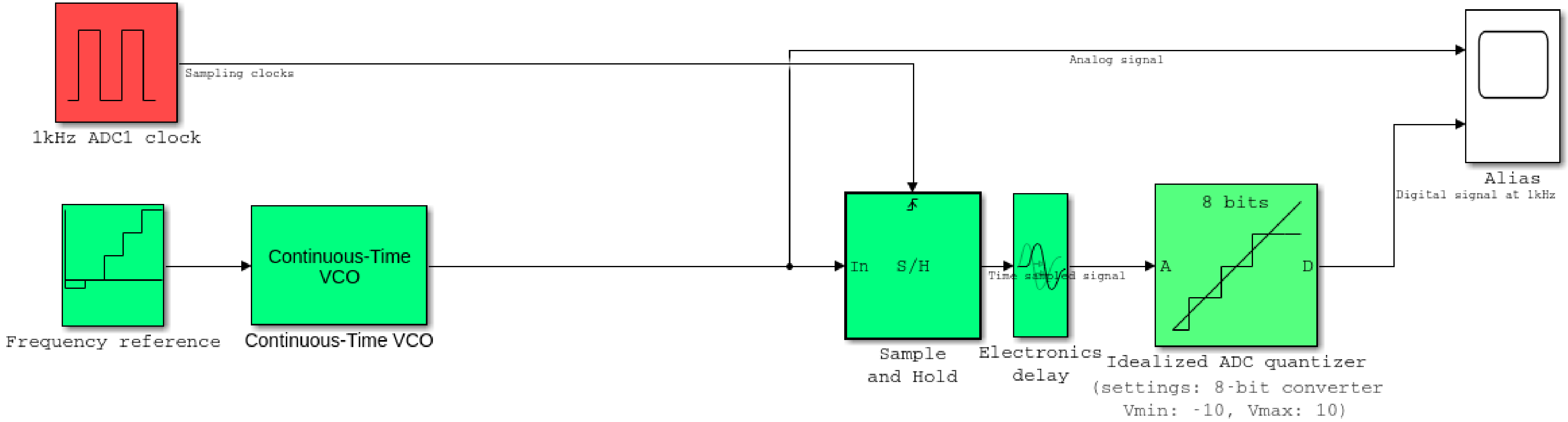
Cons:
- ~~TCV specific interfaces not well supported (TDI, MDS+ and Simulink)~~
- Quite big multiclass C++ project to learn.
- ~~RT performances on sub-ms multi CPU systems to be assessed.~~

Analog to digital conversion (3)



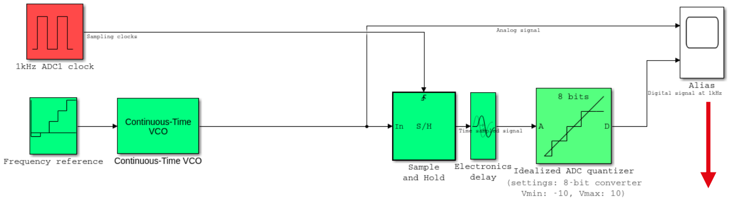
Swiss Plasma Center

Maximum speed, the alias effect (1)

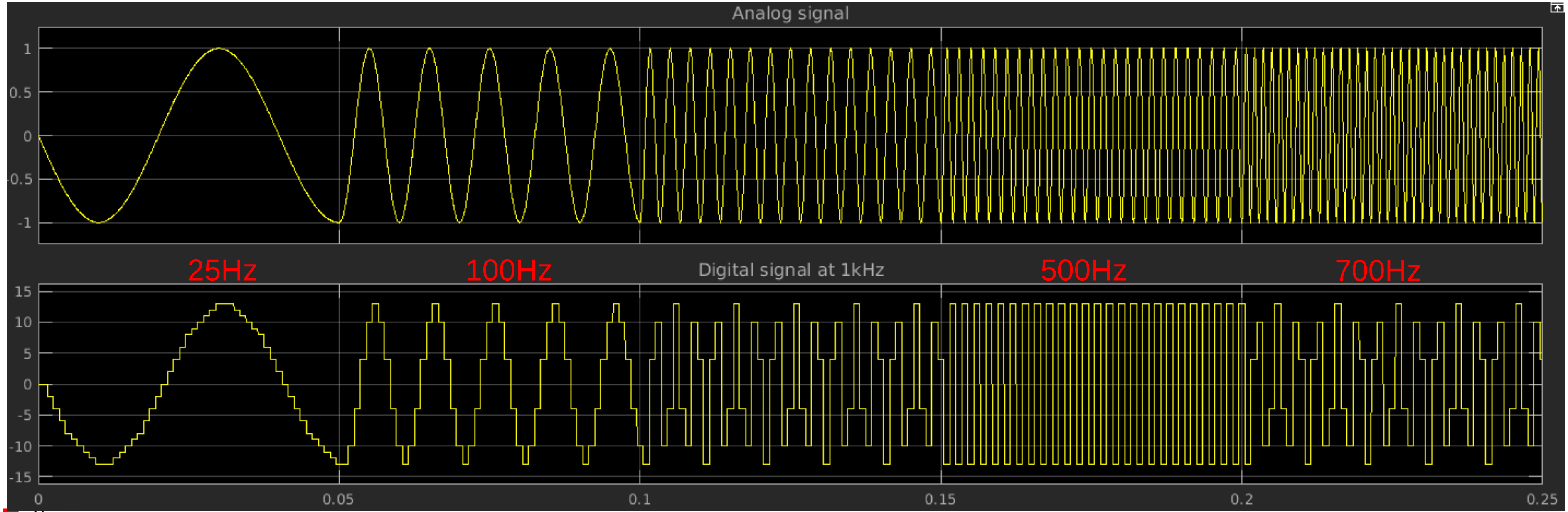


Swiss Plasma Center

Maximum speed, the alias effect (2)

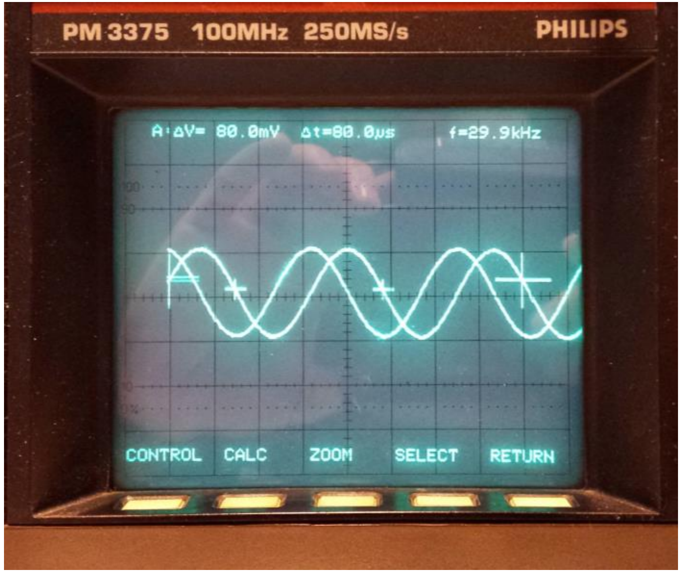


300Hz



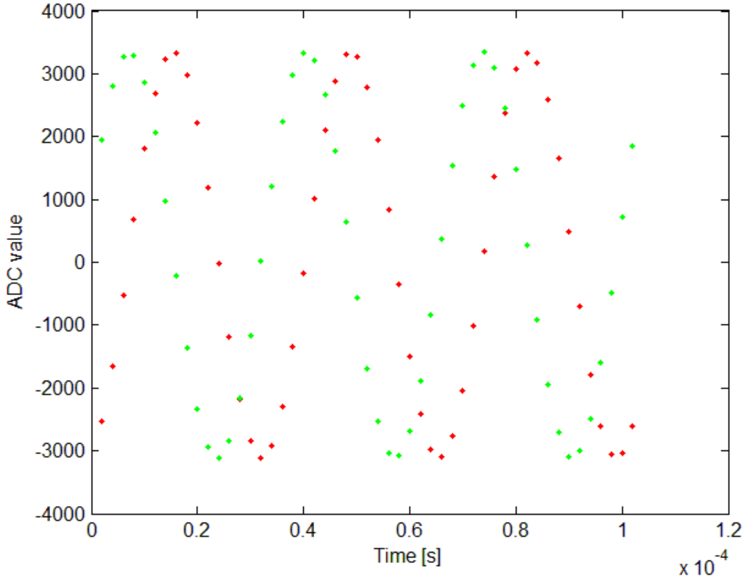
A real example (1)

Galperti Cristian



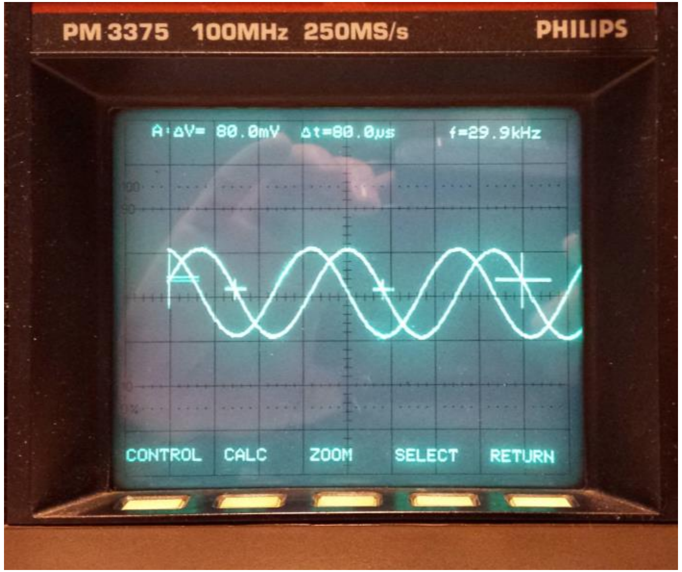
30 kHz analog sinewaves

500ksps
ADC



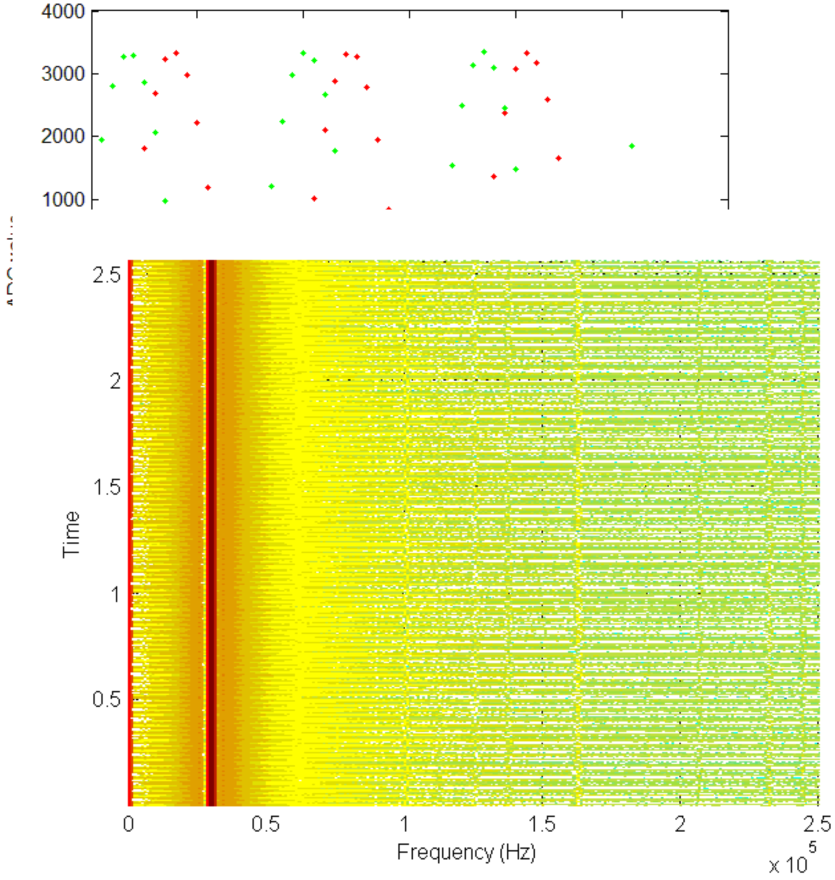
EPFL

A real example (2)



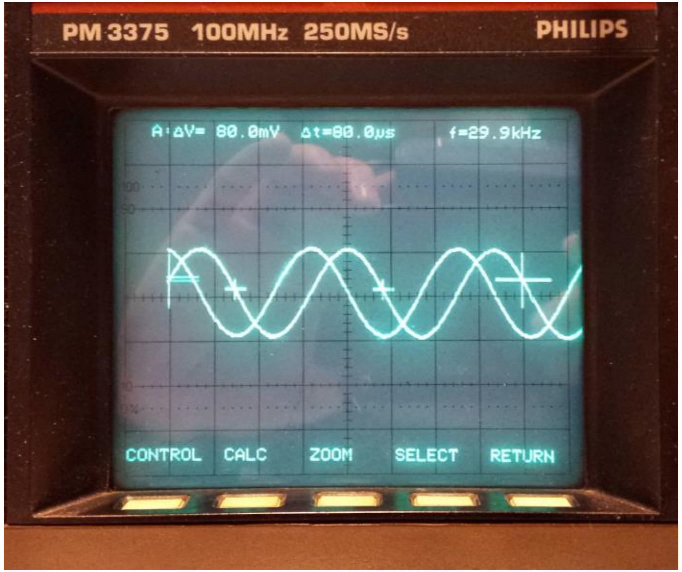
30 kHz analog sinewaves

500ksps
ADC



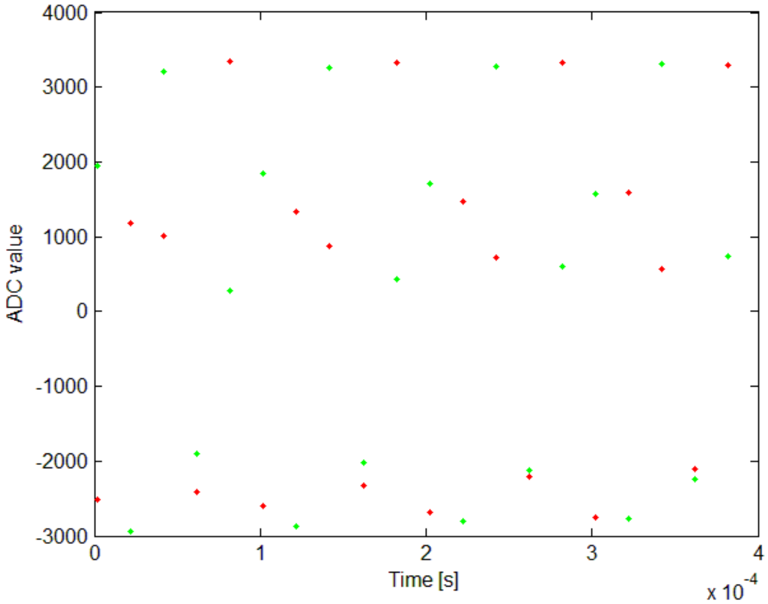
Galperti Cristian

A real example (3)



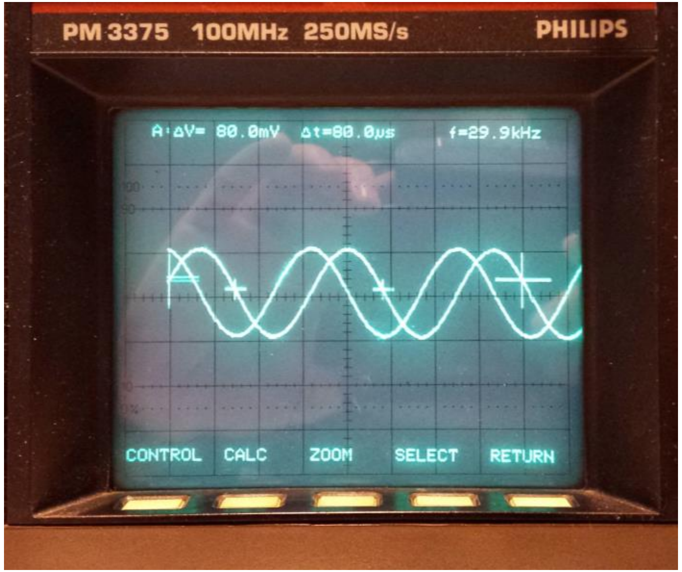
30 kHz analog sinewaves

50ksps
ADC



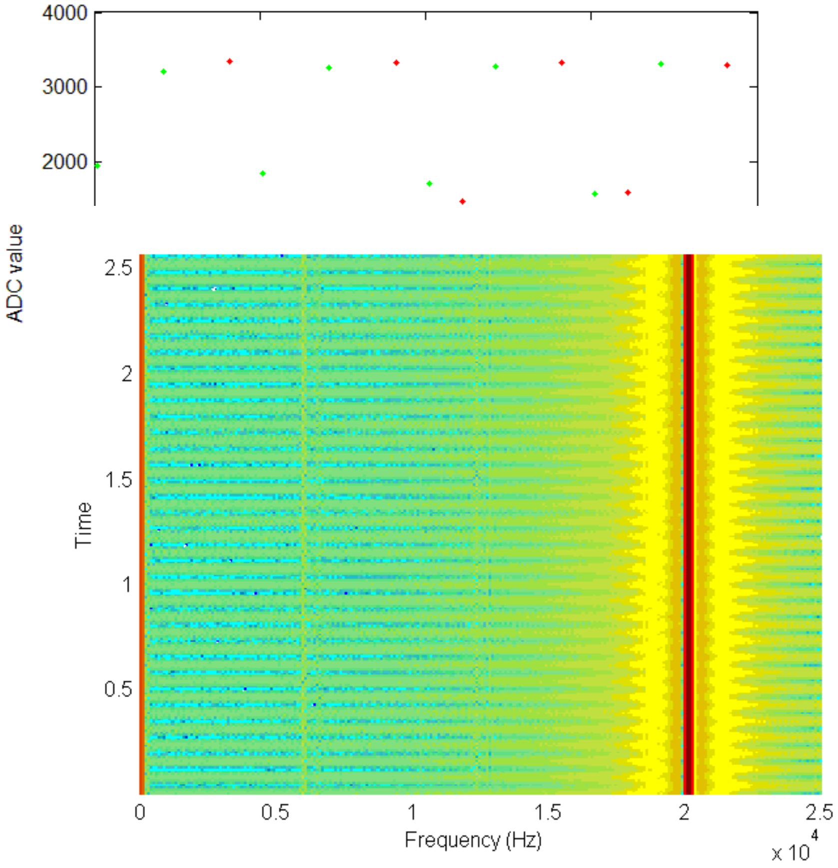
Galperti Cristian

A real example (4)



30 kHz analog sinewaves

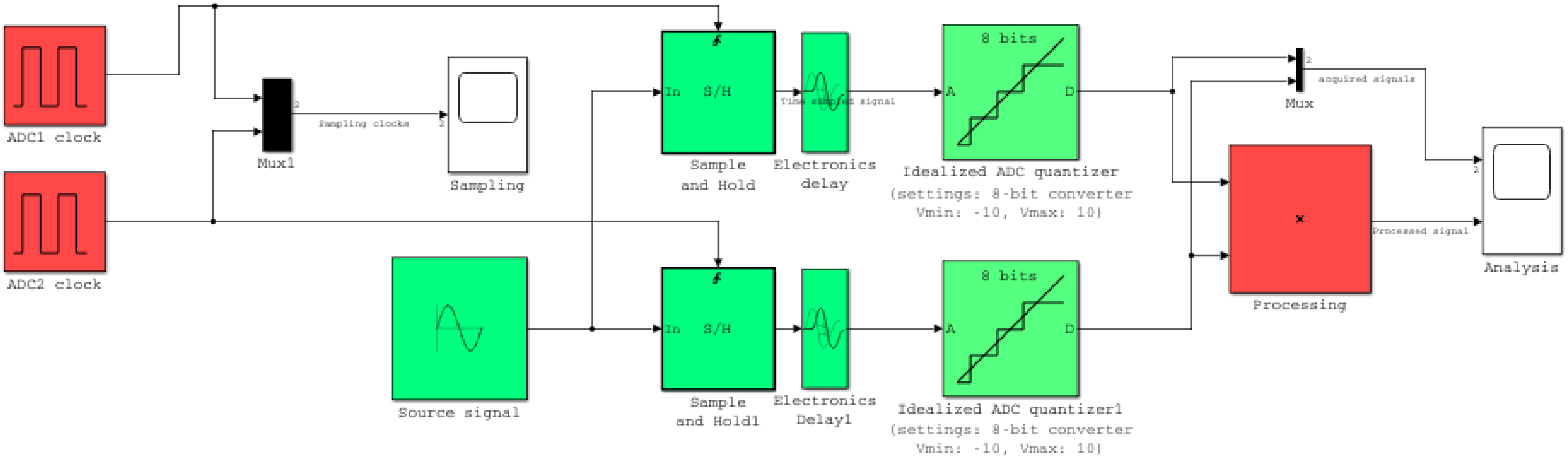
50ksps ADC



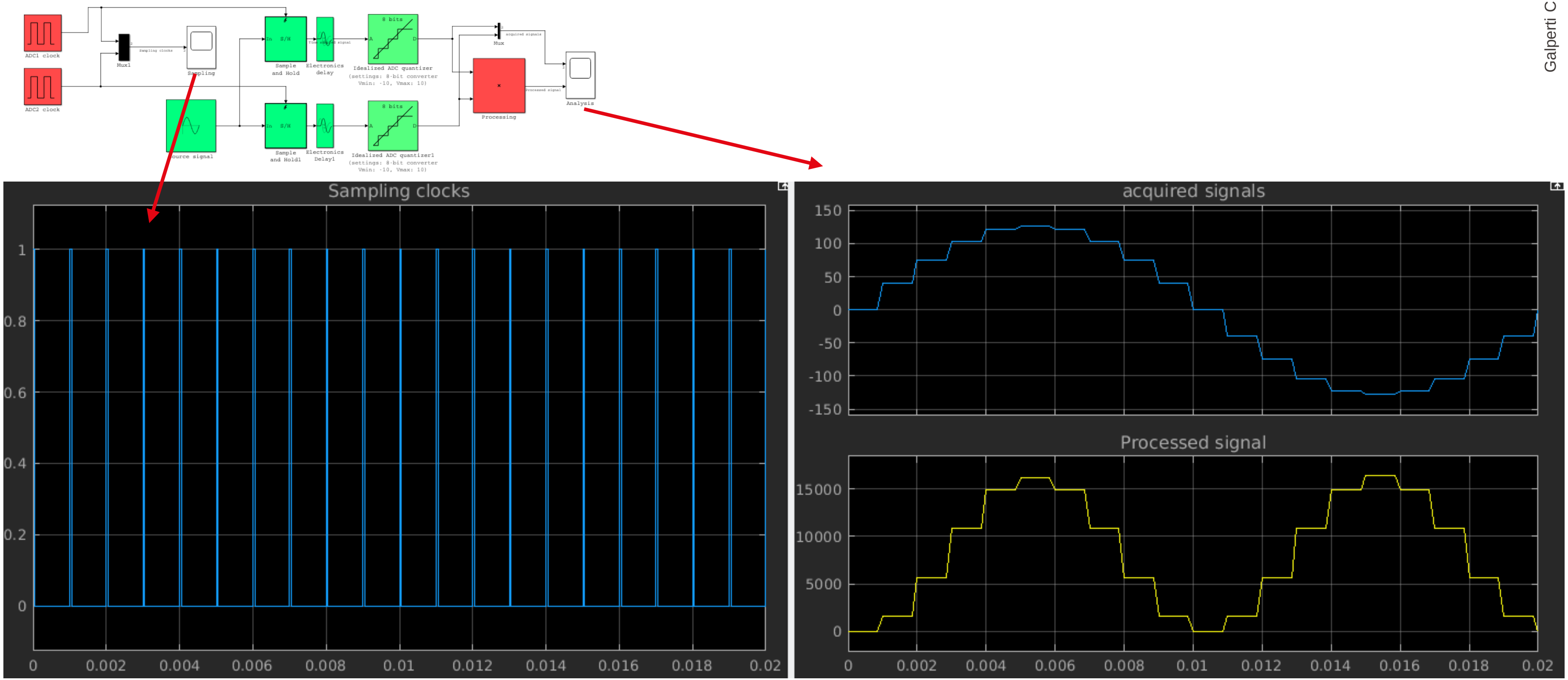
Swiss Plasma Center

Galperti Cristian

Multiple channels, simultaneous vs interlaced sampling

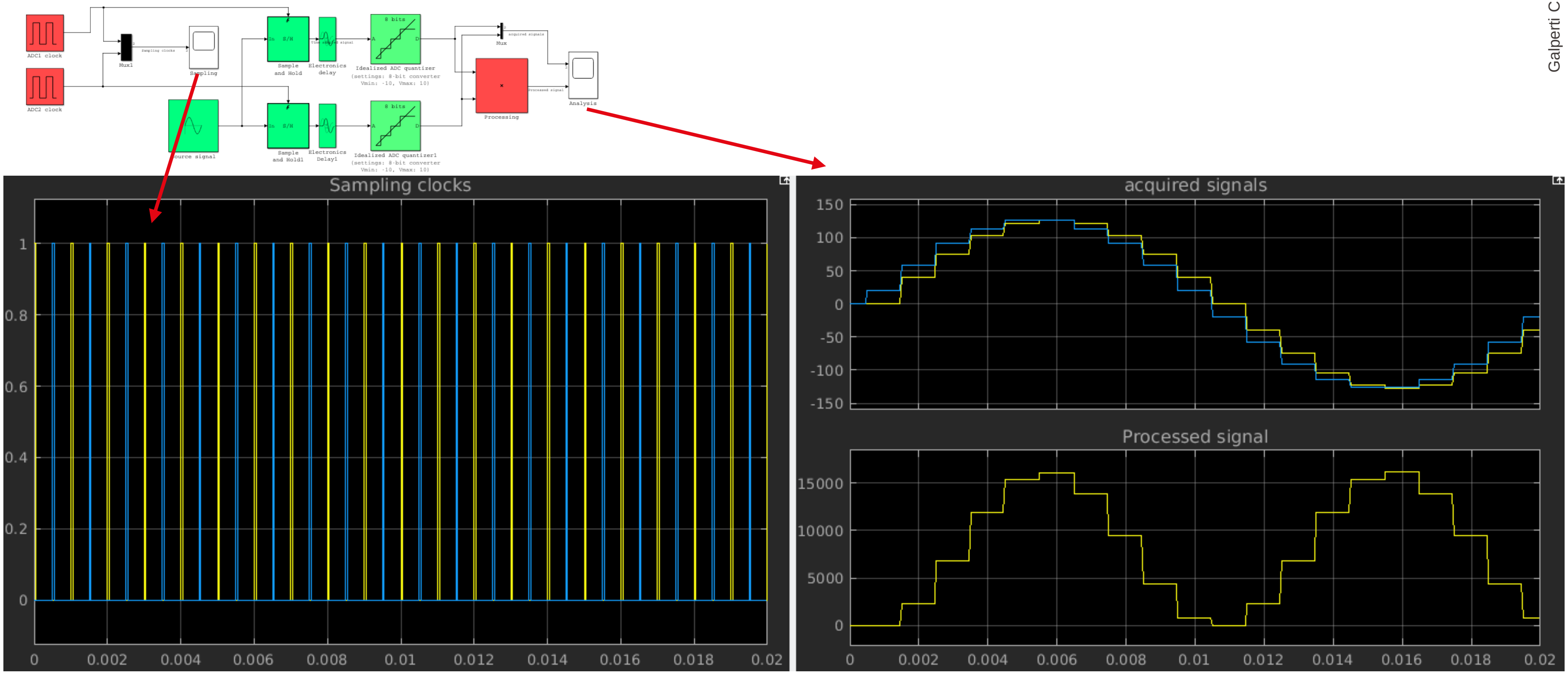


Multiple channels, simultaneous sampling



EPFL

Multiple channels, interlaced sampling

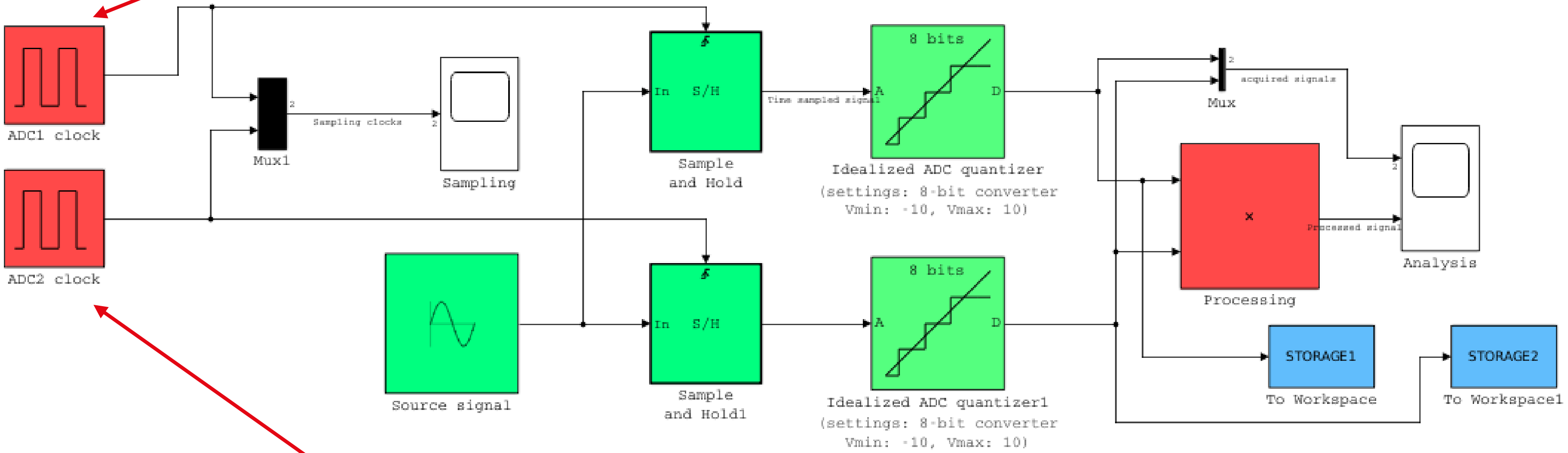


Galperti Cristian

Swiss
Plasma
Center

Not isochronous timebase (1)

1ms timebase

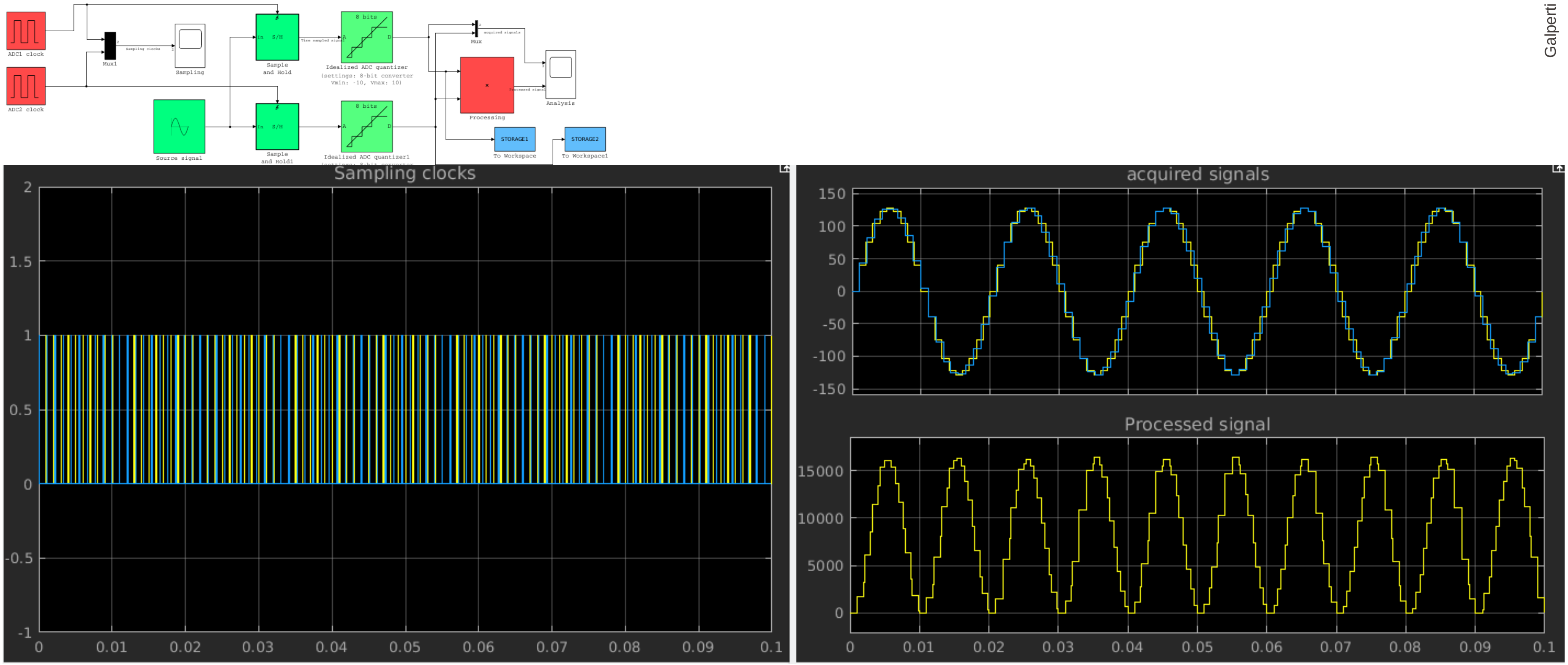


1.1 ms timebase

Swiss Plasma Center

Not isochronous timebase (2), RT processing

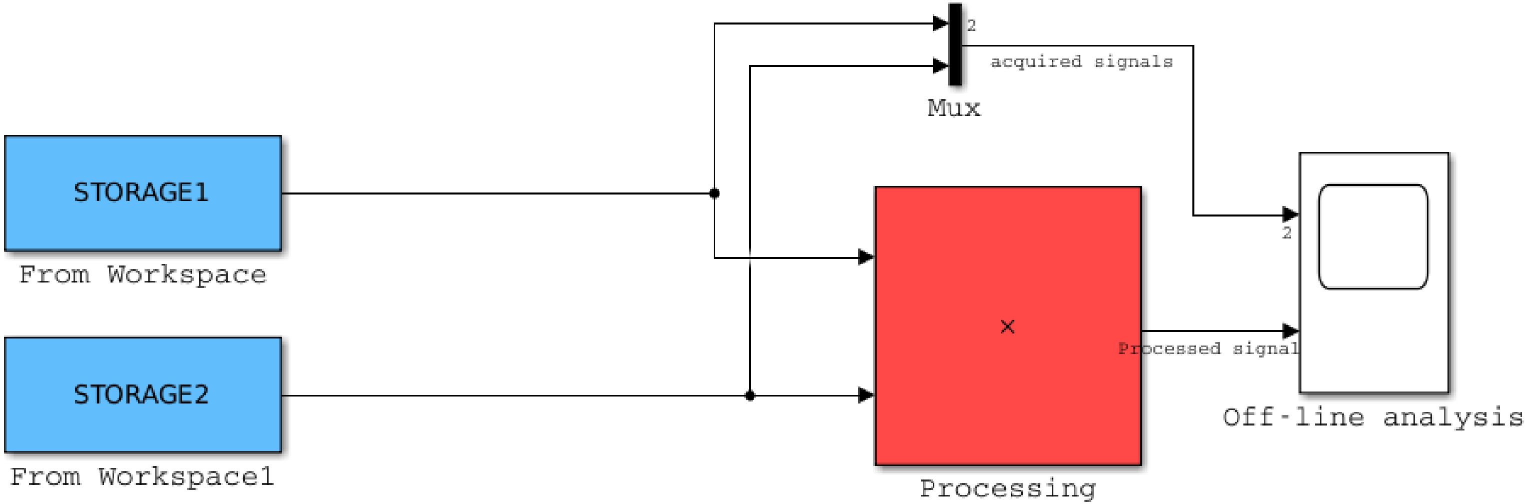
Galperti Cristian



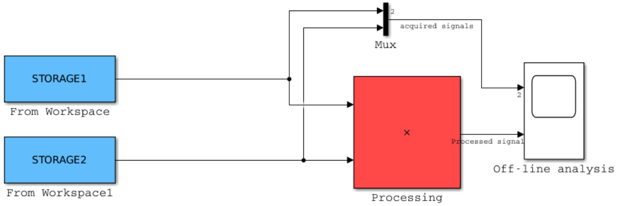
Swiss Plasma Center

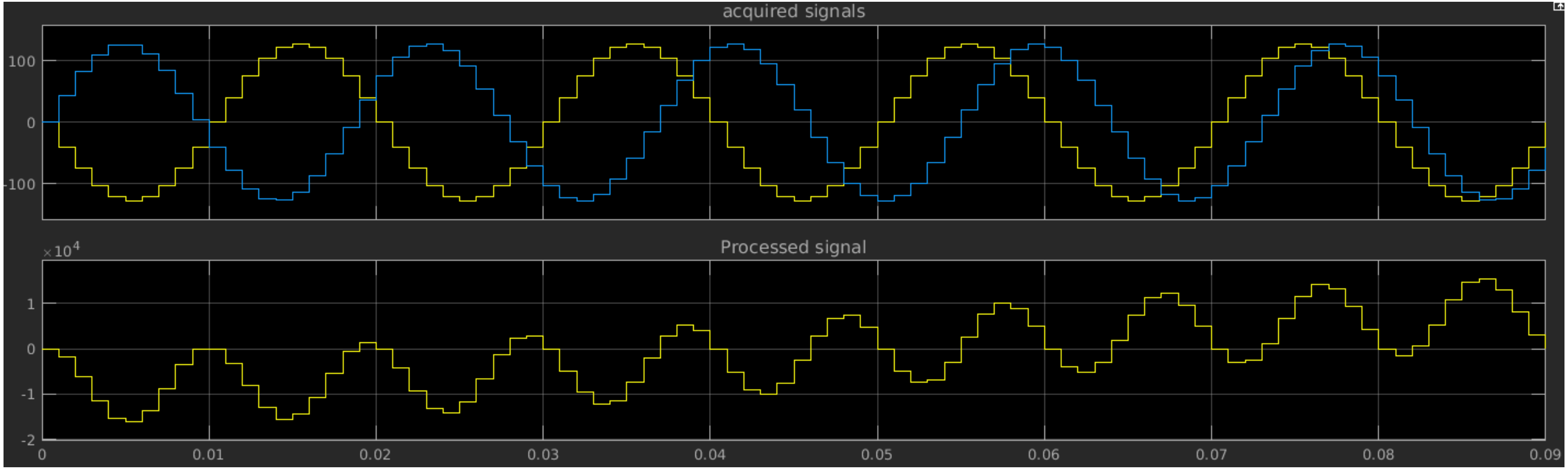
Not isochronous timebase (3), off-line analysis

Galperti Cristian

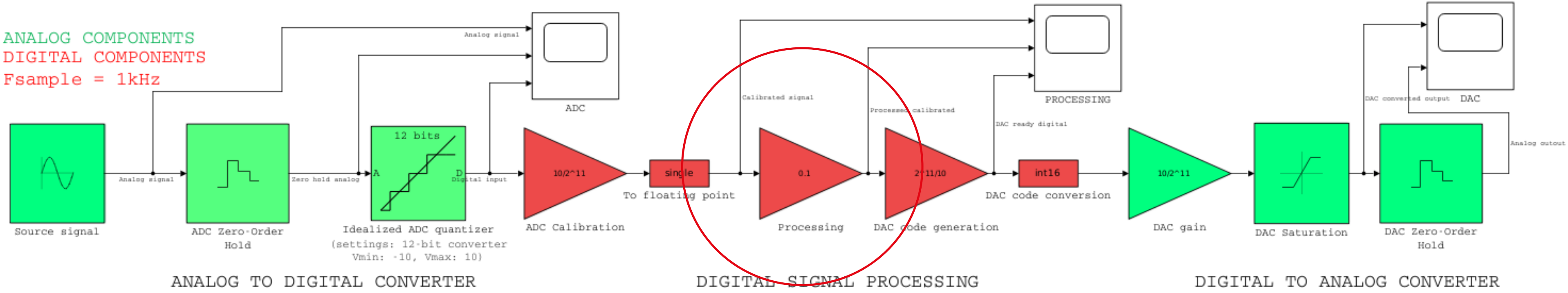


Not isochronous timebase (4), off-line analysis





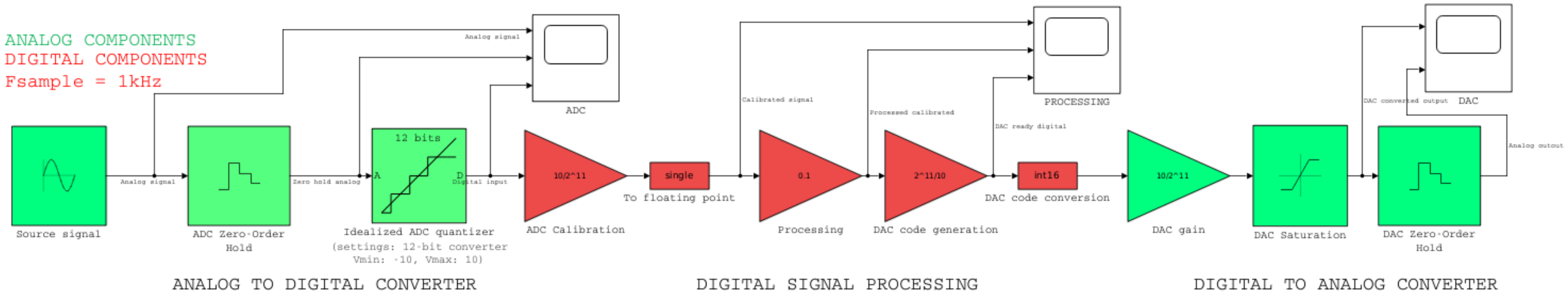
A complete control chain

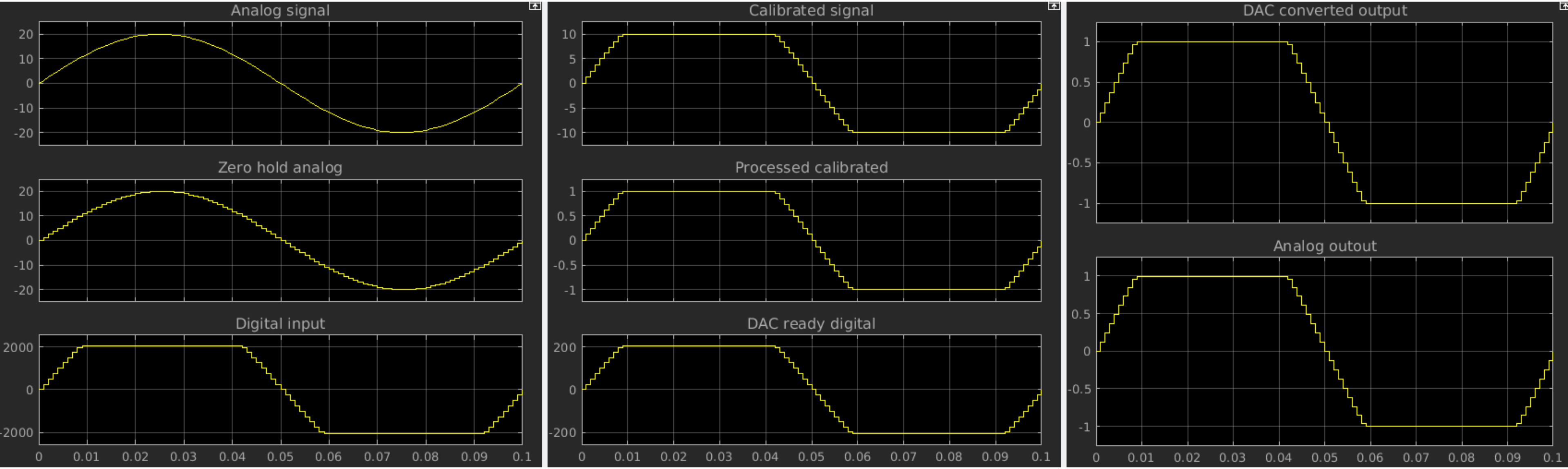


A very simple processing chain to have 0.1x input to output signal transfer.

Input saturation

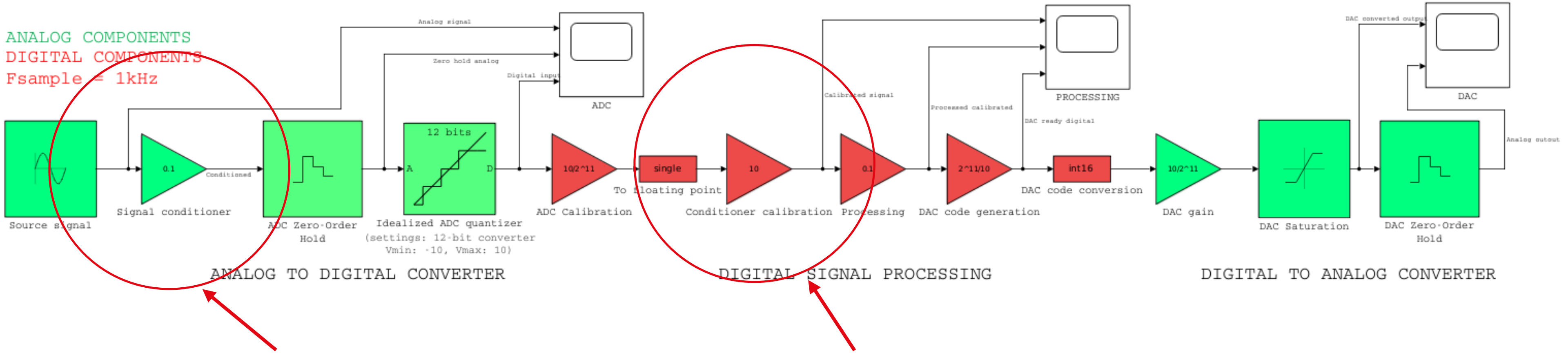
Galperti Cristian





Input signal conditioning (1)

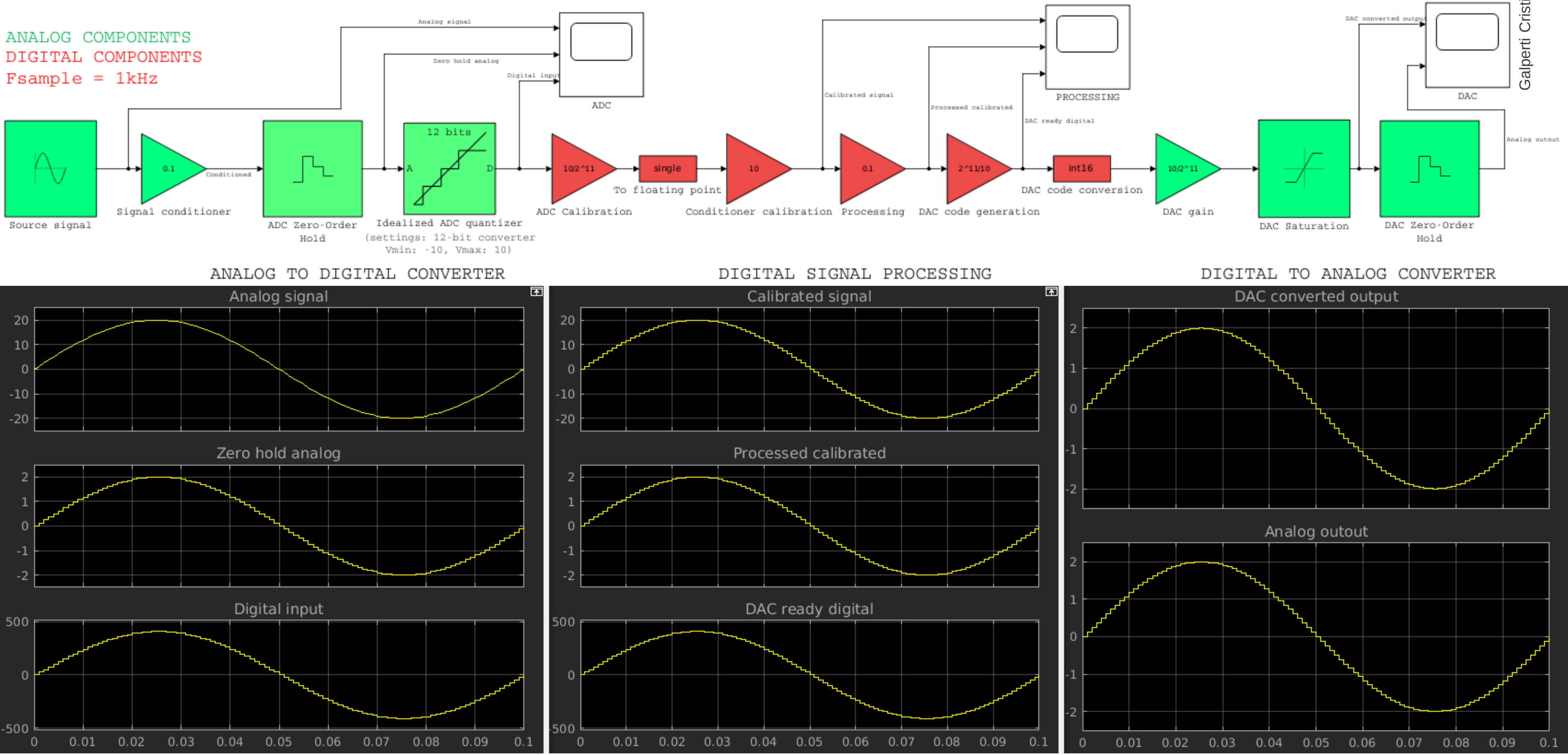
Galperti Cristian



The signal is pre-treated in analog domain before The ADC

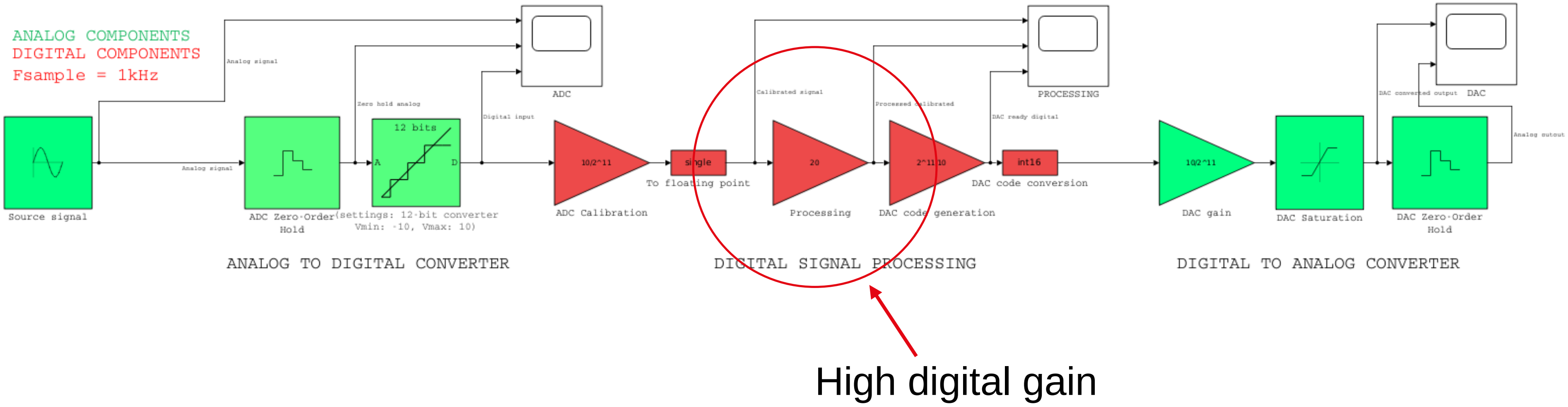
In digital we compensate for the analog pre-treatment

Input signal conditioning (2)



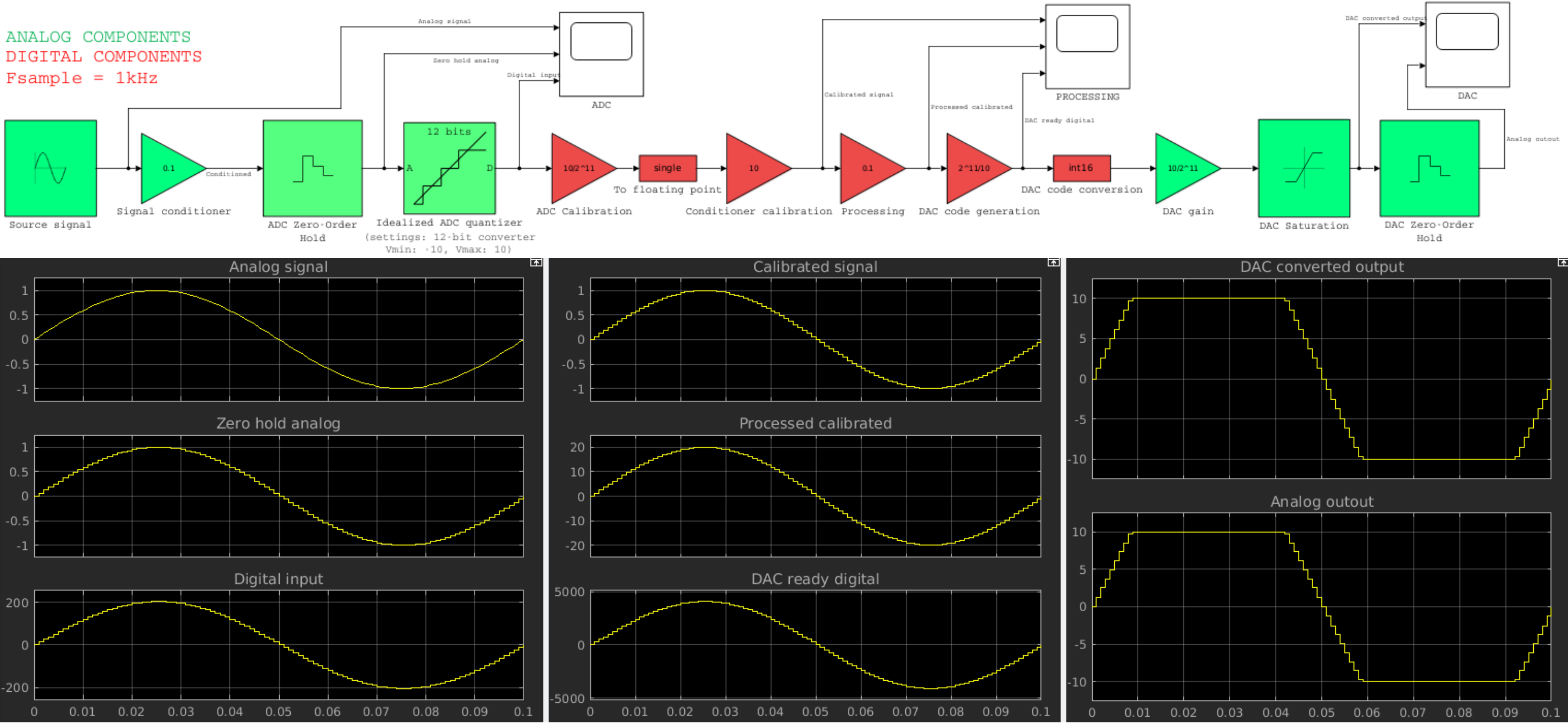
Galperti Cristian

Output saturation (1)



Swiss Plasma Center

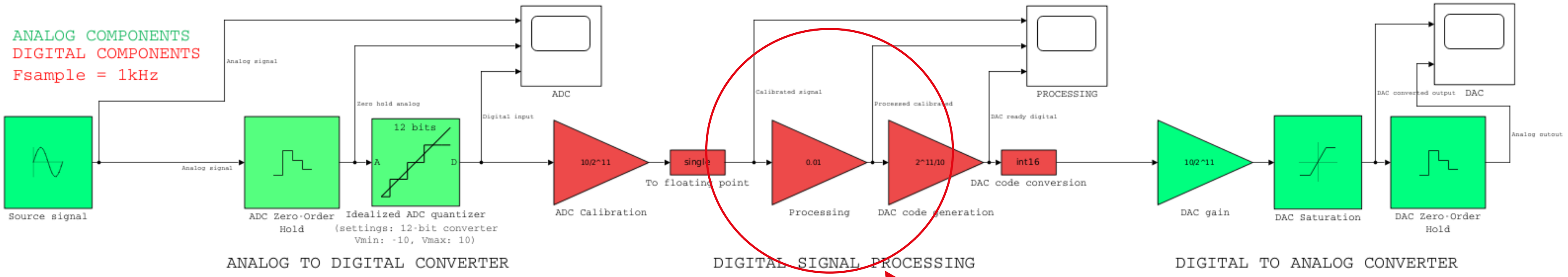
Output saturation (2)



EPFL

Output quantization (1)

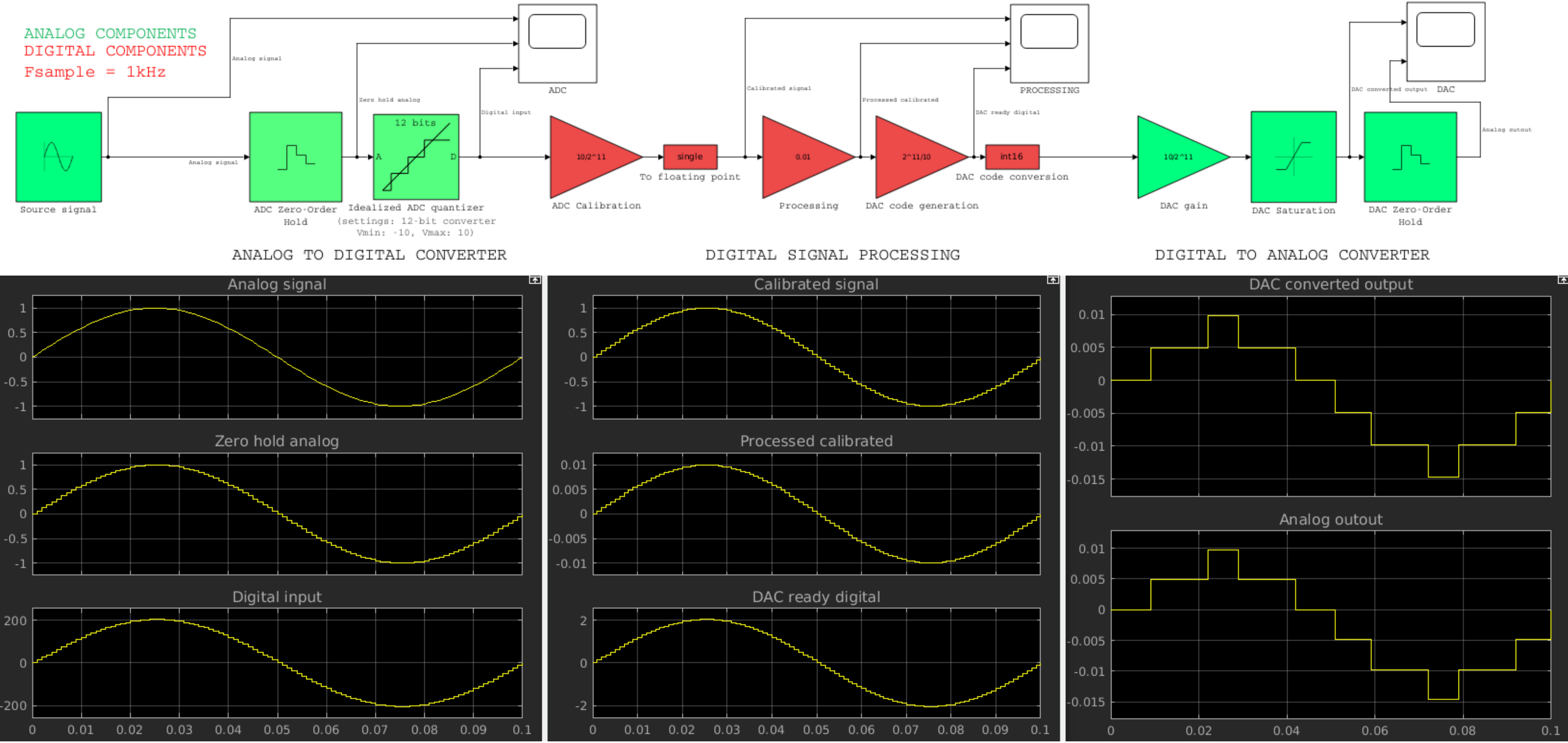
Galperti Cristian



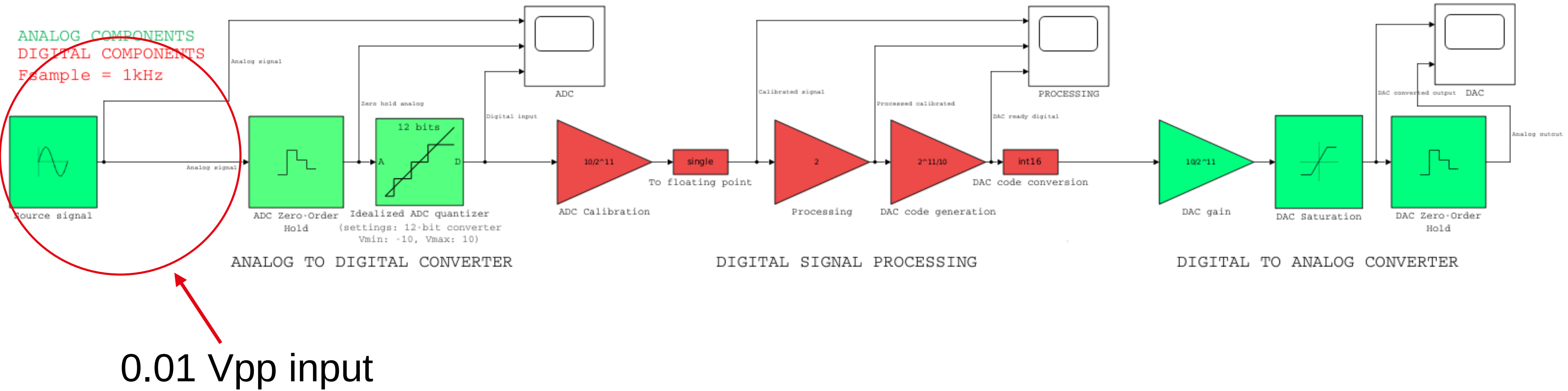
Low digital gain

Swiss Plasma Center

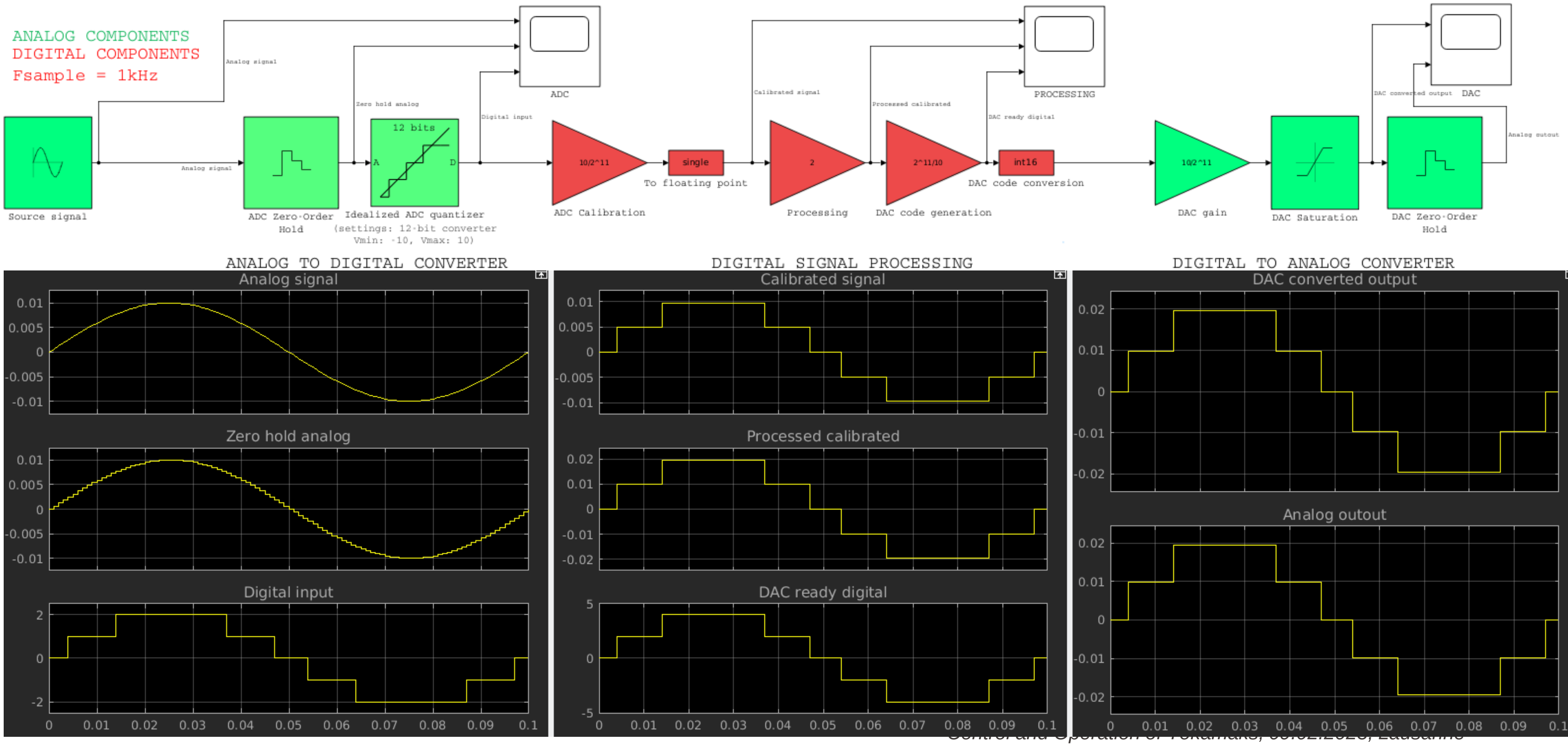
Output quantization (2)



Too low input signal (1)

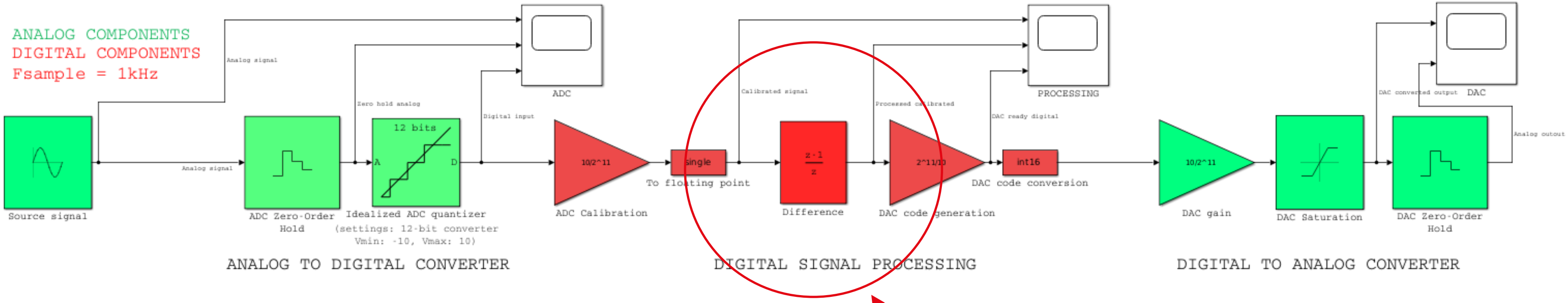


Too low input signal (2)



Quantization noise (1)

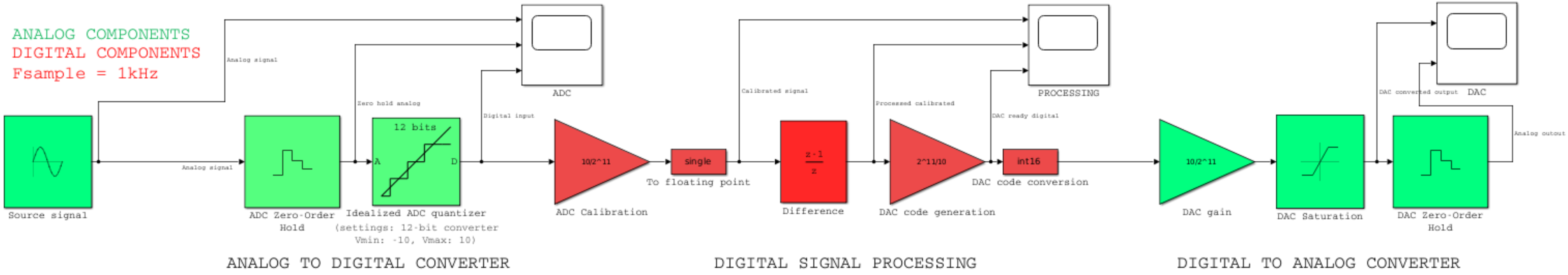
Galperti Cristian

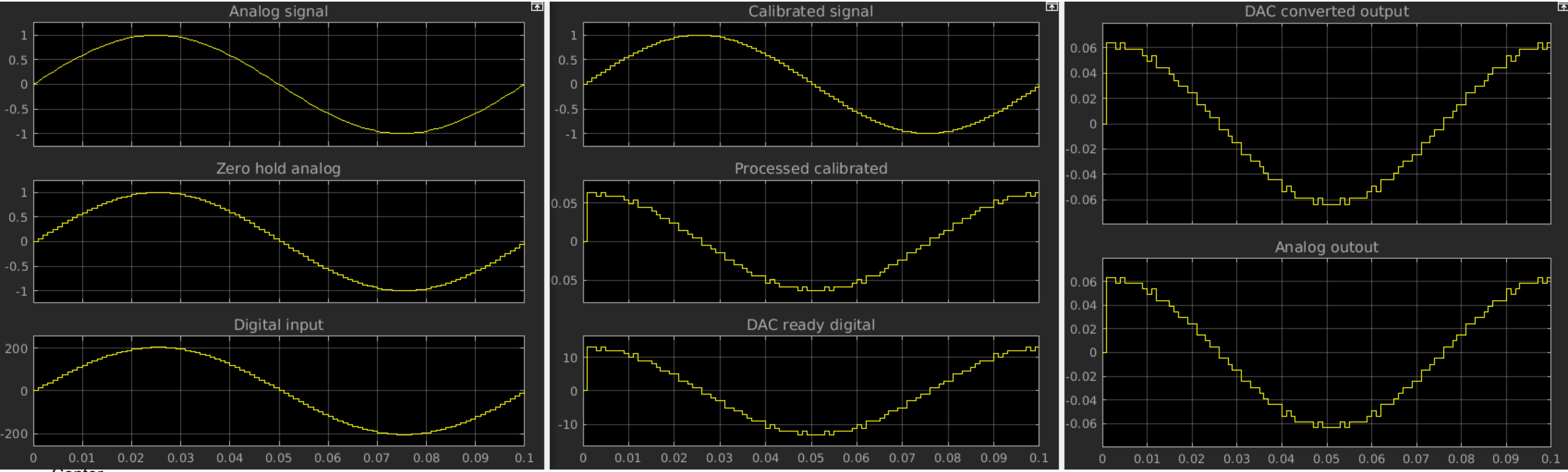


Digital differentiator (noise amplifier)

Quantization noise (2)

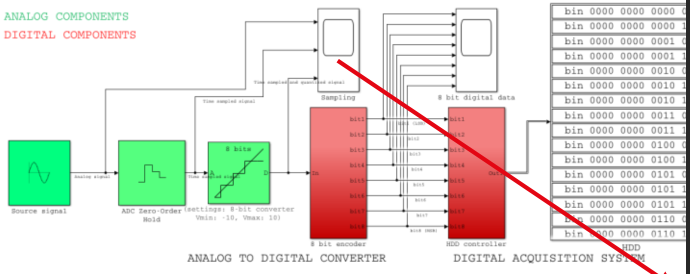
Galperti Cristian

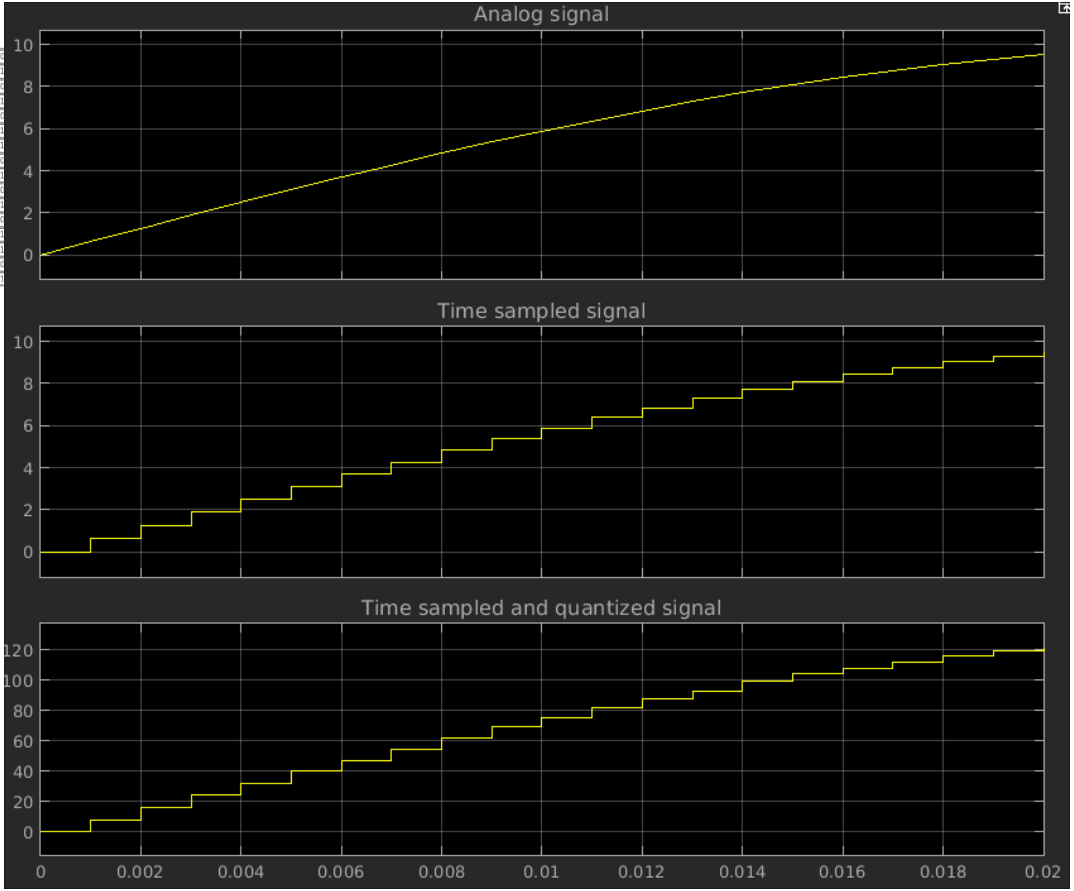




Center

Analog to digital conversion (2)

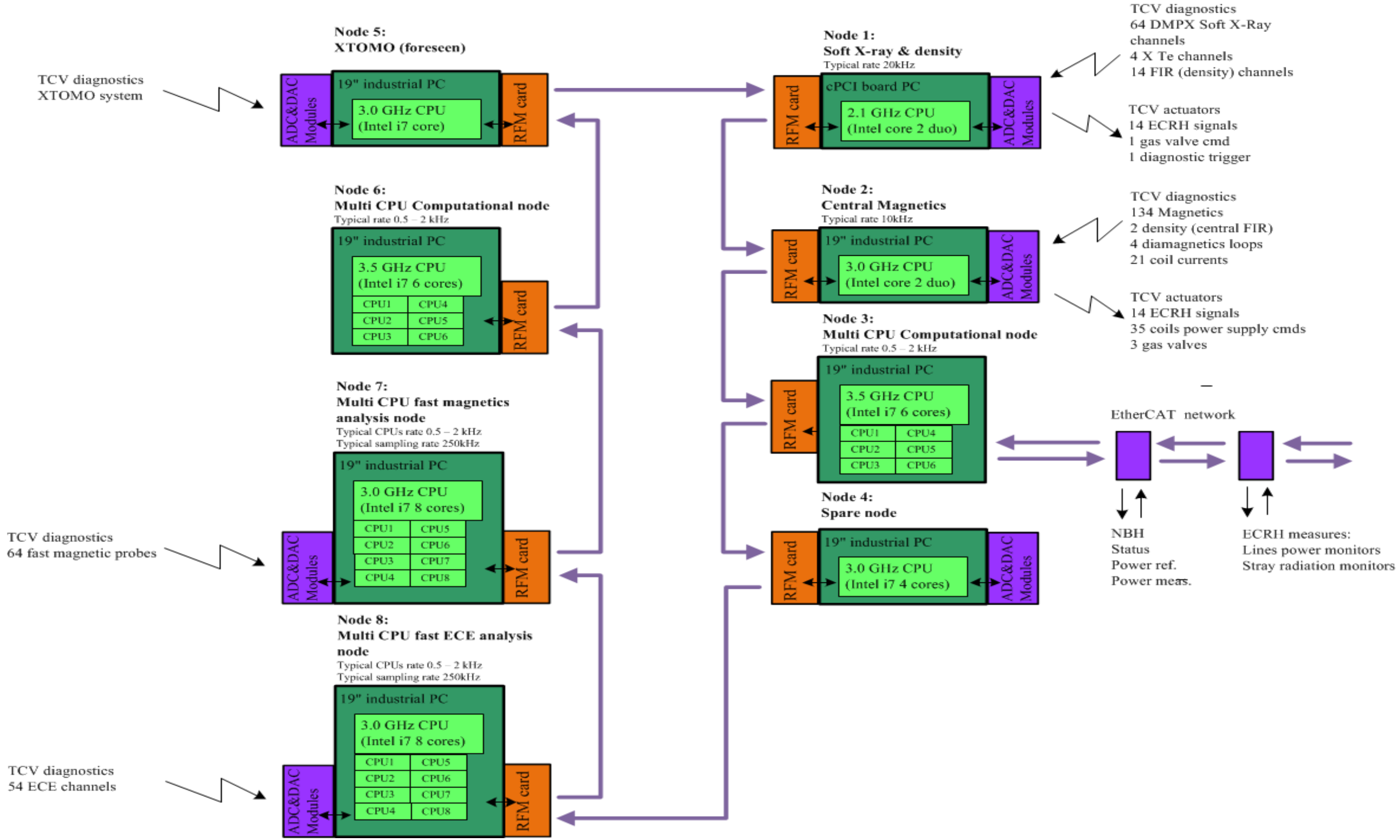




Galperti Cristian

Swiss Plasma Center

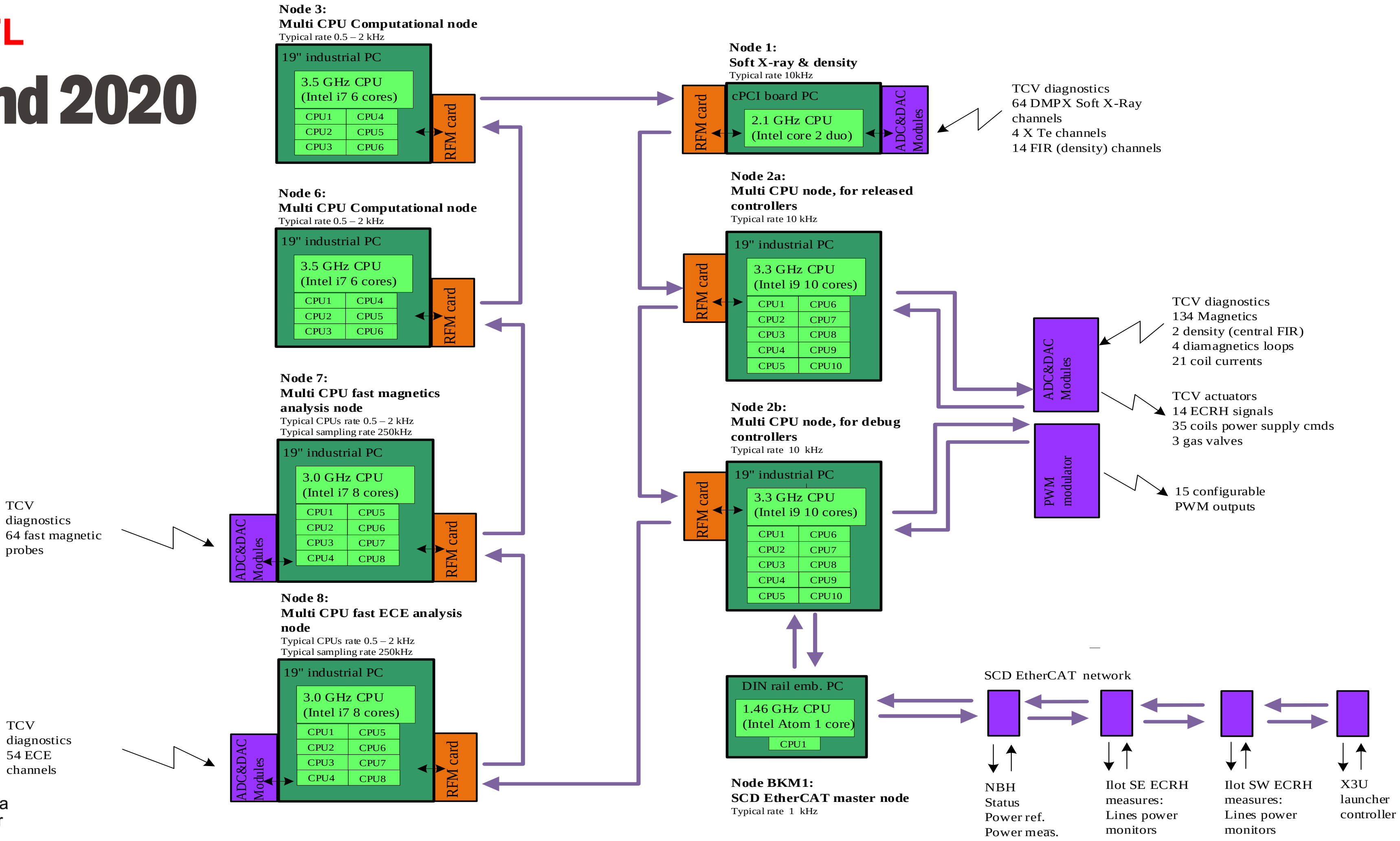
TCV Control system (SCD) in 2018



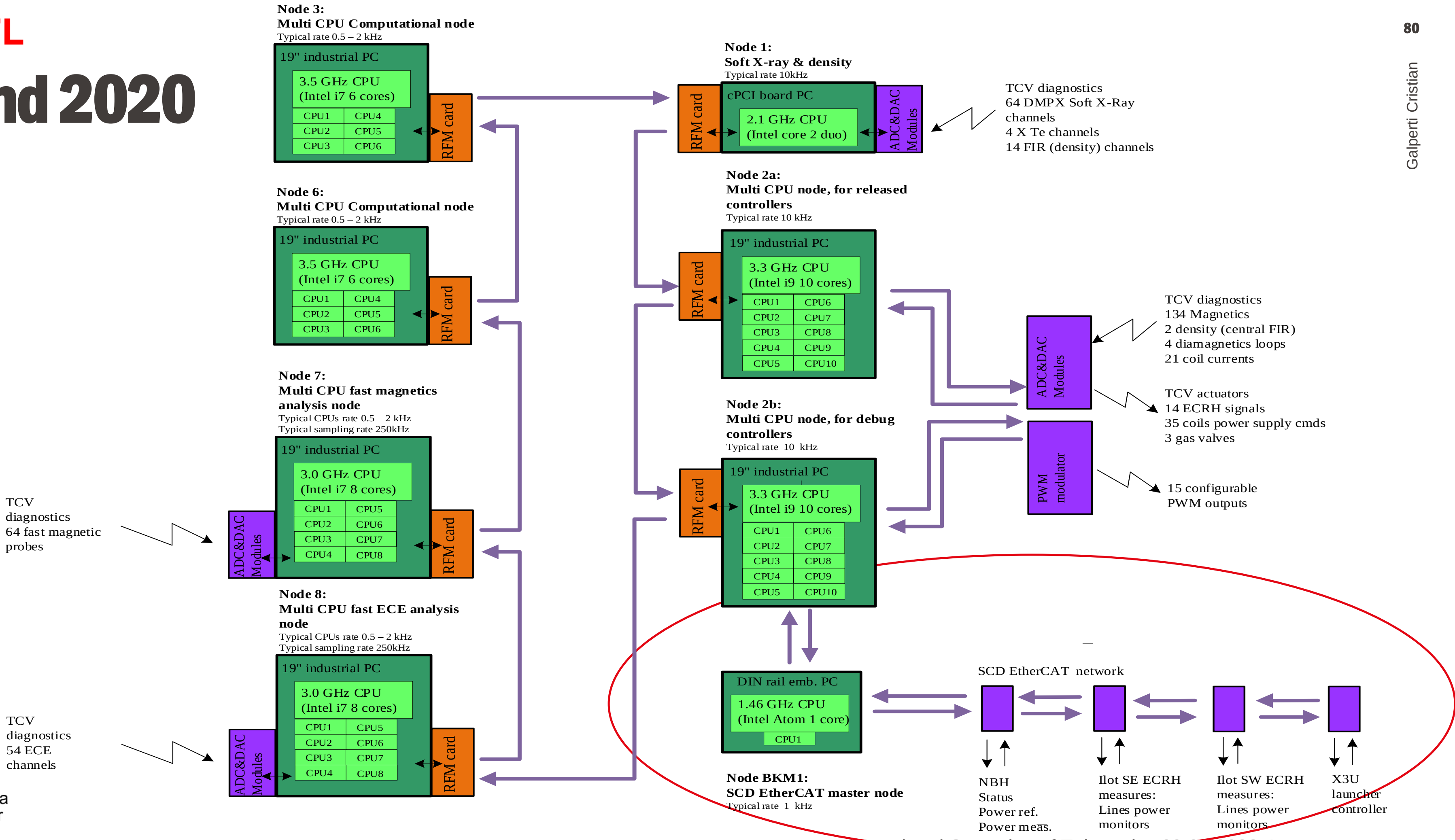
Swiss Plasma Center

Galperti Cristian

And 2020

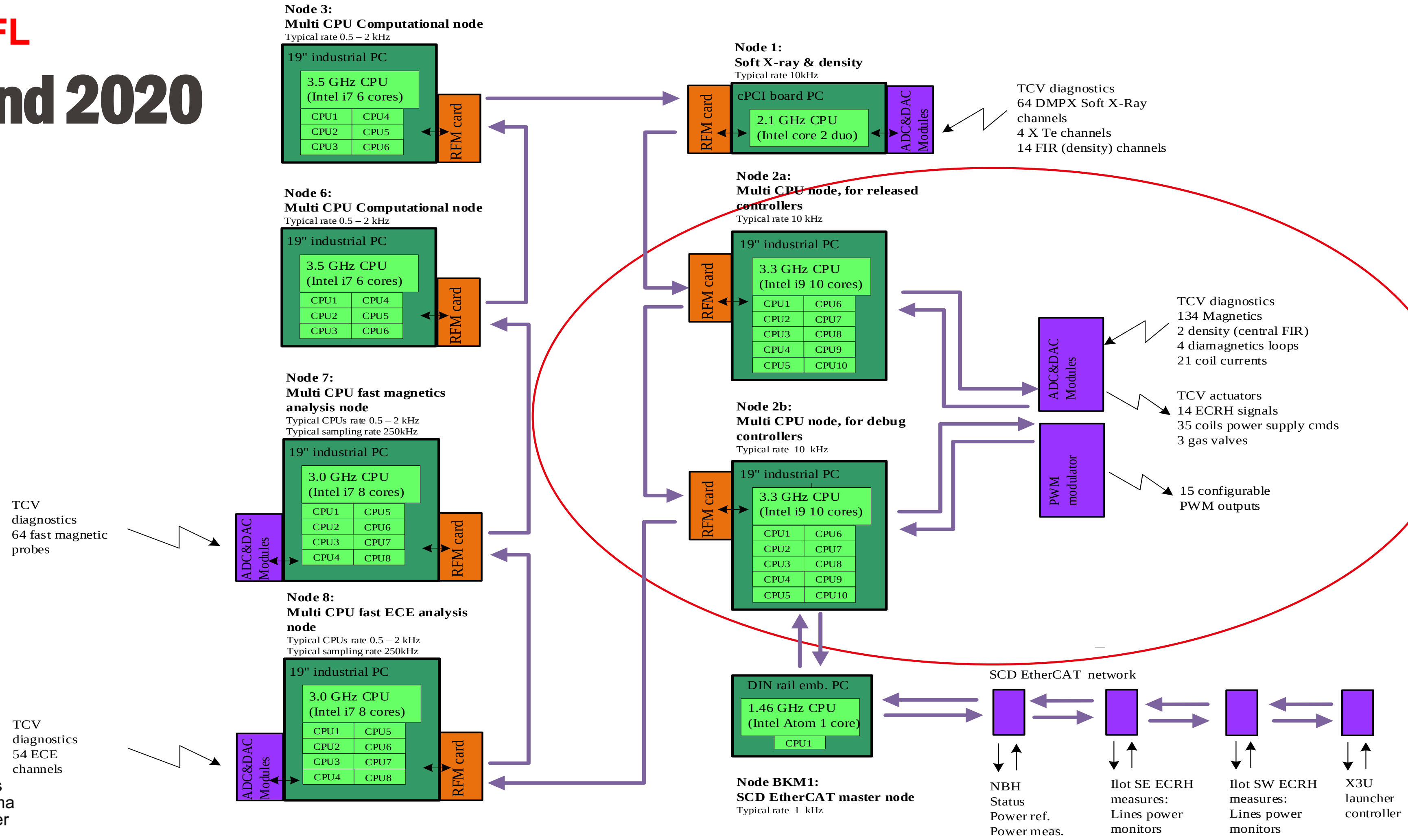

Node 3:
Multi CPU Computational node
Typical rate 0.5 – 2 kHz
19" industrial PC
3.5 GHz CPU
(Intel i7 6 cores)
CPU1 CPU4
CPU2 CPU5
CPU3 CPU6
RFM card
Node 6:
Multi CPU Computational node
Typical rate 0.5 – 2 kHz
19" industrial PC
3.5 GHz CPU
(Intel i7 6 cores)
CPU1 CPU4
CPU2 CPU5
CPU3 CPU6
RFM card
Node 7:
Multi CPU fast magnetics analysis node
Typical CPUs rate 0.5 – 2 kHz
Typical sampling rate 250kHz
19" industrial PC
3.0 GHz CPU
(Intel i7 8 cores)
CPU1 CPU5
CPU2 CPU6
CPU3 CPU7
CPU4 CPU8
ADC&DAC Modules
RFM card
TCV diagnostics
64 fast magnetic probes
Node 8:
Multi CPU fast ECE analysis node
Typical CPUs rate 0.5 – 2 kHz
Typical sampling rate 250kHz
19" industrial PC
3.0 GHz CPU
(Intel i7 8 cores)
CPU1 CPU5
CPU2 CPU6
CPU3 CPU7
CPU4 CPU8
ADC&DAC Modules
RFM card
TCV diagnostics
54 ECE channels
Node 1:
Soft X-ray & density
Typical rate 10kHz
RFM card
cPCI board PC
2.1 GHz CPU
(Intel core 2 duo)
ADC&DAC Modules
TCV diagnostics
64 DMPX Soft X-Ray channels
4 X Te channels
14 FIR (density) channels
Node 2a:
Multi CPU node, for released controllers
Typical rate 10 kHz
RFM card
19" industrial PC
3.3 GHz CPU
(Intel i9 10 cores)
CPU1 CPU6
CPU2 CPU7
CPU3 CPU8
CPU4 CPU9
CPU5 CPU10
Node 2b:
Multi CPU node, for debug controllers
Typical rate 10 kHz
RFM card
19" industrial PC
3.3 GHz CPU
(Intel i9 10 cores)
CPU1 CPU6
CPU2 CPU7
CPU3 CPU8
CPU4 CPU9
CPU5 CPU10
ADC&DAC Modules
TCV diagnostics
134 Magnetics
2 density (central FIR)
4 diamagnetics loops
21 coil currents
TCV actuators
14 ECRH signals
35 coils power supply cmds
3 gas valves
PWM modulator
15 configurable PWM outputs
DIN rail emb. PC
1.46 GHz CPU
(Intel Atom 1 core)
CPU1
Node BKM1:
SCD EtherCAT master node
Typical rate 1 kHz
SCD EtherCAT network
NBH
Status
Power ref.
Power meas.
Ilot SE ECRH
measures:
Lines power
monitors
Ilot SW ECRH
measures:
Lines power
monitors
X3U
launcher
controller

And 2020


Node 3:
Multi CPU Computational node
Typical rate 0.5 – 2 kHz
19" industrial PC
3.5 GHz CPU (Intel i7 6 cores)
CPU1 CPU4 CPU2 CPU5 CPU3 CPU6
RFM card
Node 6:
Multi CPU Computational node
Typical rate 0.5 – 2 kHz
19" industrial PC
3.5 GHz CPU (Intel i7 6 cores)
CPU1 CPU4 CPU2 CPU5 CPU3 CPU6
RFM card
Node 7:
Multi CPU fast magnetics analysis node
Typical CPUs rate 0.5 – 2 kHz
Typical sampling rate 250kHz
19" industrial PC
3.0 GHz CPU (Intel i7 8 cores)
CPU1 CPU5 CPU2 CPU6 CPU3 CPU7 CPU4 CPU8
ADC&DAC Modules
RFM card
TCV diagnostics
64 fast magnetic probes
Node 8:
Multi CPU fast ECE analysis node
Typical CPUs rate 0.5 – 2 kHz
Typical sampling rate 250kHz
19" industrial PC
3.0 GHz CPU (Intel i7 8 cores)
CPU1 CPU5 CPU2 CPU6 CPU3 CPU7 CPU4 CPU8
ADC&DAC Modules
RFM card
TCV diagnostics
54 ECE channels
Node 1:
Soft X-ray & density
Typical rate 10kHz
RFM card
cPCI board PC
2.1 GHz CPU (Intel core 2 duo)
ADC&DAC Modules
TCV diagnostics
64 DMPX Soft X-Ray channels
4 X Te channels
14 FIR (density) channels
Node 2a:
Multi CPU node, for released controllers
Typical rate 10 kHz
RFM card
19" industrial PC
3.3 GHz CPU (Intel i9 10 cores)
CPU1 CPU6 CPU2 CPU7 CPU3 CPU8 CPU4 CPU9 CPU5 CPU10
ADC&DAC Modules
TCV diagnostics
134 Magnetics
2 density (central FIR)
4 diamagnetics loops
21 coil currents
TCV actuators
14 ECRH signals
35 coils power supply cmds
3 gas valves
Node 2b:
Multi CPU node, for debug controllers
Typical rate 10 kHz
RFM card
19" industrial PC
3.3 GHz CPU (Intel i9 10 cores)
CPU1 CPU6 CPU2 CPU7 CPU3 CPU8 CPU4 CPU9 CPU5 CPU10
PWM modulator
15 configurable PWM outputs
DIN rail emb. PC
1.46 GHz CPU (Intel Atom 1 core)
CPU1
SCD EtherCAT network
Node BKM1:
SCD EtherCAT master node
Typical rate 1 kHz
NBH Status Power ref. Power meas.
Ilot SE ECRH measures: Lines power monitors
Ilot SW ECRH measures: Lines power monitors
X3U launcher controller

EPFL

And 2020



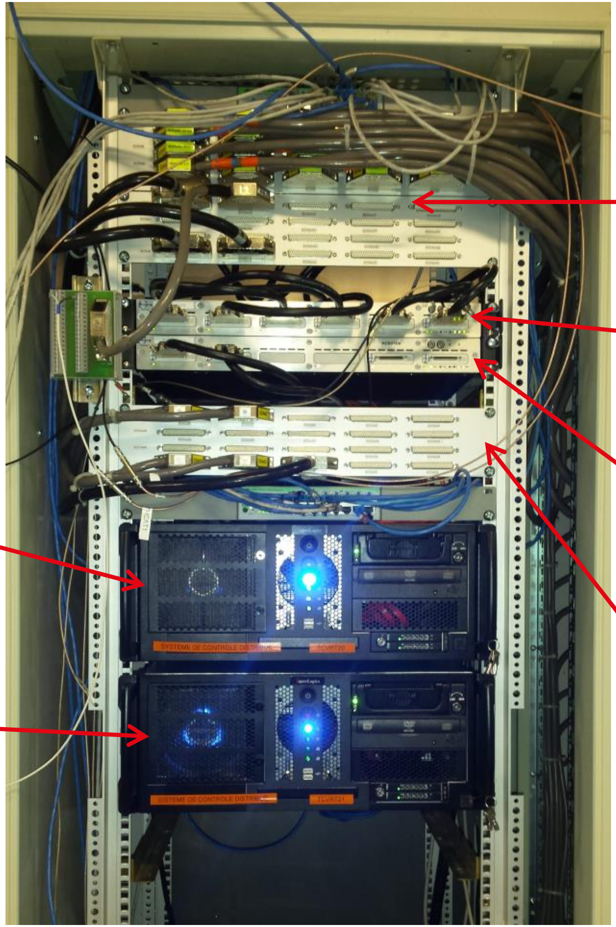
Galperti Cristian

Swiss Plasma Center

New node 02

Node 02 A
Industrial PC
(tcvrt20.crpp.tcv)

Node 02 B
Industrial PC
(tcvrt21.crpp.tcv)



Inputs patch panels

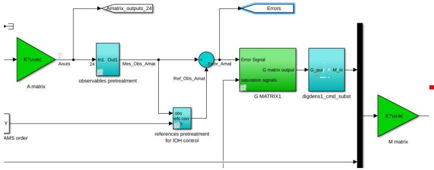
ADC system (acq2106_076)

DAC + PWM system (acq2106_079)

Outputs patch panel

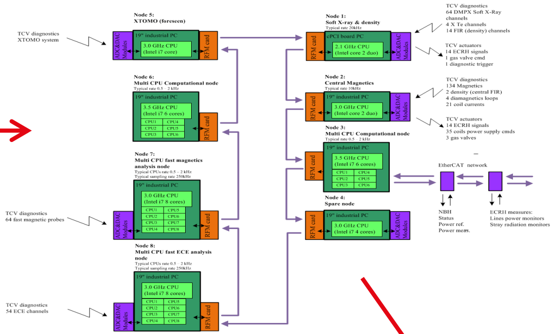
Old shot preparation workflow

MDS+ parameters and waveforms DB

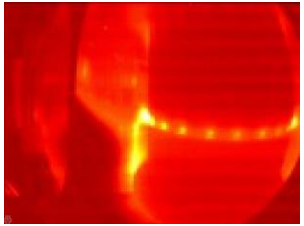


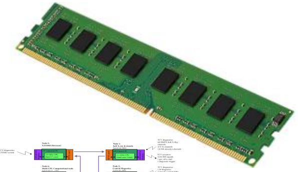
Shot configured Simulink block diagrams

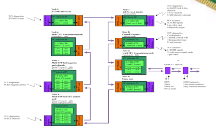
Shot configured control system



Plasma discharge







Discharge data on control computers RAM

Central server pulls all (binary, custom format) data and uploads them on MDS+

MDS+ shot data

What works and what not ?

Galperti Cristian

- Simulink guarantees algorithms maintainability, simulability and standardization.
- JIT compilation enables an enormous degree of flexibility.
- JIT guarantees code simulability w.r.t. shots
- Multi-computer architecture protects us against crash prone control codes.

- JIT compilation isn’t shot-frendly at all. It’s too slow, error-prone and heavy to maintain to cope with a tokamak shot sequence.
- JIT is a huge waste of resources for consolidated algorithms.
- In house HIC code demands too much maintenance efforts, preventing system extensibility.
- MDS+ RT data storage phase now is simply too complicated and involved, we need simplification and standardization.

How to improve, which constraints ?

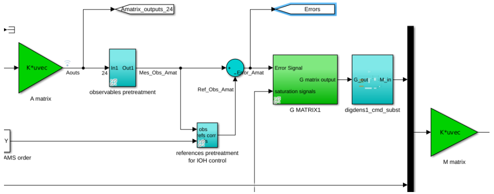
- Avoid JIT compilation, at least for consolidated algorithms
- Retain algorithms simulability
- Retain and tighten MATLAB/Simulink and MDS+ parameters and waveforms links
- Tighten hardware interface code and MDS+
- Retain TCV MDS+ DB (also used by many others actors)

2020 shot preparation

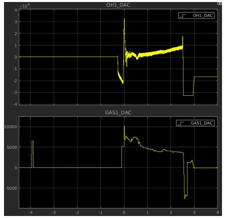
Galperti Cristian

MDS+ parameters and waveforms DB

Simulink model



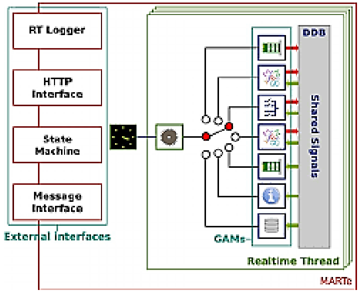
Simulink simulation



Common set of loader mechanism

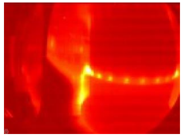
Automatic code generation

=



Control system

RT execution

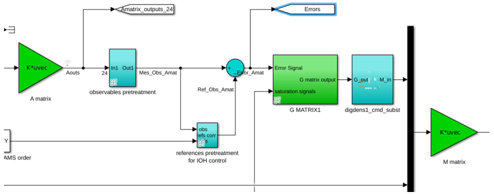


MDS+ shot data

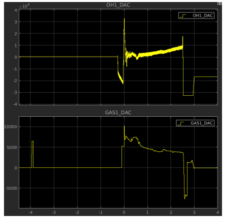
Swiss Plasma Center

2020 shot preparation

Simulink model



Simulink simulation



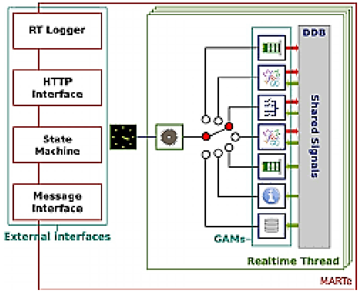
=

MDS+ parameters and waveforms DB

Common set of loader mechanism

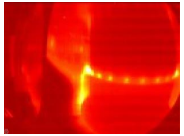
Automatic code generation

Equivalent function MATLAB and C++ loader classes => data driven RT application required



Control system

RT execution



MDS+ shot data

Swiss Plasma Center

Not JIT control code compilation

Galperti Cristian

MDS+ parameters and waveforms DB

A, B, w1(t) and **w2(t)** retrieved within MATLAB only for simulation

Simulink block diagram implementing:

Y(t) = **A***u1(t) + **B***u2(t) + w1(t) + w2(t)

Simulation

Simulink codegen and C comp. now embed automatically **data retrieval infos into the generated code**

A, B, w1(t) and **w2(t)** retrieved directly by the HIC before every shot

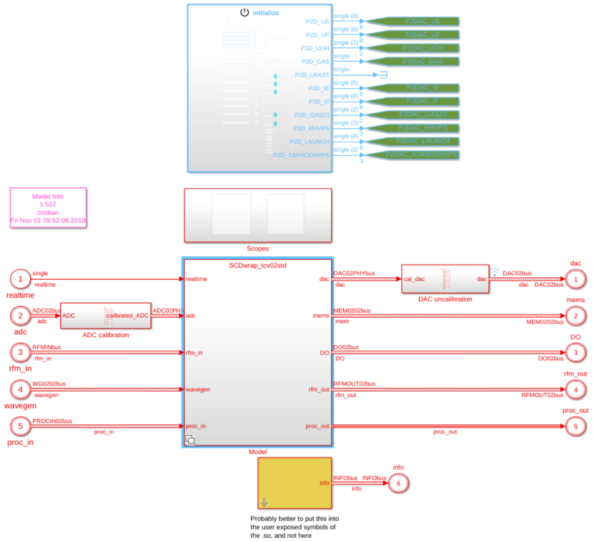
Object code executed by HIC:

MDS+ connect, take A,B,w1(1),w2(2) and then:

Y(t) = **A***u1(t) + **B***u2(t) + w1(t) + w2(t)

"MDS+ aware" Code generation from Simulink

Simulink algorithm with tunable parameters



Simulink coder with enabled CAPI + C compiler (gcc or icc)

Linux shared library

Introspective interface for I/O ports

Introspective interface for tunable parameters

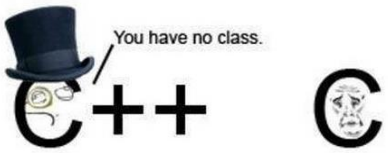
Introspective interface for internal states

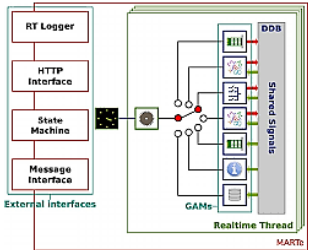
Introspective interface for internal signals

Galperti Cristian

EPFL

SCD hardware interface code at the turning point





In house developed C++ project

Pros:

- Full control on every part of the code.
- Fine tuning of RT performances (we deal with 100us cycle time machines).
- Code complexity can be kept low.
- Code functionalities can be tailored specifically for TCV.

Cons:

- Maintainable code architecture, skills and time needed.
- Used only here, no support form any community.
- **MDS+ integration must be developed on purpose**
- **Data driven application required for avoiding just in time code compilation**

F4E / MARTe2 framework

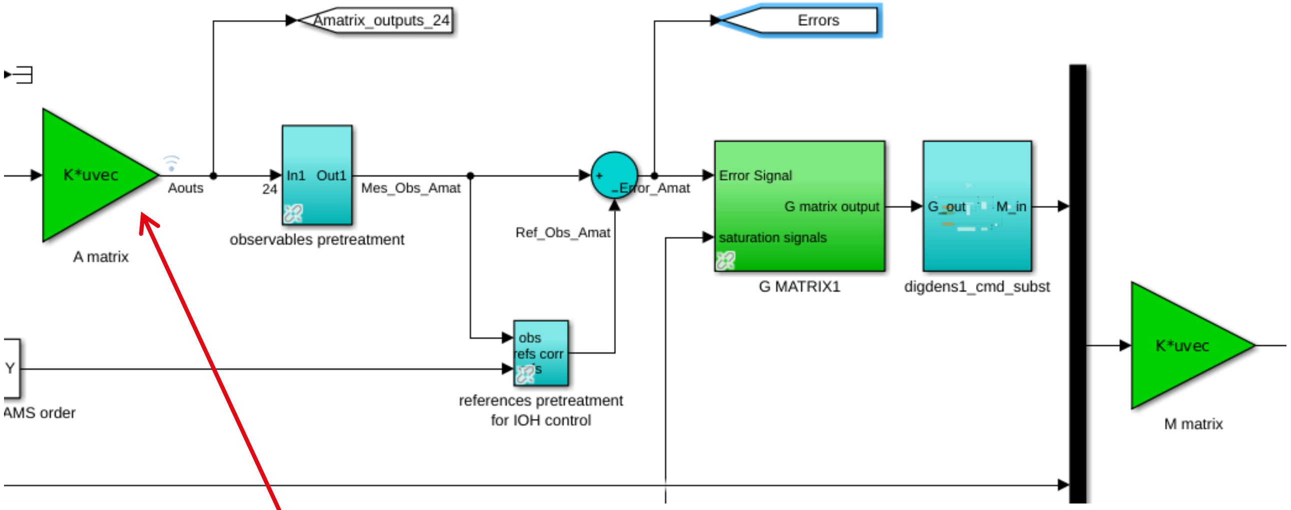
Pros:

- Comprehensive multi CPU control systems framework with a number of already developed functionalities: web interface, UDP logger, state machine, signal collection, **MDS+ read/write interface**
- Support from a community, **standardization of control nodes enforced.**
- **Data driven application natively supported**

Cons:

- ~~TCV specific interfaces not well supported (TDI, MDS+ and Simulink)~~
- Quite big multiclass C++ project to learn.
- ~~RT performances on sub-ms multi CPU systems to be assessed.~~

MDS+ -> MATLAB loaders classes




```
obj=obj.addparameter(SCDclass_mdsparmatrix          ('\pcs::phys_mat_a',                    'A_matrix'     ));
```

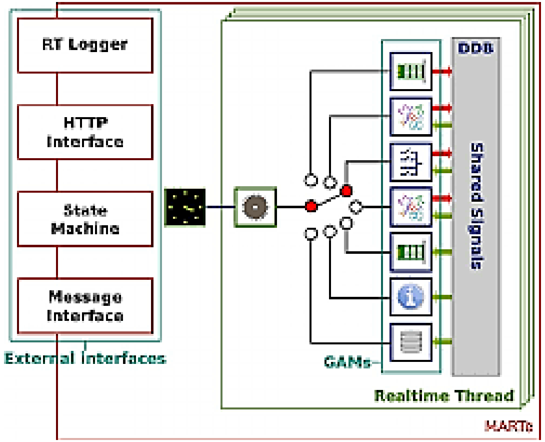
Upon model actualization, loader classes fetch actual parameters from MDS+ database(s) and actualize them in Simulink tunable parameters

MDS+ -> MARTe2 loader classes

Galperti Cristian

Upon MARTe2 start, loader classes fetch actual parameters from MDS+ database(s) and make them available within MARTe2 as referenceable objects, with methods for interacting with them

MARTe2




```
+SCDalgo_hybrid_tp-A_matrix    = { Class=MDSParMatrix          Path="\\pcs::phys_mat_a"                    }
```

MDS+

In this example, after this process, there is a instance of a MDSParMatrix class in MARTe2 whose name is "SCDalgo_hybrid_tp-A_matrix" holding the actual value of the parameter, retrieved from MDS+ at \\pcs::phys_mat_a

Swiss Plasma Center

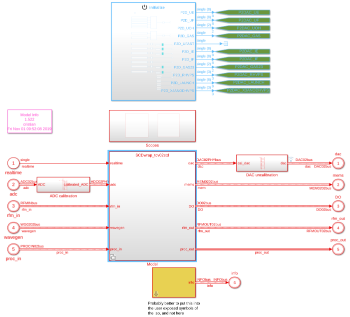
TCV Simulink plasma control simulation example

Matlab loader classes

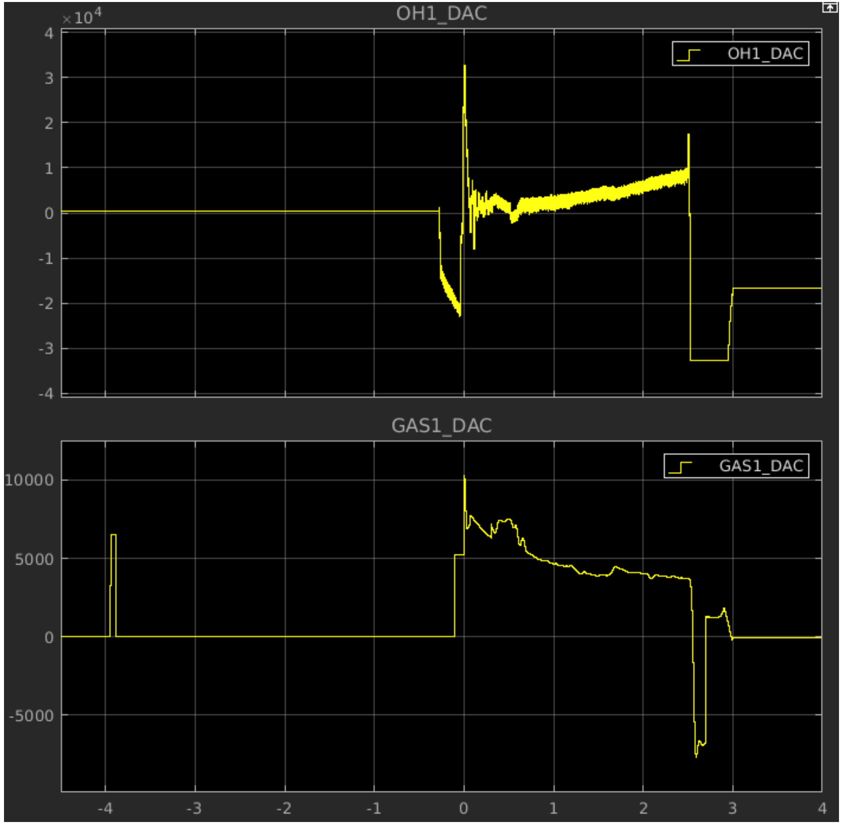
Shot parameters From MDS+

Shot ADC data From MDS+

Matlab loader classes



Shot simulation



Galperti Cristian

Swiss Plasma Center

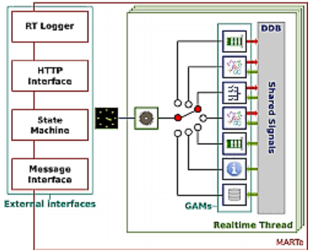
TCV real-time shot reprocessing with MARTe2 example

MARTe2
loader
classes

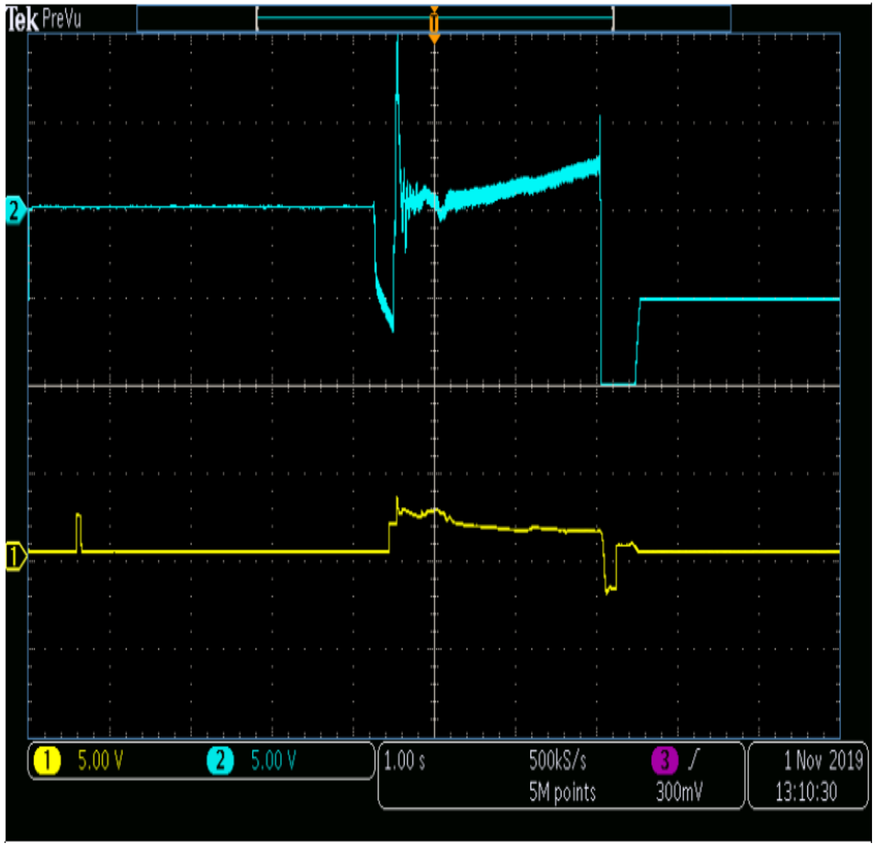
Shot parameters
From MDS+

Shot ADC data
From MDS+

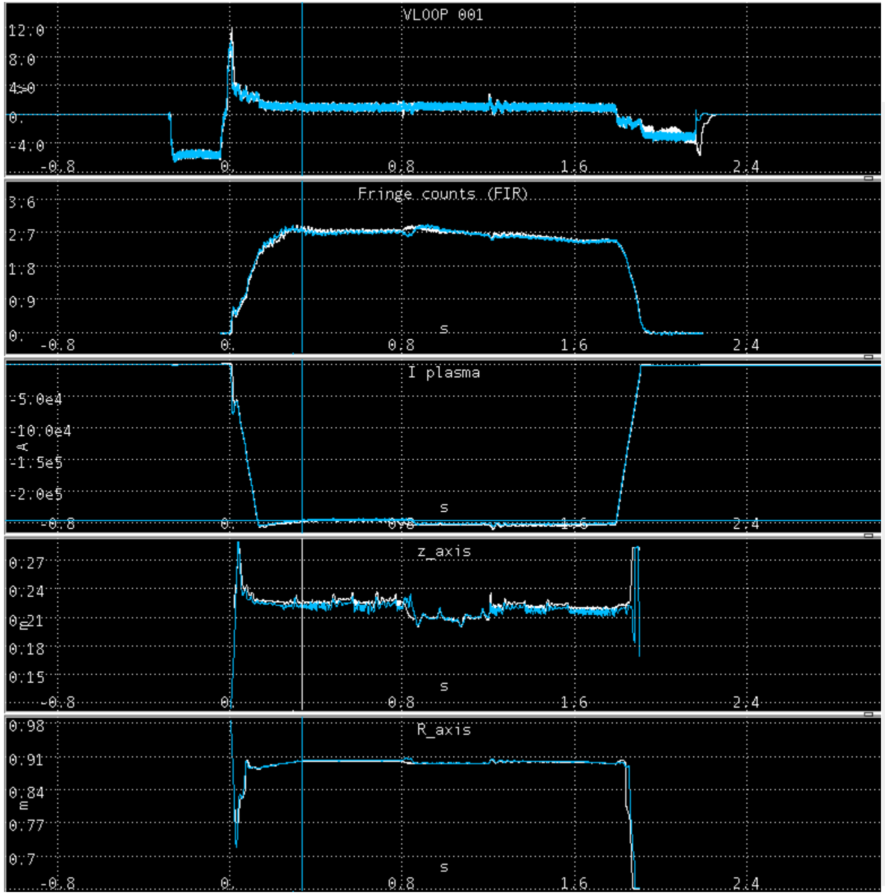
MARTe2
loader
classes

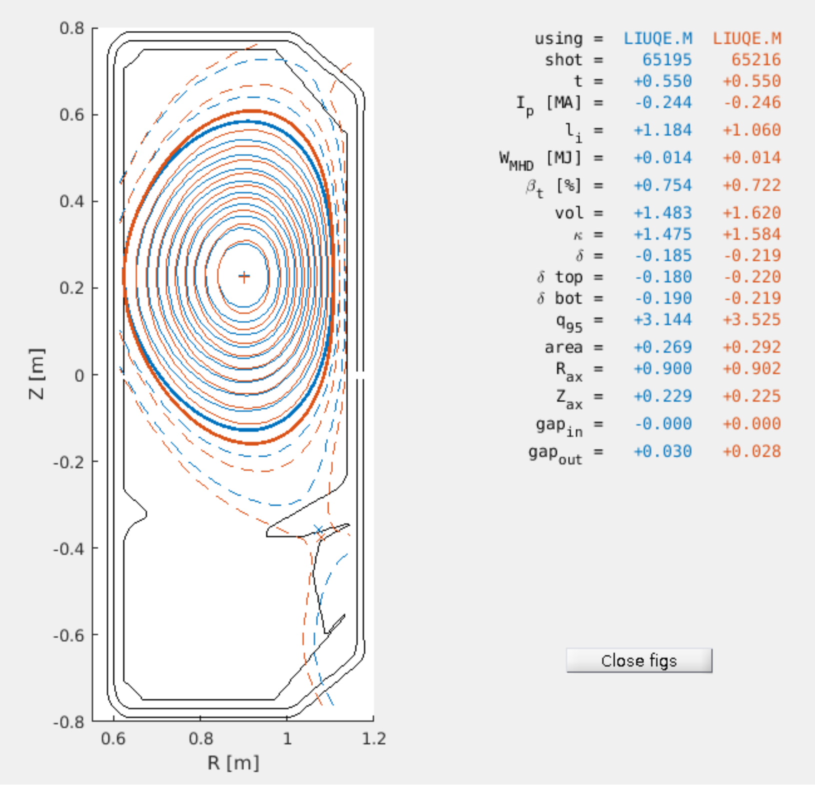


RT reprocess run



First shot controlling TCV (65195 vs. 65216)

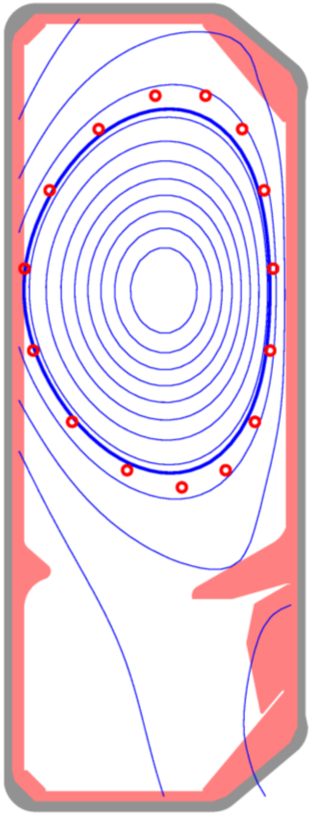




Swiss Plasma Center

Legacy analog controller vs. SCD

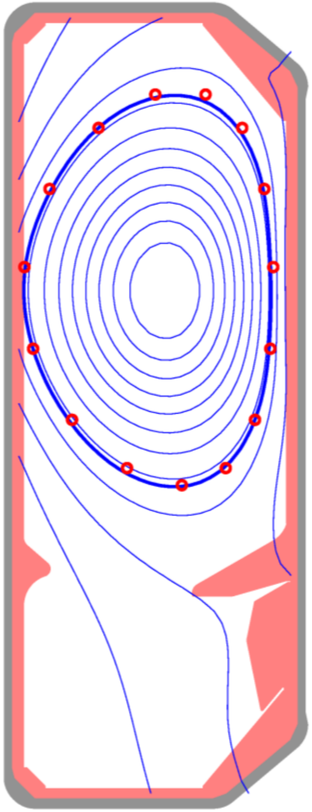
65195 0.55 s
hybrid
controller



The SCD clearly reproduces a better shape.

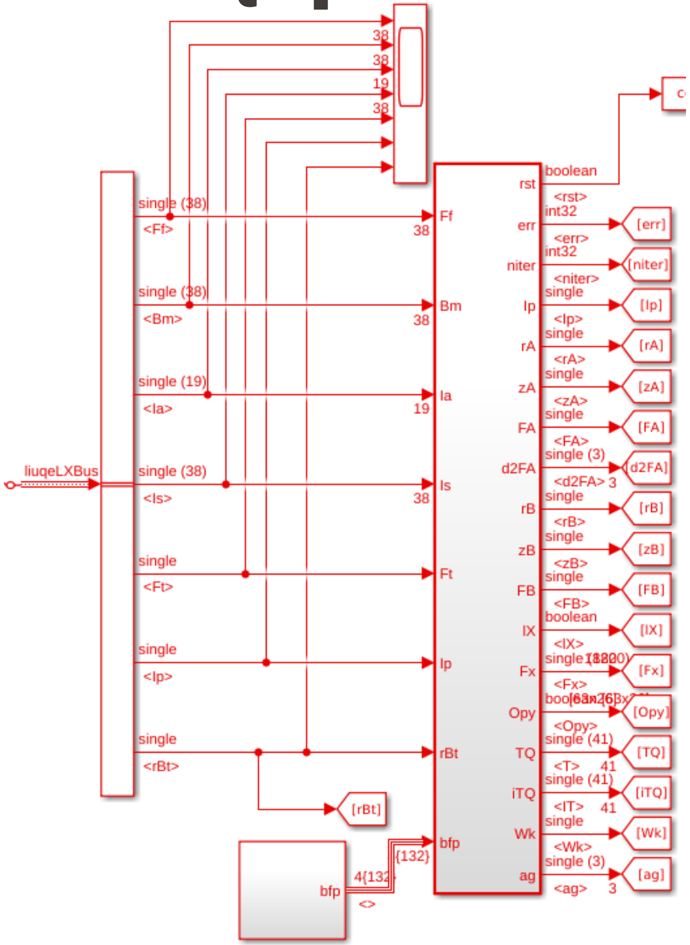
This is due (my speculations):

1) Analog offsets in old wavegens units
2) Analog offsets in hybrid matrix signal paths



65216 0.55 s
SCD with
hybrid
emulator

RTLIUQE ported to new node 02

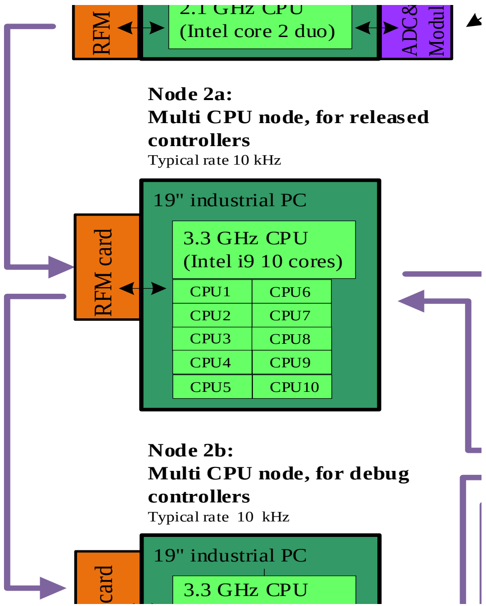


- Inserted on a second CPU (thread) of node 02 debug machine at 1 ms cycle time
- <250us execution time on a 65x28 flux grid
- Demonstrates feasibility of inter process communication at sub-ms speed on MARTe2
- Can be routinely run for every discharge when SCD is inserted

Galperti Cristian

Swiss Plasma Center

MST1 T06/T12 setup of node 02



- CPU1 (10kHz) : hybrid emulator (modified for taking gas command from SAM)
- CPU2 (1kHz): RT LIUQE
- CPU3 (1kHz): Supervisory and Actuator Manager code (SAM) configured with tasks for disruption avoidance experiments

- 10 kHz multi CPU system run for the first time
- Results in Olivier's and Trang Vu's presentations

Swiss Plasma Center

Status of SCD algorithms

✔

- Hybrid Emulator (fully interfaced with MDS+ PCS parameters)
- LIUQE-RT
- Supervisor & Actuator Manager (MDS+ interface pending)
- NBI interface
- Detachment Control (MDS+ interface pending)

✘

- Shape Control
- ECRH interface
- RAPTOR
- RAPTOR-LIUQE coupling
- RT-TORBEAM (forward)
- RT-TORBEAM (reverse)
- Sawtooth detector(s)
- Magnetics analysis (SVD, LM and standard)
- RT-ECE correlation
- RABBIT
- NTM controller
- RAPDENS
- ELM – LH observer

EPFL

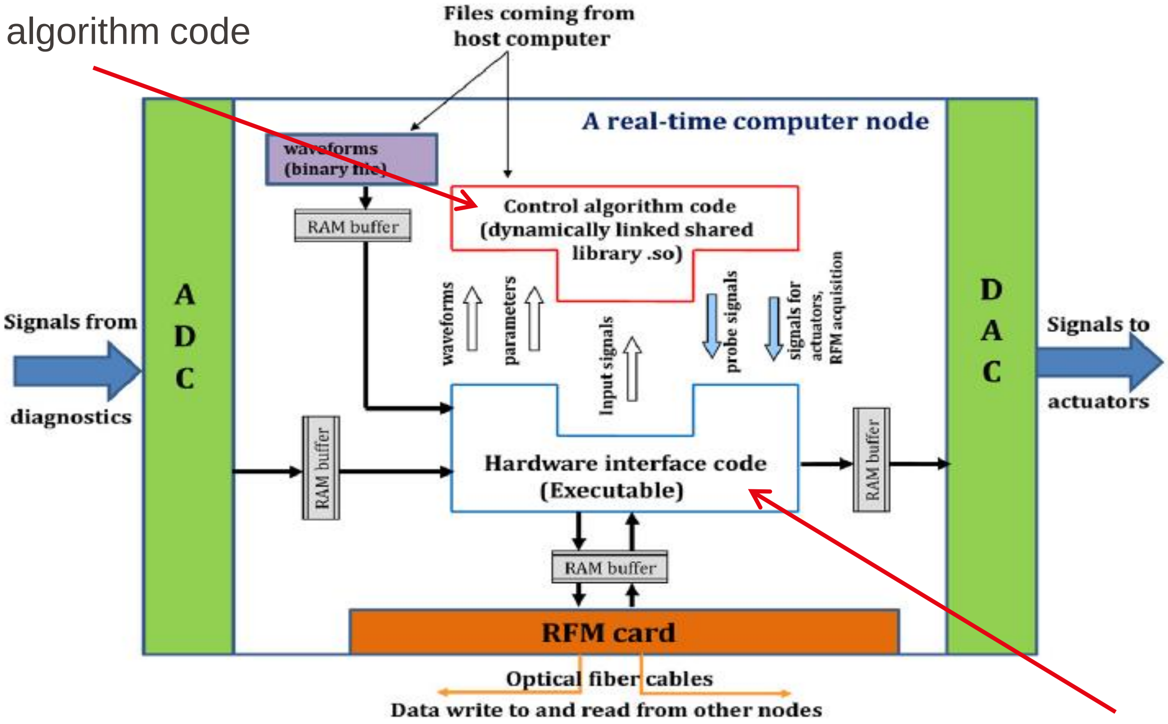
Outlook

- Port all control code to the new compile less approach in MATLAB/Simulink
- Port all rest of the nodes to MARTe2
- Integrate with tcvpc
- Commission new 64 bit node 01 in the new ilot NE
- Test a pilot “mini Thomson” in parallel with the present system ?

Control node architecture

Galperti Cristian

Control algorithm code

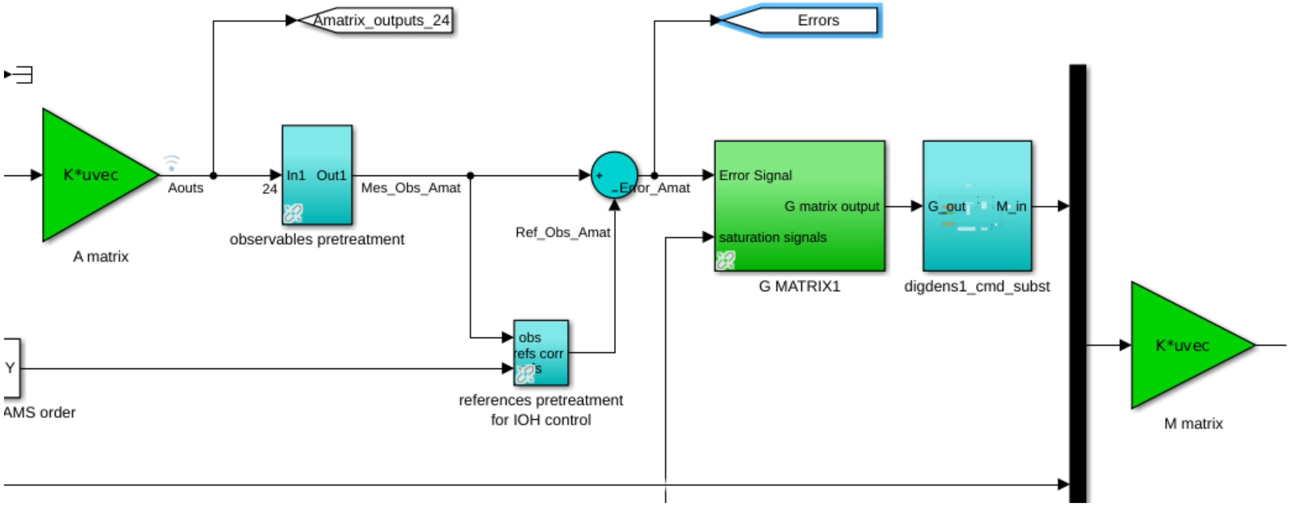


Hardware interface code (HIC from now on)

Swiss Plasma Center

MATLAB MDS+ loaders classes

Galperti Cristian

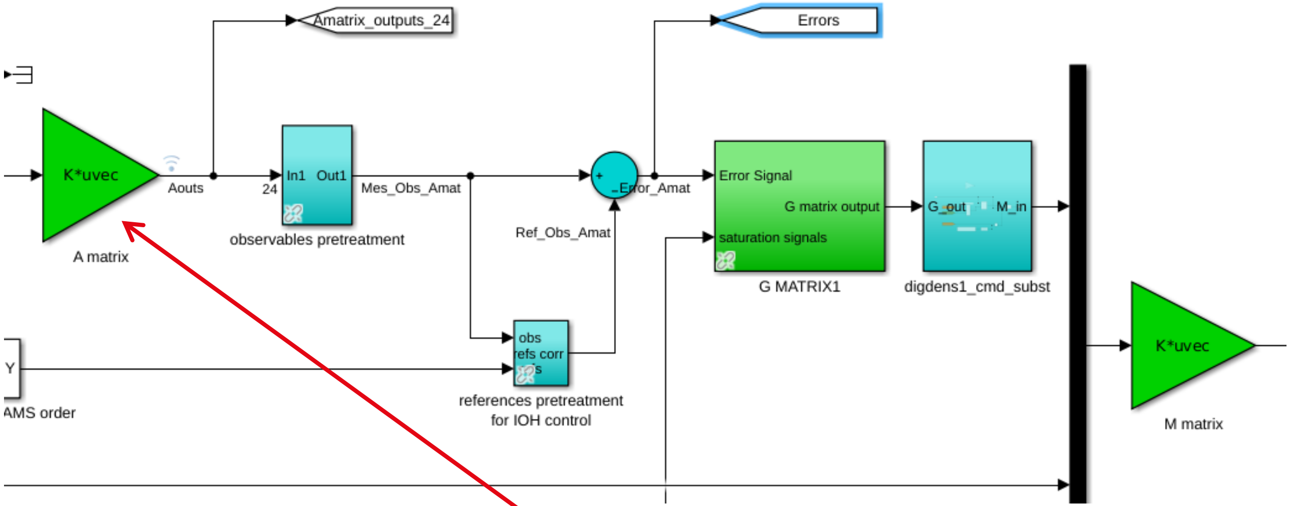



```
obj=SCDclass_algo('SCDalgo_hybrid');
obj=obj.settiming(-4.5,1e-4,3);
obj=obj.addtunparamstruct('SCDalgo_hybrid_tp');
obj=obj.addparameter(SCDclass_mdsparmatrix           ('\pcs::phys_mat_a',                        'A_matrix'    ));
obj=obj.addparameter(SCDclass_mdsparmatrix           ('\pcs::phys_mat_m',                        'M_matrix'    ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix         ('\pcs::phys_mat_g1',                       'G1_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix         ('\pcs::phys_mat_g2',                       'G2_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix         ('\pcs::phys_mat_g3',                       'G3_matrix'   ));
obj=obj.addparameter(SCDclass_mdsparfixdimvector     ('dim_of(\pcs::phys_mat_addresses:G)',      'G_time'   ,50));
obj=obj.addparameter(SCDclass_mdsparfixdimvectorint  ('\pcs::phys_mat_addresses:G',              'G_order'  ,50));
```

Swiss Plasma Center

MATLAB MDS+ loaders classes

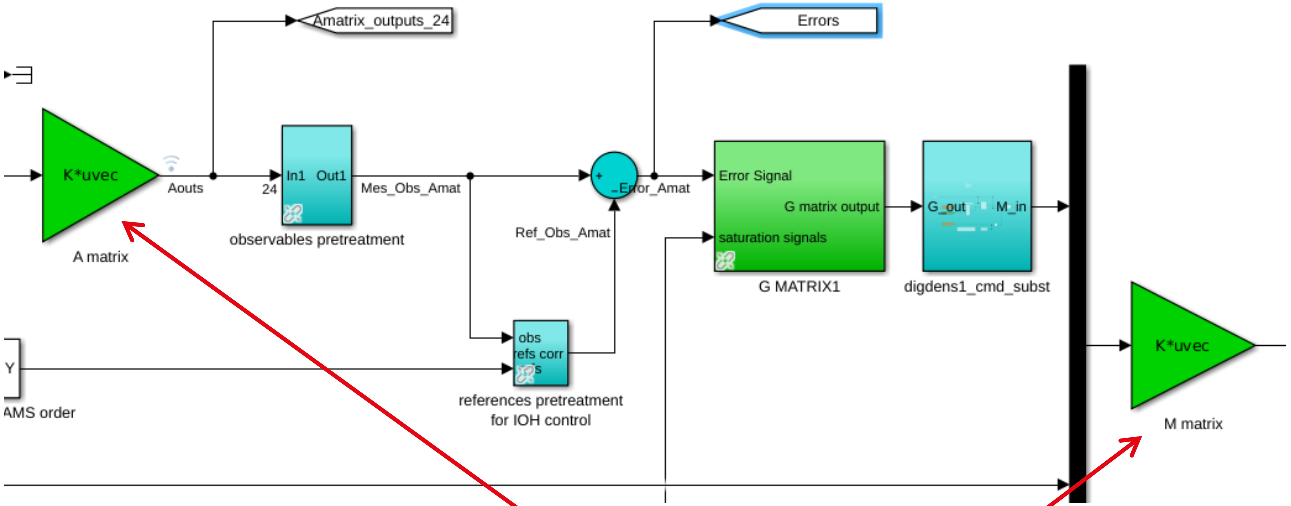
Galperti Cristian




```
obj=SCDclass_algo('SCDalgo_hybrid');
obj=obj.settiming(-4.5,1e-4,3);
obj=obj.addtunparamstruct('SCDalgo_hybrid_tp');
obj=obj.addparameter(SCDclass_mdsparmatrix          ('\pcs::phys_mat_a',                    'A_matrix'    ));
obj=obj.addparameter(SCDclass_mdsparmatrix          ('\pcs::phys_mat_m',                    'M_matrix'    ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g1',                   'G1_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g2',                   'G2_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g3',                   'G3_matrix'   ));
obj=obj.addparameter(SCDclass_mdsparfixdimvector    ('dim_of(\pcs::phys_mat_addresses:G)',  'G_time'   ,50));
obj=obj.addparameter(SCDclass_mdsparfixdimvectorint ('\pcs::phys_mat_addresses:G',          'G_order'  ,50));
```

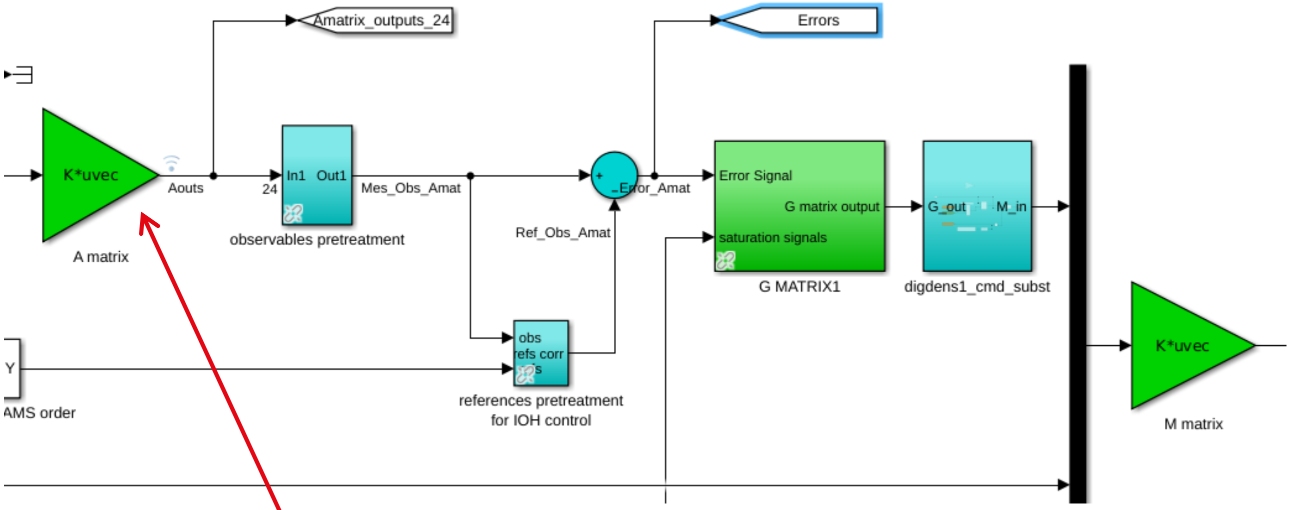
Swiss
Plasma
Center

MATLAB MDS+ loaders classes




```
obj=SCDclass_algo('SCDalgo_hybrid');
obj=obj.settiming(-4.5,1e-4,3);
obj=obj.addtunparamstruct('SCDalgo_hybrid_tp');
obj=obj.addparameter(SCDclass_mdsparmatrix          ('\pcs::phys_mat_a',                      'A_matrix'    ));
obj=obj.addparameter(SCDclass_mdsparmatrix          ('\pcs::phys_mat_m',                      'M_matrix'    ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g1',                     'G1_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g2',                     'G2_matrix'   ));
obj=obj.addparameter(SCDclass_mdspar3Dmatrix        ('\pcs::phys_mat_g3',                     'G3_matrix'   ));
obj=obj.addparameter(SCDclass_mdsparfixdimvector    ('dim_of(\pcs::phys_mat_addresses:G)',    'G_time'   ,50));
obj=obj.addparameter(SCDclass_mdsparfixdimvectorint ('\pcs::phys_mat_addresses:G',            'G_order'  ,50));
```

MATLAB MDS+ loaders classes




```
obj=obj.addparameter(SCDclass_mdsparmatrix    ('\pcs::phys_mat_a',    'A_matrix'    ));
```

Upon model actualization, loader classes fetch actual parameters from MDS+ database(s) and actualize them in Simulink tunable parameters

Swiss Plasma Center

And their translation in MARTe2

Galperti Cristian

MDSObjLoader instance, hosting a set of MDS+ connections

MDSObjConnection instance, hosting a set of MDS+ loader classes from that connection

MDSParMatrix instance, describing a link between a MDS+ parameter and a MARTe2 class which loads and holds its actual value

```
+MDSParameters = {
 Class=MDSObjLoader
 Shot=53600
 +Connection_tcvdata_tcv_shot = {
  Class=MDSObjConnection
  Server=tcvdata.epfl.ch
  Tree=tcv_shot
  +SCDalgo_hybrid_tp-A_matrix   = { Class=MDSParMatrix          Path="\\pcs::phys_mat_a"                       }
  +SCDalgo_hybrid_tp-M_matrix   = { Class=MDSParMatrix          Path="\\pcs::phys_mat_m"                       }
  +SCDalgo_hybrid_tp-G1_matrix  = { Class=MDSPar3DMatrix        Path="\\pcs::phys_mat_g1"                      }
  +SCDalgo_hybrid_tp-G2_matrix  = { Class=MDSPar3DMatrix        Path="\\pcs::phys_mat_g2"                      }
  +SCDalgo_hybrid_tp-G3_matrix  = { Class=MDSPar3DMatrix        Path="\\pcs::phys_mat_g3"                      }
  +SCDalgo_hybrid_tp-G_time     = { Class=MDSParFixDimVector    Path="dim_of(\\pcs::phys_mat_addresses:G)" Dim=50 }
  +SCDalgo_hybrid_tp-G_order    = { Class=MDSParFixDimVectorInt Path="\\pcs::phys_mat_addresses:G"         Dim=50 }
```

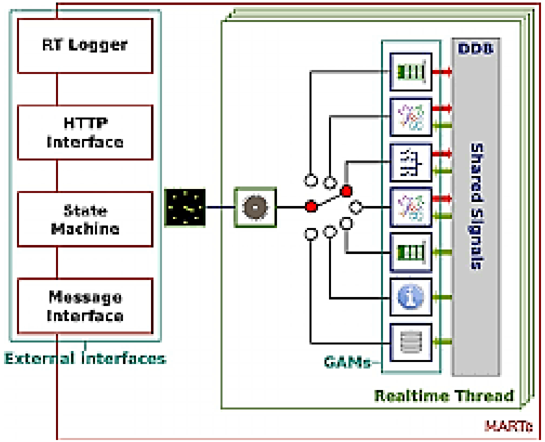
EPFL

MDS+ -> MARTe2 loader classes

Galperti Cristian

Upon MARTe2 start, loader classes fetch actual parameters from MDS+ database(s) and make them available within MARTe2 as referenceable objects, with methods for interacting with them

MARTe2




```
+SCDalgo_hybrid_tp-A_matrix    = { Class=MDSParMatrix          Path="\\pcs::phys_mat_a"                    }
```

MDS+

In this example, after this process, there is a instance of a MDSParMatrix class in MARTe2 whose name is “SCDalgo_hybrid_tp-A_matrix” holding the actual value of the parameter, retrieved from MDS+ at \\pcs::phys_mat_a

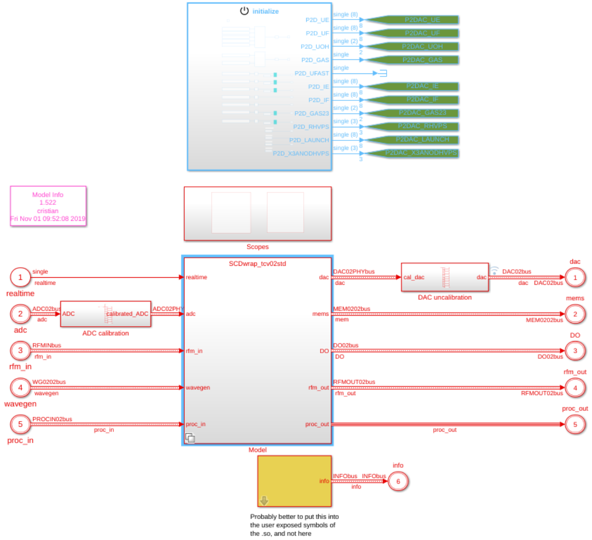
Swiss Plasma Center

EPFL

"MDS+ aware" Code generation from Simulink

Galperti Cristian

Simulink algorithm with tunable parameters



Simulink coder with enabled CAPI + C compiler (gcc or icc)

Linux shared library

Introspective interface for I/O ports

Introspective interface for tunable parameters

Introspective interface for internal states

Introspective interface for internal signals

Swiss Plasma Center

The SimulinkWrapperGAM

Name of the Linux .so library generated with MATLAB/Simulink code generator

Prefix name of Simulink global symbols to be retrieved from the .so

Name of the MDSObjLoader instance used to retrieve tunable parameters values

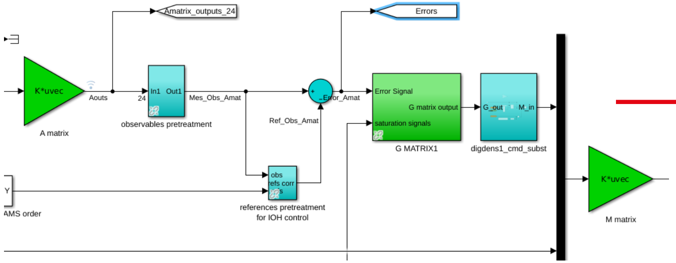
Simulink I/O ports mapped to MARTe2

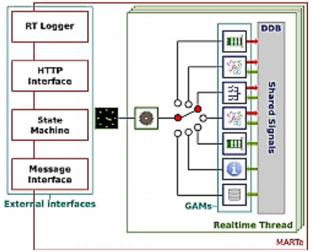
```
+GAMSimulink = {
 Class = SimulinkWrapperGAM
 Library = "/home/dt100/simulinkcodegen/SCD_rtccode_02_02.so"
 SymbolPrefix = "SCD_rtccode_02_02"
 Verbosity = 1
 TunParSourceGAM = MDSParameters
 SkipNotOkTunParams = 0
 InputSignals = {
  realtime    = { DataSource = DDB1 Type = float32  NumberOfElements = 1   CheckSimulinkType = true  NumberOfDimensions=1 }
  adc         = { DataSource = DDB1 Type = int16    NumberOfElements = 192 CheckSimulinkType = true  NumberOfDimensions=1 }
  rfm_in      = { DataSource = DDB1 Type = uint8    NumberOfElements = 664 CheckSimulinkType = false NumberOfDimensions=1 }
  wavegen     = { DataSource = DDB1 Type = float32  NumberOfElements = 62  CheckSimulinkType = true  NumberOfDimensions=1 }
  proc_in     = { DataSource = DDB1 Type = float32  NumberOfElements = 3   CheckSimulinkType = false NumberOfDimensions=1 }
 }
 OutputSignals = {
  dac         = { DataSource = DDB1 Type = int16    NumberOfElements = 64  CheckSimulinkType = true  NumberOfDimensions=1 }
  mem         = { DataSource = DDB1 Type = uint8    NumberOfElements = 156 CheckSimulinkType = false NumberOfDimensions=1 }
  DO          = { DataSource = DDB1 Type = uint8    NumberOfElements = 4   CheckSimulinkType = true  NumberOfDimensions=1 }
  rfm_out     = { DataSource = DDB1 Type = uint8    NumberOfElements = 640 CheckSimulinkType = false NumberOfDimensions=1 }
  proc_out    = { DataSource = DDB1 Type = float32  NumberOfElements = 1   CheckSimulinkType = false NumberOfDimensions=1 }
  info        = { DataSource = DDB1 Type = uint8    NumberOfElements = 16  CheckSimulinkType = false NumberOfDimensions=1 }
 }
 +SimulinkReadyMsg = { Class = Message Destination = RTApp.Data.MDSWriter Function = SetupBusSignals Mode = ExpectsReply
+Parameters = { Class = ConfigurationDatabase param1 = RTApp.Functions.GAMSimulink } }
}
```

Swiss Plasma Center

SimulinkWrapperGAM, parameters introspection

Galperti Cristian



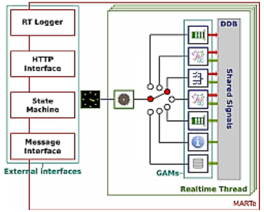



```
[Information - SimulinkWrapperGAM.cpp:363]: Allocating Simulink model dynamic memory
[Information - SimulinkWrapperGAM.cpp:387]: /home/dt100/simulinkcodegen/SCD_rtccode_02_02.so, number of main tunable parameters: 5
[Information - SimulinkWrapperGAM.cpp:708]: Simulink C API version number: 1
[Information - SimulinkWrapperGAM.cpp:751]: SCDalgo_hybrid_tp      , struct with 11 elems, size: 13068 bytes,                base addr: 0x00007FE4AD728E00
[Information - SimulinkWrapperGAM.cpp:914]: | A_matrix             , offset 0     , type float (4 bytes), ndims 2, dims [24,120],      addr: 0x7fe4ad728e00, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | M_matrix             , offset 11520, type float (4 bytes), ndims 2, dims [20,40],       addr: 0x7fe4ad72bb00, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | G1_matrix            , offset 14720, type float (4 bytes), ndims 3, dims [22,24,10],    addr: 0x7fe4ad72c780, orient: matrix col major nd
[Information - SimulinkWrapperGAM.cpp:914]: | G2_matrix            , offset 35840, type float (4 bytes), ndims 3, dims [22,24,10],    addr: 0x7fe4ad731a00, orient: matrix col major nd
[Information - SimulinkWrapperGAM.cpp:914]: | G3_matrix            , offset 56960, type float (4 bytes), ndims 3, dims [22,24,10],    addr: 0x7fe4ad736c80, orient: matrix col major nd
[Information - SimulinkWrapperGAM.cpp:914]: | G_time               , offset 78080, type float (4 bytes), ndims 2, dims [50,1],        addr: 0x7fe4ad73bf00, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | G_order              , offset 78280, type int (4 bytes)  , ndims 2, dims [50,1],        addr: 0x7fe4ad73bfc8, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | WG_start_time        , offset 78480, type float (4 bytes), ndims 2, dims [24,1],        addr: 0x7fe4ad73c090, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:806]: | mgams                , nested struct with 5 elems, offset: 78576,                base addr: 0x00007FE4AD73C0F0
[Information - SimulinkWrapperGAM.cpp:914]: | | mvloop             , offset 0     , type int (4 bytes)  , ndims 2, dims [1,1],         addr: 0x7fe4ad73c0f0, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | | ierat              , offset 4     , type int (4 bytes)  , ndims 2, dims [1,1],         addr: 0x7fe4ad73c0f4, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | | ikriz              , offset 8     , type int (4 bytes)  , ndims 2, dims [1,1],         addr: 0x7fe4ad73c0f8, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | | inova              , offset 12    , type int (4 bytes)  , ndims 2, dims [1,1],         addr: 0x7fe4ad73c0fc, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | | nfast              , offset 16    , type int (4 bytes)  , ndims 2, dims [1,1],         addr: 0x7fe4ad73c100, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | ensignbits           , offset 78596, type float (4 bytes), ndims 2, dims [1,1],         addr: 0x7fe4ad74f3f4, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:914]: | endigdens1           , offset 78600, type float (4 bytes), ndims 2, dims [1,1],         addr: 0x7fe4ad74f3f8, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:751]: SCDalgo_signbits_tp    , struct with 1 elems, size: 11520 bytes,                 base addr: 0x00007FE4AD726100
[Information - SimulinkWrapperGAM.cpp:914]: | A_matrix             , offset 0     , type float (4 bytes), ndims 2, dims [24,120],      addr: 0x7fe4ad726100, orient: matrix col major
[Information - SimulinkWrapperGAM.cpp:751]: SCDalgo_02stddiag_tp   , struct with 2 elems, size: 6096 bytes,                  base addr: 0x00007FE4AD724920
[Information - SimulinkWrapperGAM.cpp:806]: | ADCpre               , nested struct with 3 elems, offset: 0,                    base addr: 0x00007FE4AD724920
```

Swiss
Plasma
Center

SimulinkWrapperGAM, parameters actualization

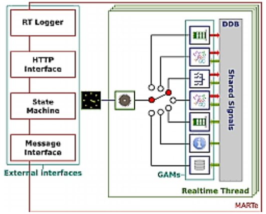
Galperti Cristian



1) Parameters are loaded into MARTe by the MDSObjLoader class, one time per run.

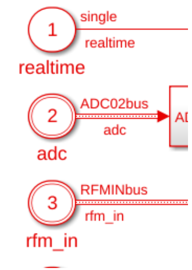
```
[Information - MDSObjConnection.cpp:81]: Connection_tcvdata_tcv_shot init, Server: tcvdata.epfl.ch, Tree: tcv_shot, number of subclasses: 49
[Information - MDSObjLoader.cpp:81]:MDSParameters connecting to server: tcvdata.epfl.ch, tree: tcv_shot, shot: 53600
...
[Information - MDSParameters.cpp:847]: MDSParMatrix actualize:   \pcs::phys_mat_a
  -> MDSParameters.Connection_tcvdata_tcv_shot.SCDalgo_hybrid_tp-A_matrix, vector numDims: 2, [24,120]
[Information - MDSParameters.cpp:847]: MDSParMatrix actualize:   \pcs::phys_mat_m
  -> MDSParameters.Connection_tcvdata_tcv_shot.SCDalgo_hybrid_tp-M_matrix, vector numDims: 2, [20,40]
[Information - MDSParameters.cpp:964]: MDSPar3DMatrix actualize: \pcs::phys_mat_g1
  -> MDSParameters.Connection_tcvdata_tcv_shot.SCDalgo_hybrid_tp-G1_matrix, vector numDims: 3, [22,24,10]
[Information - MDSParameters.cpp:964]: MDSPar3DMatrix actualize: \pcs::phys_mat_g2
  -> MDSParameters.Connection_tcvdata_tcv_shot.SCDalgo_hybrid_tp-G2_matrix, vector numDims: 3, [22,24,10]
[Information - MDSParameters.cpp:964]: MDSPar3DMatrix actualize: \pcs::phys_mat_g3
  -> MDSParameters.Connection_tcvdata_tcv_shot.SCDalgo_hybrid_tp-G3_matrix, vector numDims: 3, [22,24,10]
```

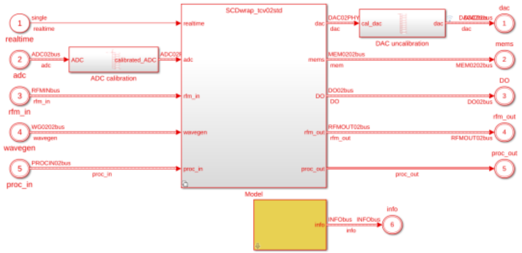
2) Parameters are actualized into exposed Simulink tunable parameters by the instances of SimulinkWrapperGAM

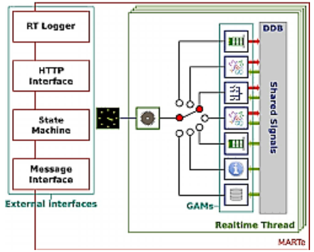



```
[Information - SimulinkWrapperGAM.cpp:632]: Parameter SCDalgo_hybrid_tp-A_matrix      correctly actualized
[Information - SimulinkWrapperGAM.cpp:632]: Parameter SCDalgo_hybrid_tp-M_matrix      correctly actualized
[Information - SimulinkWrapperGAM.cpp:632]: Parameter SCDalgo_hybrid_tp-G1_matrix     correctly actualized
[Information - SimulinkWrapperGAM.cpp:632]: Parameter SCDalgo_hybrid_tp-G2_matrix     correctly actualized
[Information - SimulinkWrapperGAM.cpp:632]: Parameter SCDalgo_hybrid_tp-G3_matrix     correctly actualized
...
[Information - SimulinkWrapperGAM.cpp:634]: Parameter SCDalgo_hybrid_tp-ensignbits    unlinked, using compile time value
[Information - SimulinkWrapperGAM.cpp:634]: Parameter SCDalgo_hybrid_tp-endigdens1    unlinked, using compile time value
```

SimulinkWrapperGAM, I/O ports introspection








```
[Information - SimulinkWrapperGAM.cpp:397]: /home/dt100/simulinkcodegen/SCD_rtccode_02_02.so, number of root inputs: 5
[Information - SimulinkWrapperGAM.cpp:1203]: realtime                    , offset 0, type float (4 bytes), ndims 2, dims [1,1]           , addr: 0x164e8890, orient: scalar
[Information - SimulinkWrapperGAM.cpp:1046]: adc                         , struct with 192 elems, size: 384 bytes,           base addr: 0x00000000164E8894
[Information - SimulinkWrapperGAM.cpp:1203]:  | ctlint_bpol_avg_001      , offset 0  , type short (2 bytes), ndims 2, dims [1,1]         , addr: 0x164e8894, orient: scalar
[Information - SimulinkWrapperGAM.cpp:1203]:  | ctlint_bpol_avg_002      , offset 2  , type short (2 bytes), ndims 2, dims [1,1]         , addr: 0x164e8896, orient: scalar
...
[Information - SimulinkWrapperGAM.cpp:1046]: rfm_in                      , struct with 6 elems, size: 664 bytes,             base addr: 0x00000000164E8A14
[Information - SimulinkWrapperGAM.cpp:1103]:  | Node01_RFM               , nested struct with 2 elems, offset: 0,            base addr: 0x00000000164E8A14
[Information - SimulinkWrapperGAM.cpp:1203]:  | | a1                     , offset 0  , type unsigned int (4 bytes), ndims 2, dims [1,1], addr: 0x164e8a14, orient: scalar
[Information - SimulinkWrapperGAM.cpp:1203]:  | | zero                   , offset 4  , type float (4 bytes), ndims 2, dims [1,1]         , addr: 0x164e8a18, orient: scalar
[Information - SimulinkWrapperGAM.cpp:1103]:  | Node02_RFM               , nested struct with 10 elems, offset: 8, base addr:          0x00000000164E8A1C
[Information - SimulinkWrapperGAM.cpp:1203]:  | | flags                  , offset 0  , type unsigned int (4 bytes), ndims 2, dims [1,1], addr: 0x164e8a1c, orient: scalar
[Information - SimulinkWrapperGAM.cpp:1203]:  | | Ff                     , offset 4  , type float (4 bytes), ndims 2, dims [38,1]        , addr: 0x164e8a20, orient: vector
[Information - SimulinkWrapperGAM.cpp:1203]:  | | Bm                     , offset 156, type float (4 bytes), ndims 2, dims [38,1]        , addr: 0x164e8ab8, orient: vector
[Information - SimulinkWrapperGAM.cpp:1203]:  | | IPF                    , offset 308, type float (4 bytes), ndims 2, dims [16,1]        , addr: 0x164e8b50, orient: vector
[Information - SimulinkWrapperGAM.cpp:1203]:  | | IG                     , offset 372, type float (4 bytes), ndims 2, dims [1,1]         , addr: 0x164e8b90, orient: scalar
...
[Information - SimulinkWrapperGAM.cpp:405]: /home/dt100/simulinkcodegen/SCD_rtccode_02_02.so, configured input/output ports:
[Information - SimulinkClasses.cpp:380]: IN  port, name: realtime, at 0x164e8890, elems 1   , size 4   , homogeneus type, float(4)
[Information - SimulinkClasses.cpp:380]: IN  port, name: adc     , at 0x164e8894, elems 192 , size 384 , homogeneus type, short(2)
[Information - SimulinkClasses.cpp:380]: IN  port, name: rfm_in  , at 0x164e8a14, elems 166 , size 664 , mixed types
[Information - SimulinkClasses.cpp:380]: IN  port, name: wavegen , at 0x164e8cac, elems 62  , size 248 , homogeneus type, float(4)
[Information - SimulinkClasses.cpp:380]: IN  port, name: proc_in , at 0x164e8da4, elems 3   , size 12  , homogeneus type, float(4)
[Information - SimulinkClasses.cpp:380]: OUT port, name: dac     , at 0x164e8dc0, elems 64  , size 128 , homogeneus type, short(2)
[Information - SimulinkClasses.cpp:380]: OUT port, name: mem     , at 0x164e8e40, elems 39  , size 156 , mixed types
[Information - SimulinkClasses.cpp:380]: OUT port, name: DO      , at 0x164e8edc, elems 4   , size 4   , homogeneus type, unsigned char(1)
[Information - SimulinkClasses.cpp:380]: OUT port, name: rfm_out , at 0x164e8ee0, elems 160 , size 640 , mixed types
[Information - SimulinkClasses.cpp:380]: OUT port, name: proc_out, at 0x164e9160, elems 1   , size 4   , homogeneus type, float(4)
[Information - SimulinkClasses.cpp:380]: OUT port, name: info    , at 0x164e9168, elems 3   , size 16  , mixed types
```

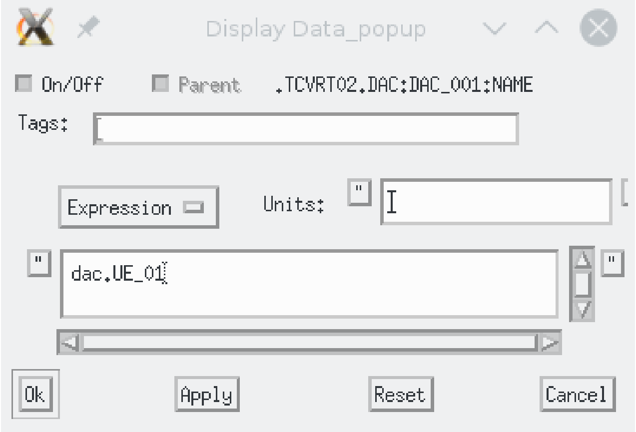
Galperti Cristian

Simulink signals info automatic storage

Modified version of the released MDSWriter DataSource

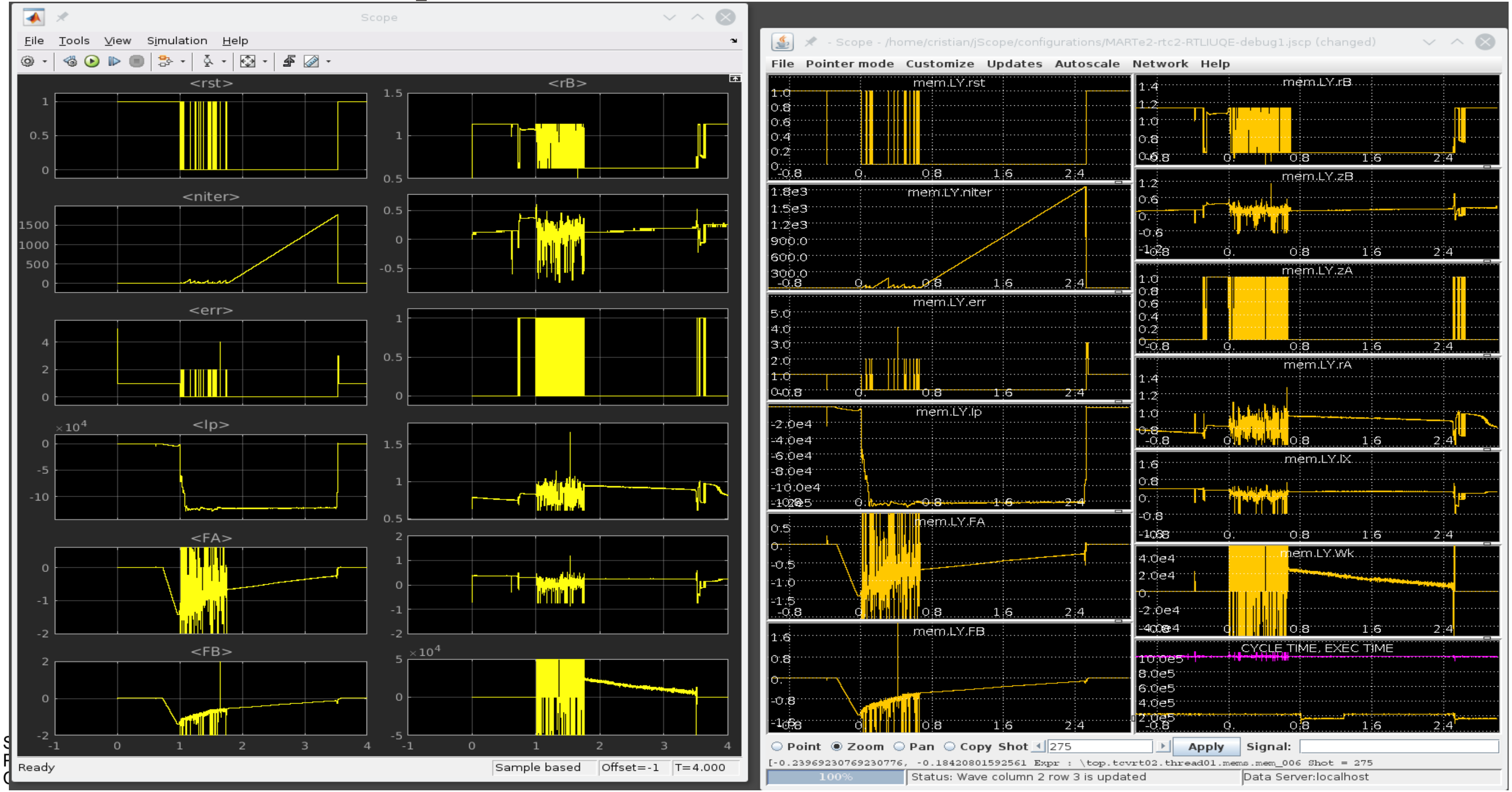
```
+MDSWriter = {
 Class = MDSSPCWriter
 NumberOfBuffers = 100000
 CPUMask = 0x10
 StackSize = 10000000
 TreeName = "rtc2"
 StoreOnTrigger = 1
 Verbosity = 0
 PulseNumber = -2
 Signals = {
  Trigger = { Type = uint8 }
  Time    = { Type = int32 TimeSignal = 1 TimeSignalMultiplier = 1e-6 }
  dac     = {
        NodeName = "TCVRT02.DAC.DAC_%03d.RAW"
        Indexed = 1
        BusSrc = RTApp.Functions.GAMSimulink
        BusName = dac
        NamePath = ".-.NAME"
        Period = 0.0001
        MakeSegmentAfterNWrites = 100000
        AutomaticSegmentation = 0
        NumberOfElements = 64
        SamplePhase = 0
  }
}
```

printf style string escape numeric for addressing multiple MDS+ channels



It takes signal information (presently only the name) from exposed methods of the given SimulinkWrapperGAM instance

RT LIUQE reprocess run results



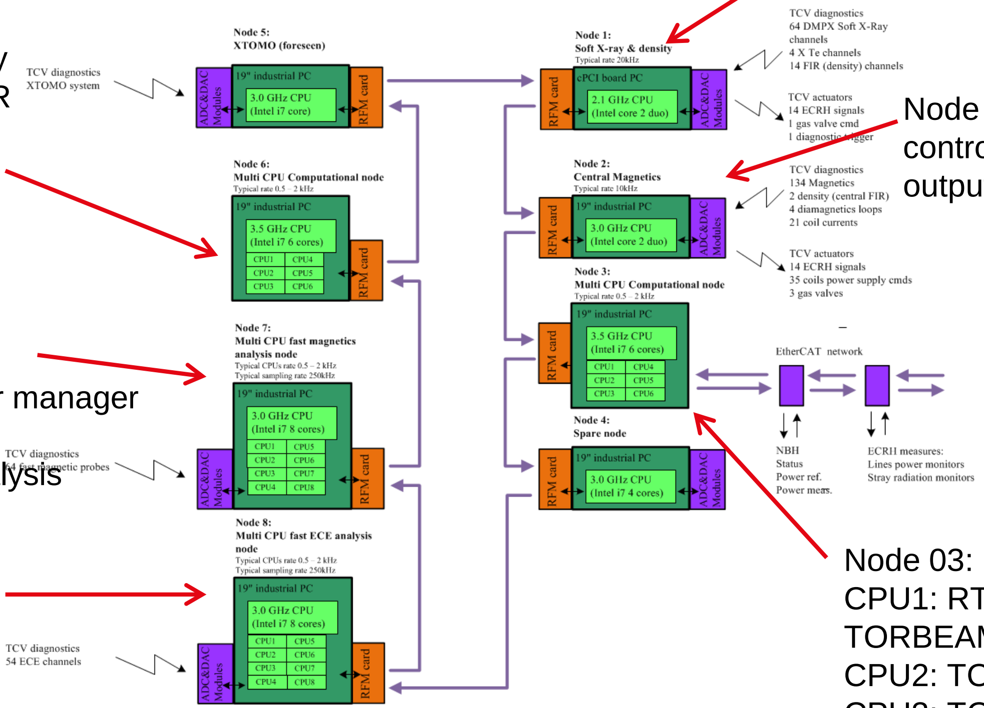
Galperti Cristian

SCD in integrated control

Node 01: Xte, FIR and DMPX pre-processing

Node 06:
CPU1: FIR profile based density control + RTLIUQE for RAPTOR
CPU2: RAPTOR simulator
CPU3: RAPTOR observer
CPU4: MPC beta and profile controller

Node 02: Hybrid emulator + shape control, central signal switch and outputs to the plant.



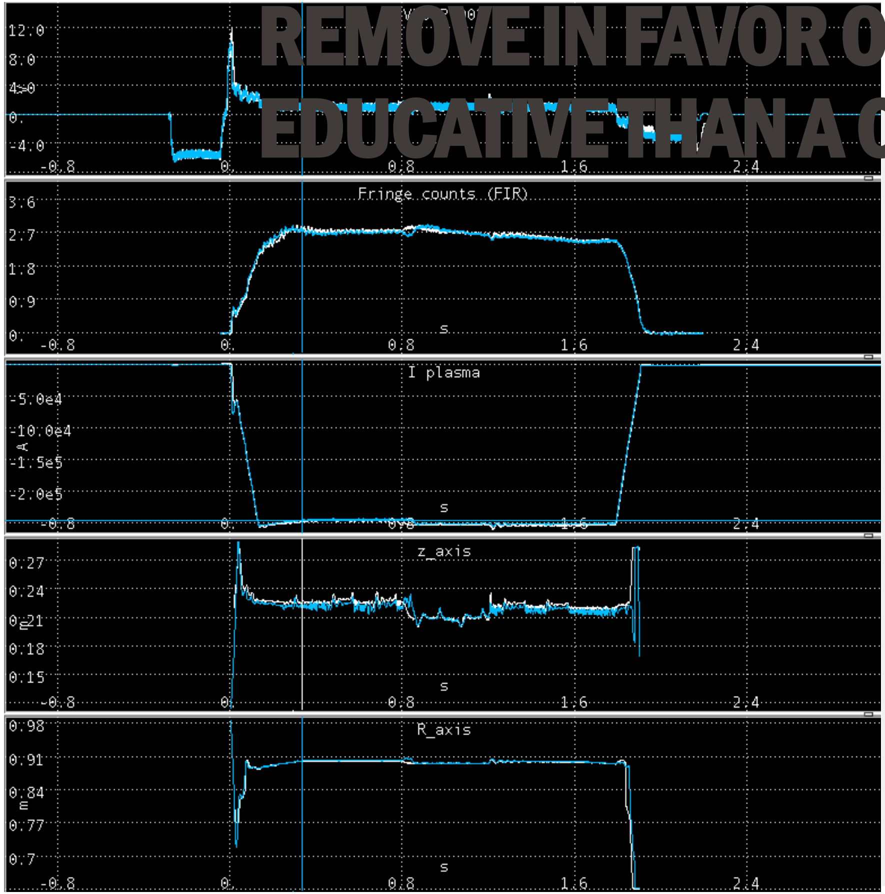
Node 07:
CPU1: MHD (rotating) analysis,
Plasma state observer+actuator manager
CPU2: MHD (rotating) analysis
CPU3: MHD (locked mode) analysis

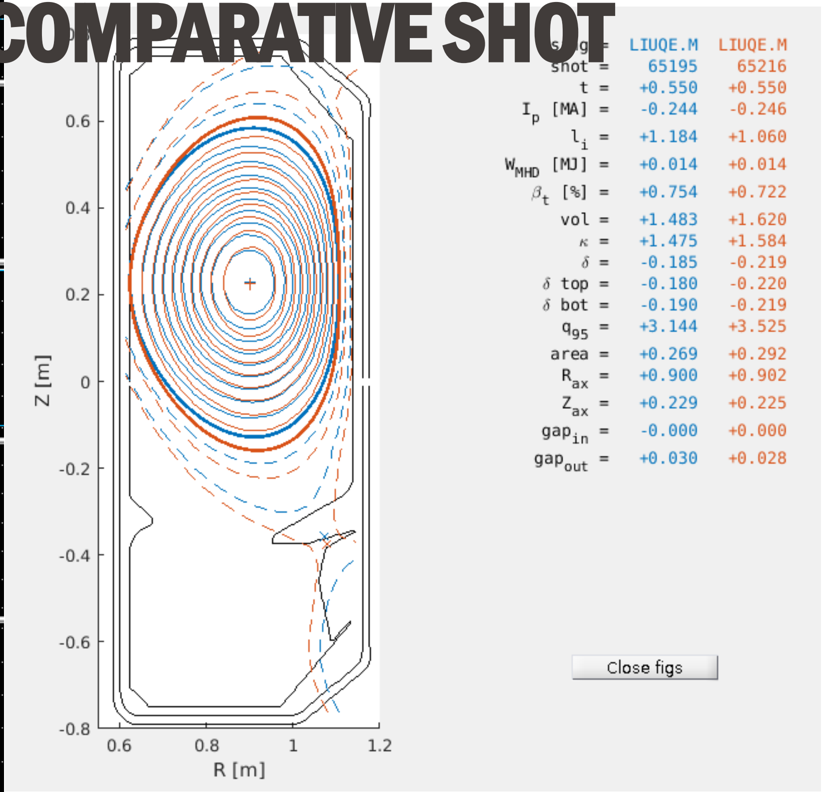
Node 08:
CPU1: ECE crosscorrelation
NTM analysis and
NTM tracking

Node 03:
CPU1: RTLIUQE for shape control and TORBEAM
CPU2: TORBEAM instance 1
CPU3: TORBEAM instance 2
CPU4: TORBEAM instance 3

14 CPUs on 6 computers to handle

First shot controlling TCV (65195 vs. 65216), REMOVE IN FAVOR OF SOMETHING MORE EDUCATIVE THAN A COMPARATIVE SHOT





Galperti Cristian

Swiss Plasma Center