Real-time framework for ITER control systems

W. Lee[1], A. Zagar[1], B. Bauvir[1], T. Tak[1], A. Winter[2], S. Lee[3], M. Knap[4], P. Karlovsek[4], P. Perek[5], D. Makowski[5]

[1] ITER Organization, St. Paul Lez Durance Cedex, France.

[2] Max Planck Institut für Plasmaphysik, Greifswald, Germany

[3] Korea Institute of Fusion Energy, Daejeon, Republic of Korea

[4] Cosylab d.d., Ljubljana, Slovenia

[5] Lodz University of Technology, Lodz, Poland

woongryol.lee@iter.org

Outline

- ❑ Introduction RTF (what it is and how it works)
- ❑ Evaluation of RTF from a use case study
- ❑ Conclusion

❖ *The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.*

Introduction

- **EPICS is an application framework** to build the applications for controllers and servers (records databases and state machines)
- **Real-Time Framework** (RTF) is a middleware providing common services and capabilities to build real-time control applications in ITER, such as the Plasma Control System (PCS) and plasma diagnostics.
- The **Plasma Control System** is a dominant factor for the ITER pulsed operation. It controls all aspects of the plasma discharge from powering the superconducting magnets up to the plasma termination.
- **Diagnostics** adopted by plasma physicists for measuring plasma properties requires a lengthy process due to complex algorithms.

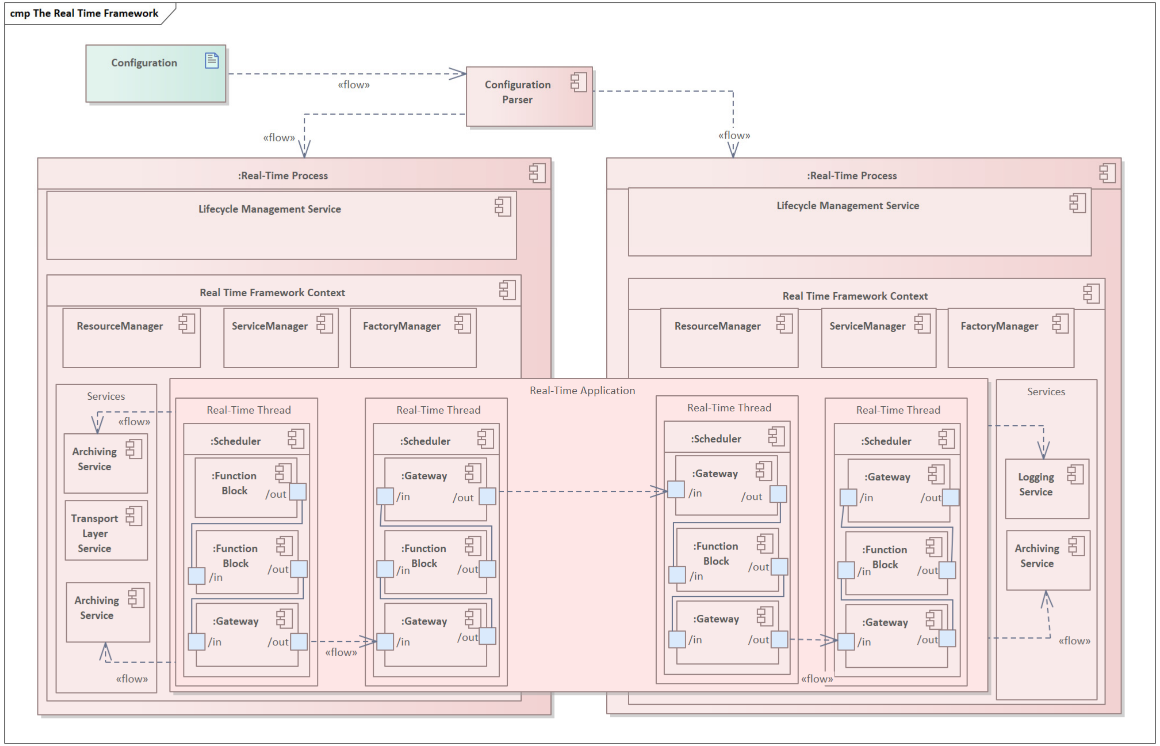
Introduction

- Approximately 190 control functions must be harmonized and coordinated as a whole, and the control scheme can be different for each pulse depending on the goal in a relatively short time interval.
- Thus a flexible high-performance software suite was needed to facilitate the development and deployment of complex real-time applications. Initially aimed to the control algorithms, the RTF can also be the basis for real-time data processing applications in diagnostics.
- A rigorous software quality assurance process compliant with CODAC Software Integrity Level 1[1] reinforced its reliability for building mission-critical systems.
- The architecture design fully considered the modularity and portability of the software, and is applicable and extendable even in none-ITER environments.

[1] SEQA-45 - Software Engineering and Quality Assurance for CODAC (2NRS2K)

Overview of Real-Time Framework

- RTF context diagram [1]



[1] Software Architecture and Design Document for the ITER real-time framework (PKT5S7)

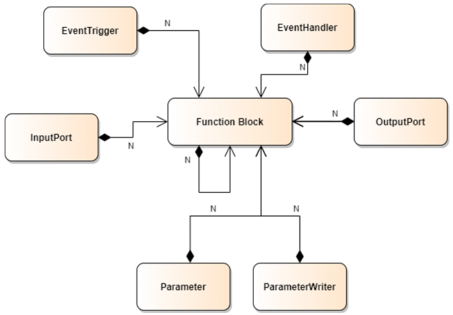
- **Managers** for resource, service and factories.
- **Services** provide site-specific facilities orthogonal to function blocks. Services include mechanisms for transferring data between nodes, logging, archiving, monitoring and control.
- **Real-time threads** that execute all processable objects
- **Function blocks** are chained by connecting output ports to input ports of other function blocks.

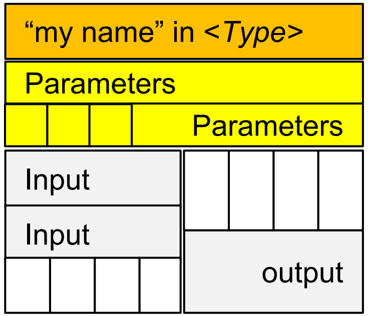
Function Block

- An atomic component to build an application.
- Influenced by parameters, event triggers or event handlers, each function block accepts ***inputs*** and produces ***outputs*** whenever it is ***processed.***
- ***Factory design pattern*** for configuration-driven instantiation.
- ***Encapsulation*** of other function blocks giving the RT application an apparent hierarchical structure.

```
<FunctionBlock Name="evtGen" Type="EventGenerator<int32>">
      <InputPort Name="In" Signal="LHmode:out"/>
      <EventTrigger Name="GenerateEvent" Id="event::LHtransition"/>
</FunctionBlock>
```

```
<FunctionBlock Name="Hpid" Type="kPID<int32>" >

   <Parameter Name="Kp" Value="1.0"/>
   <Parameter Name="Ki" Value="0.0"/>
   <Parameter Name="Kd" Value="0.0"/>
   <Parameter Name="Dt" Value="0.001"/>
   <Parameter Name="Uh" Value="10.0"/>
   <Parameter Name="Ul" Value="0"/>
   <InputPort Name="Setpoint" Signal="ref_H:out"/>
   <InputPort Name="Feedback" Signal="densityH:out"/>
   <InputPort Name="errorThH" Signal="errThrd_H:out"/>
   <OutputPort Name="Out" Signal="Hpid:cmd"/>
   <OutputPort Name="Out" Signal="Hpid:error"/>
   <OutputPort Name="Out" Signal="Hpid:Pval"/>
   <OutputPort Name="Out" Signal="Hpid:Ival"/>
   <OutputPort Name="Out" Signal="Hpid:Dval"/>

   <!--   Connects reset handler to a given event id-->
   <EventHandler Name="ResetHandler<T>" Id="event::LHtransition"/>

</FunctionBlock>
```

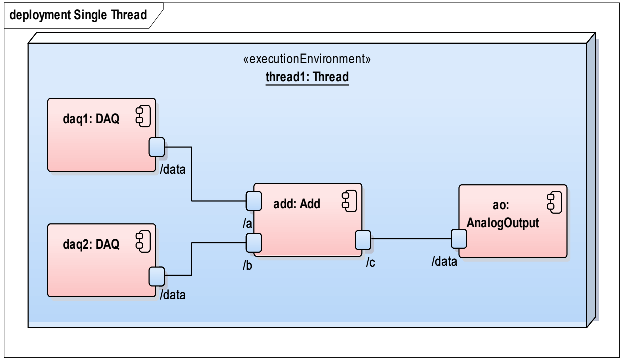




Supports various types of signals e.g. scalar, none-scalar, nested data inside framework.

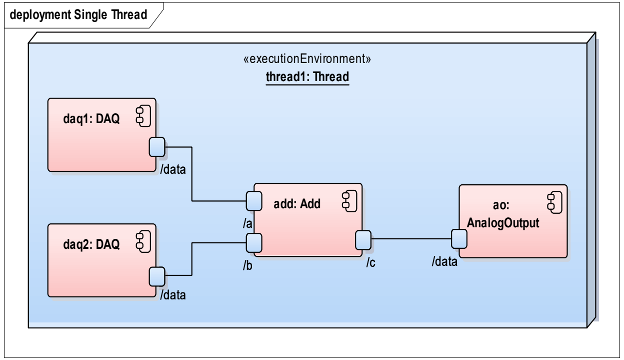
Real-Time application

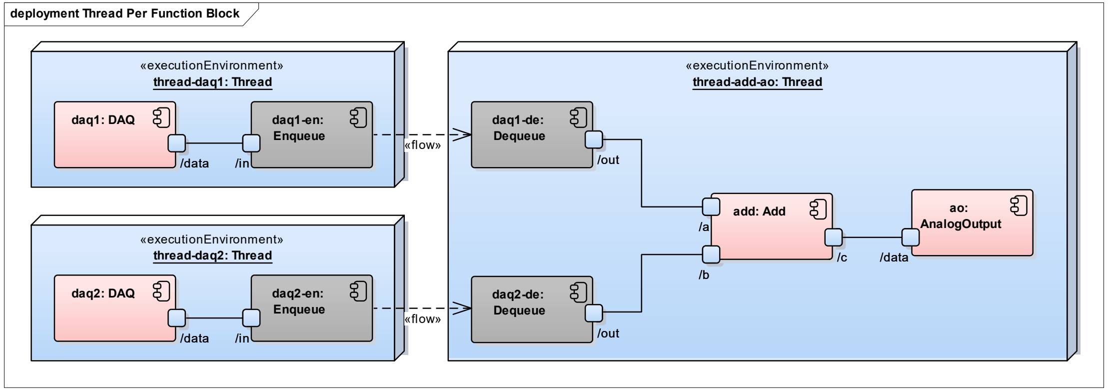
- The RT application defines the processing logic that executes the desired behaviour according to the designer's intention.
- The framework handles the dependency-based execution of FBs in either single-threaded or multi-threaded environments as specified in the deployment configuration.
- Implicit insertion inserts necessary gateway function blocks to implement inter-thread or inter-process (or node) communication



Real-Time application

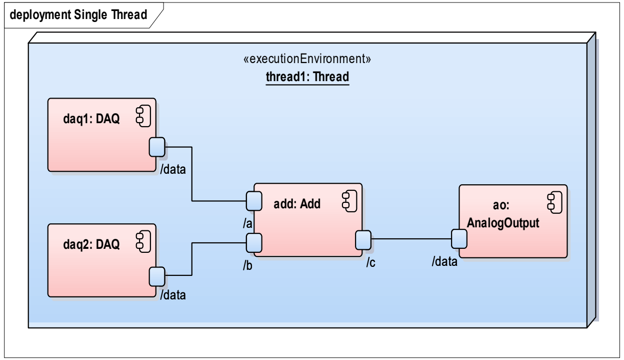
- The RT application defines the processing logic that executes the desired behaviour according to the designer’s intention.
- The framework handles the dependency-based execution of FBs in either single-threaded or multi-threaded environments as specified in the deployment configuration.
- Implicit insertion inserts necessary gateway function blocks to implement inter-thread or inter-process (or node) communication

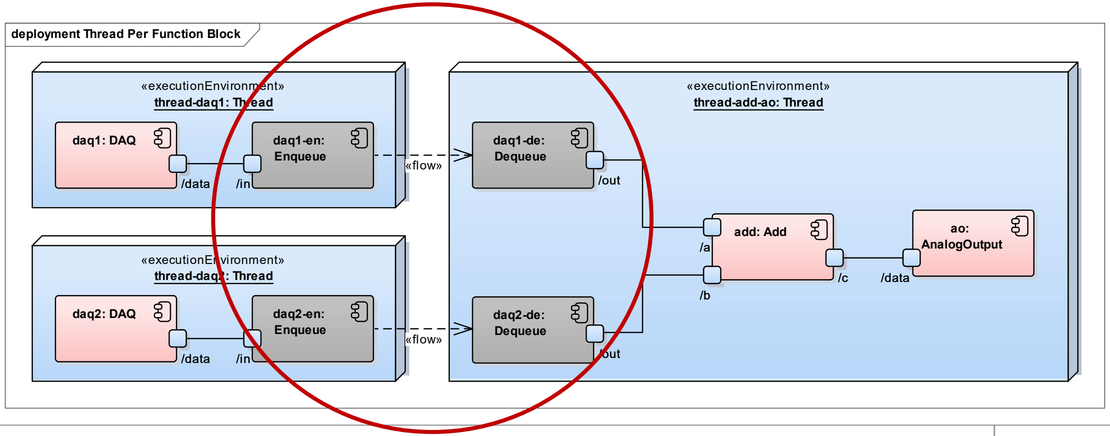




Real-Time application

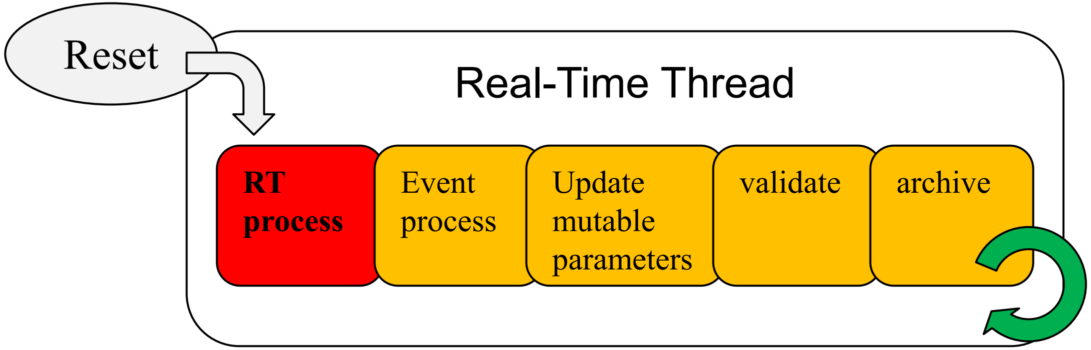
- The RT application defines the processing logic that executes the desired behaviour according to the designer's intention.
- The framework handles the dependency-based execution of FBs in either single-threaded or multi-threaded environments as specified in the deployment configuration.
- Implicit insertion inserts necessary gateway function blocks to implement inter-thread or inter-process (or node) communication





Processing of FB in a thread

- FBs are instantiated and serialized in the loading phase. The ordering is determined by the relation between function blocks once the configuration is parsed.
- Executing the FBs is implemented with busy-wait rather than through interrupts or callbacks to avoid context switching; they effectively minimize jitter and response times.



- **RT process**: handles processing of the logic. All the inputs/outputs get updated in this stage.
- **Process events**: handles processing of all the events that have been triggered.
- **Process parameters**: updates the mutable parameters.
- **Validate**: validates the execution times and/or parameters to be validated.
- **Archive**: archives signals and all archivable objects.

❖ *The RT process should be constant over all the cycles. The underneath rule of execution of FB is to execute the process method periodically under rt-thread, and thus ensure predictability of the execution times.*

Framework Life Cycle Management

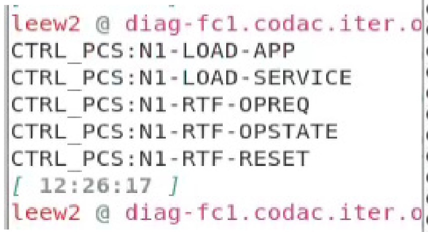
- Since the framework is part of a distributed control system, it needed to centrally control multiple instances in an organized manner
- Life Cycle Management Service (LCMS) allows centrally orchestrate the state transition in conjunction with loading configuration.
- Provision of external interface can be customized to site-specific requirements. Two basic interfaces were supported: ***pvAccess*** protocol from EPICS v7 and native ***TCP/IP***.

```
leew2 @ diag-fc1.codac.iter.org : ~ $ pvlist localhost
CTRL_PCS:N1-LOAD-APP
CTRL_PCS:N1-LOAD-SERVICE
CTRL_PCS:N1-RTF-OPREQ
CTRL_PCS:N1-RTF-OPSTATE
CTRL_PCS:N1-RTF-RESET
[ 12:26:17 ]
leew2 @ diag-fc1.codac.iter.org : ~ $
```

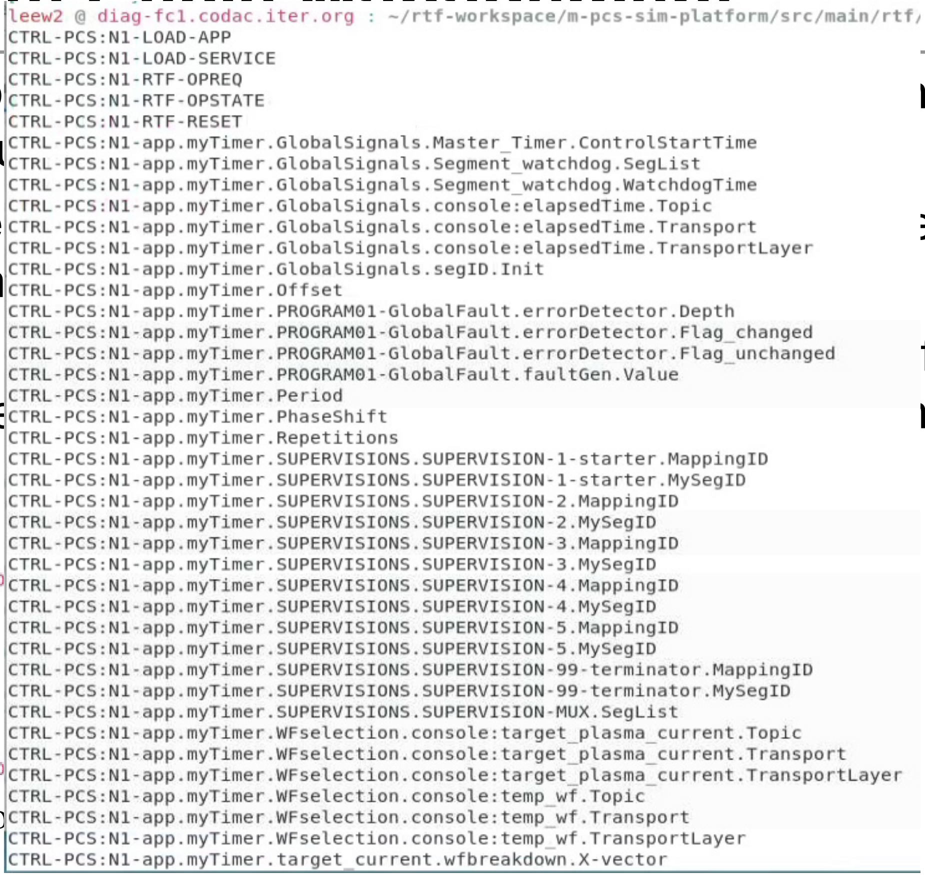
Operational PVs for interworking with central supervision system

Framework Life Cycle Management

- Since the framewo… …eeded to centrally control mu…
- Life Cycle Manage… …estrate the state transition in conjun…
- Provision of extern… …fic requirements. Two basic interface… … EPICS v7 and native ***TCP/IP***.



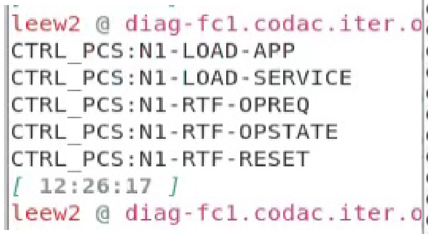
Operational PVs for interwo…



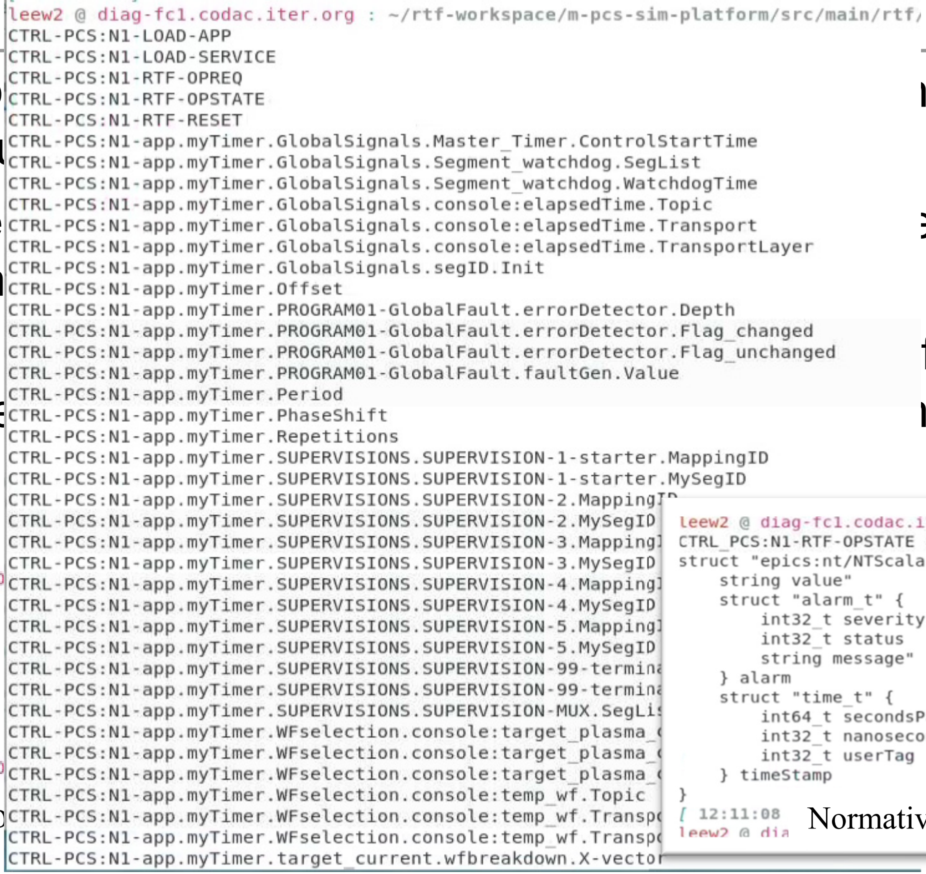
PVs are dynamically created after receive configuration.

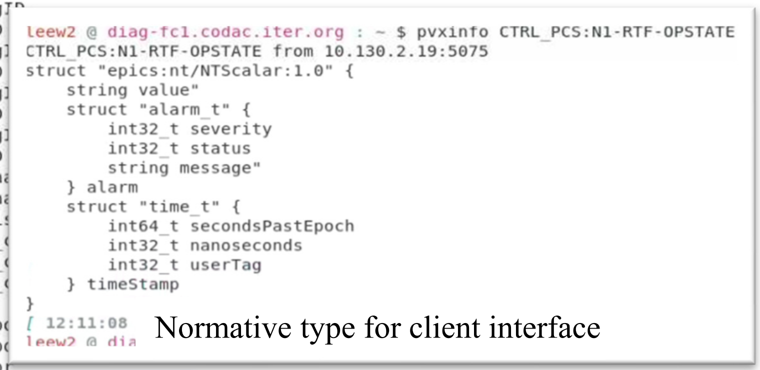
Framework Li

- Since the framewo ... ieeded to centrally control mu
- Life Cycle Manage ... estrate the state transition in conjun
- Provision of extern ... fic requirements. Two basic interface ... EPICS v7 and native ***TCP/IP***.



Operational PVs for interwo





Normative type for client interface

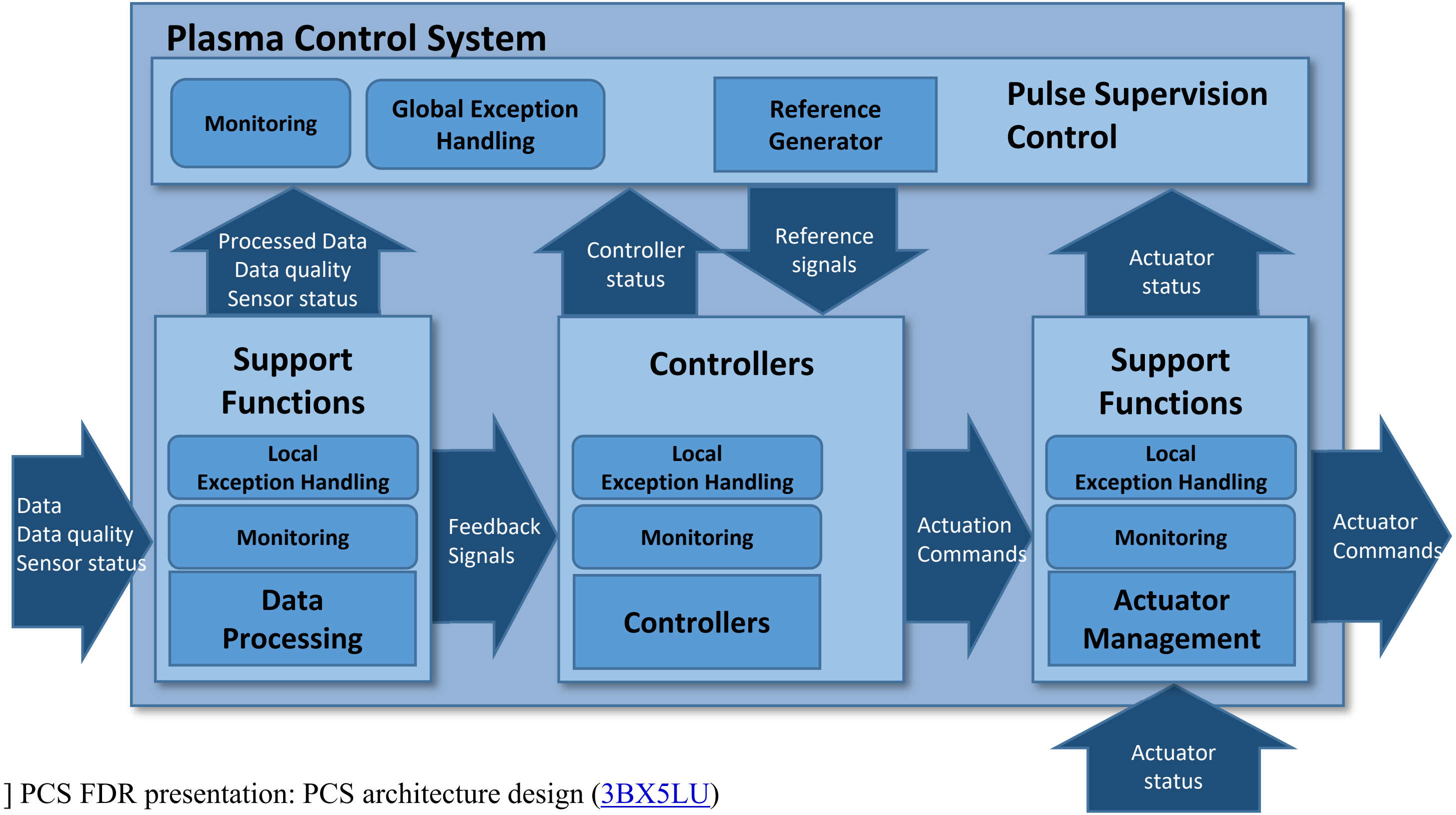
PVs are dynamically created after receive configuration.

Interface support

- Foundational networking infrastructure is implemented as a ***transport layer service,*** and supports communication on all levels of RTF
  - Synchronous Data-bus Network (SDN) for feedback control
  - Data Archiving Network (DAN) for experimental data archiving
  - Plant Operation Network (PON) in EPICS pvAccess protocol
  - Nominal Device Support (NDS) for physical hardware interfacing is under development
  - Any necessary in the future
- Simulink interface uses generated code from Simulink Coder™
  - The wrapper FB loads the compiled library, and RTF performs a validation process to verify the interface data structure.

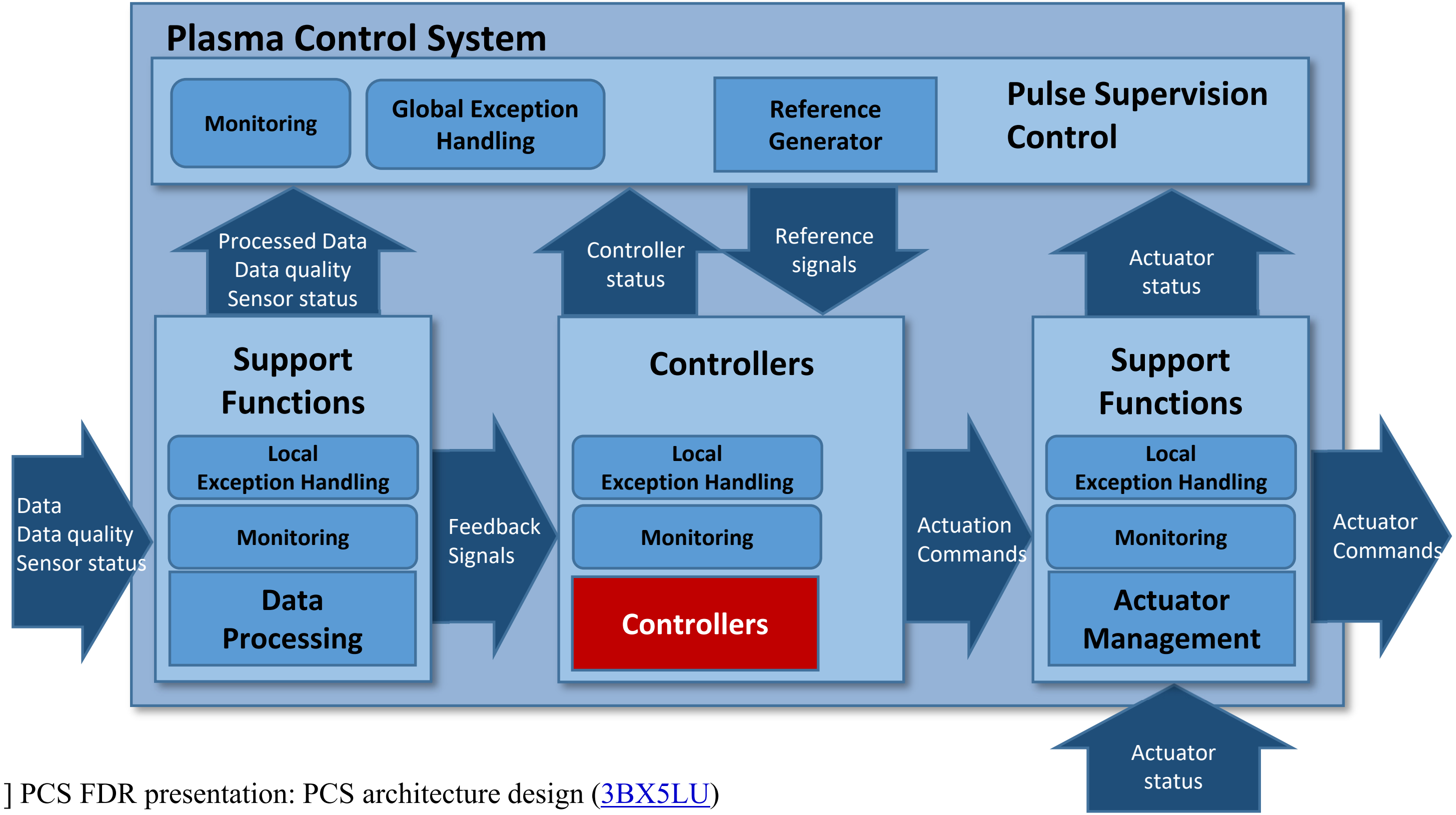
Introduction of a use case study

PCS Prototyping : Basic sketch of the PCS architecture



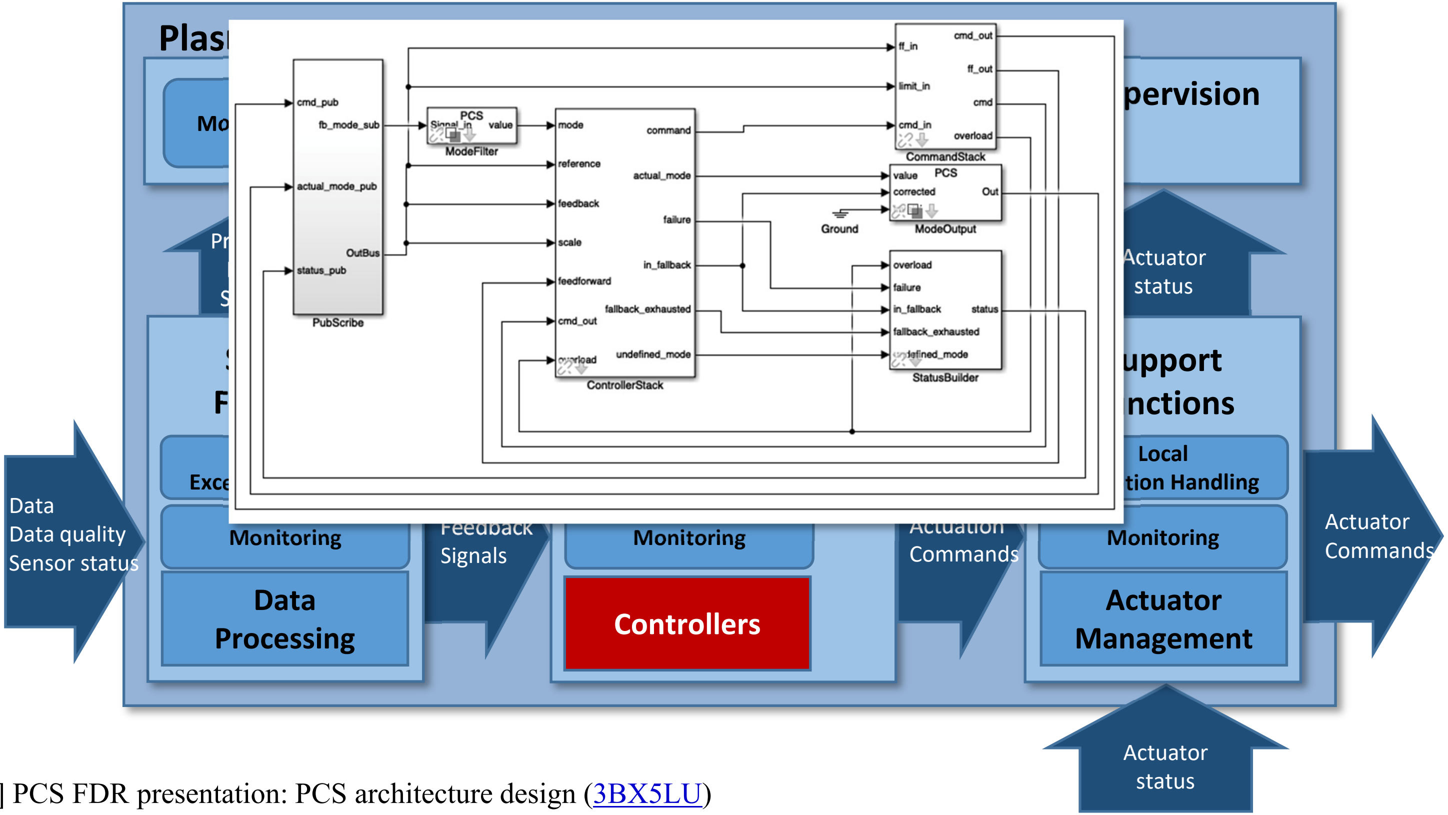
[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Basic sketch of the PCS architecture



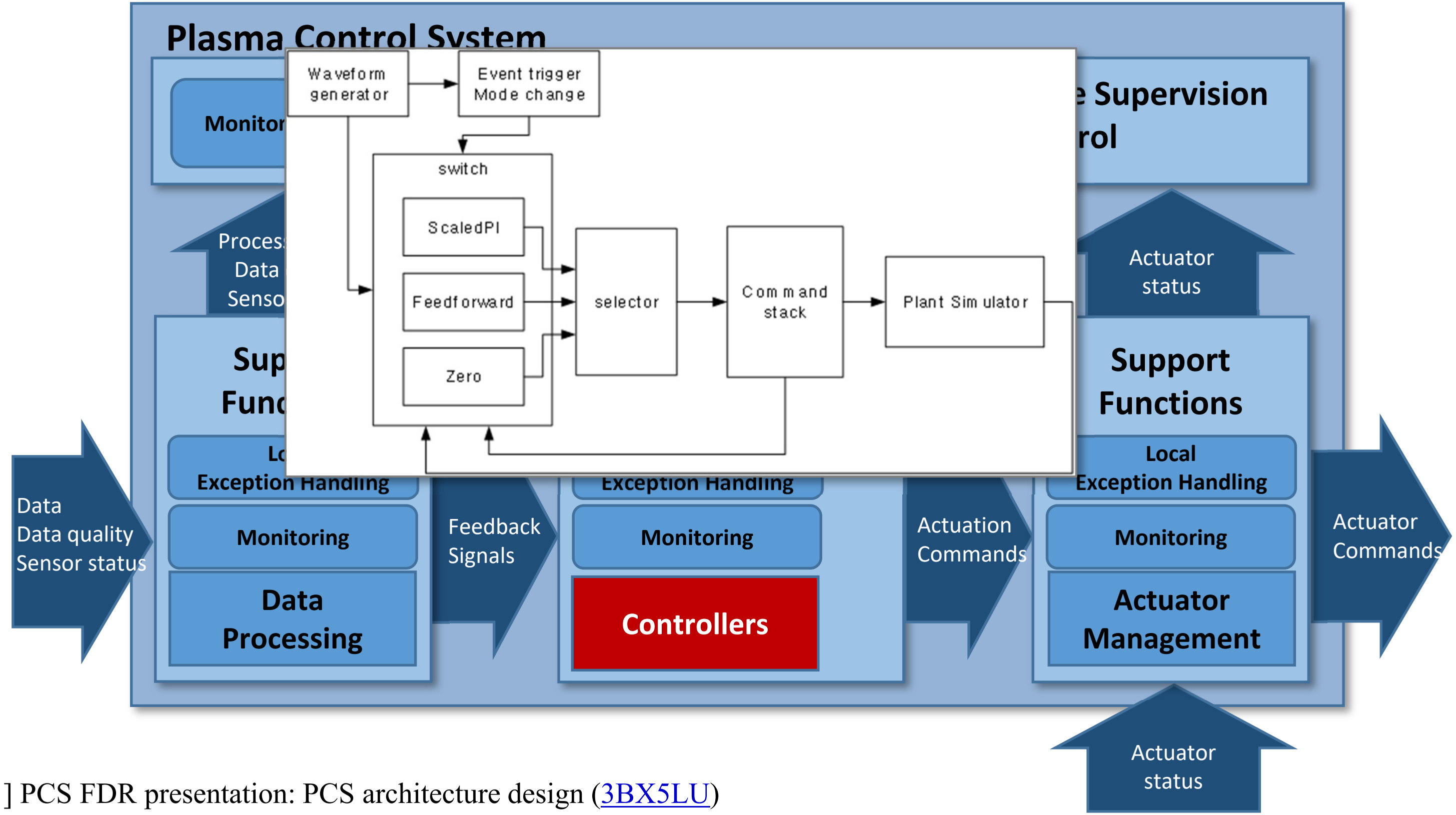
[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Basic sketch of the PCS architecture



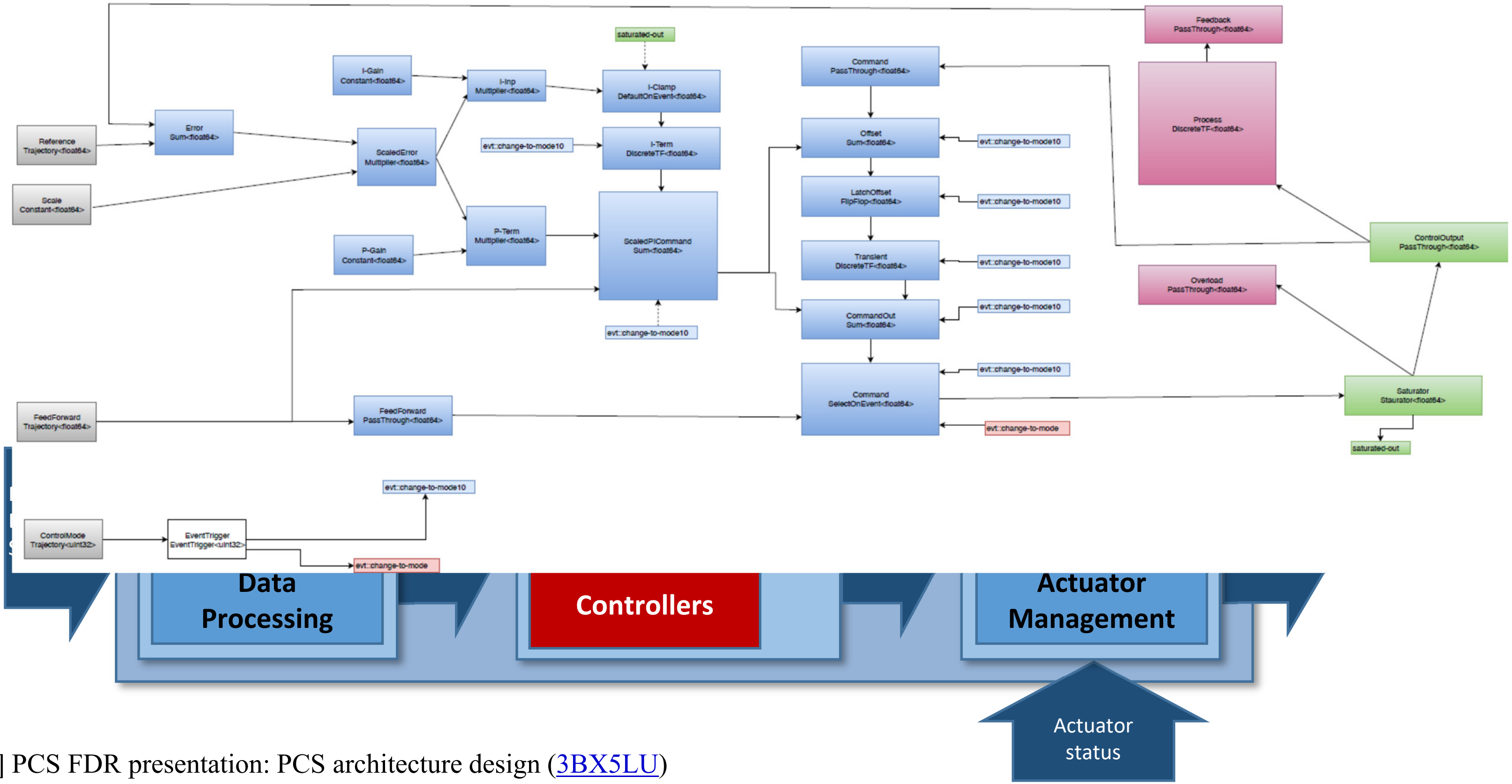
[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Basic sketch of the PCS architecture



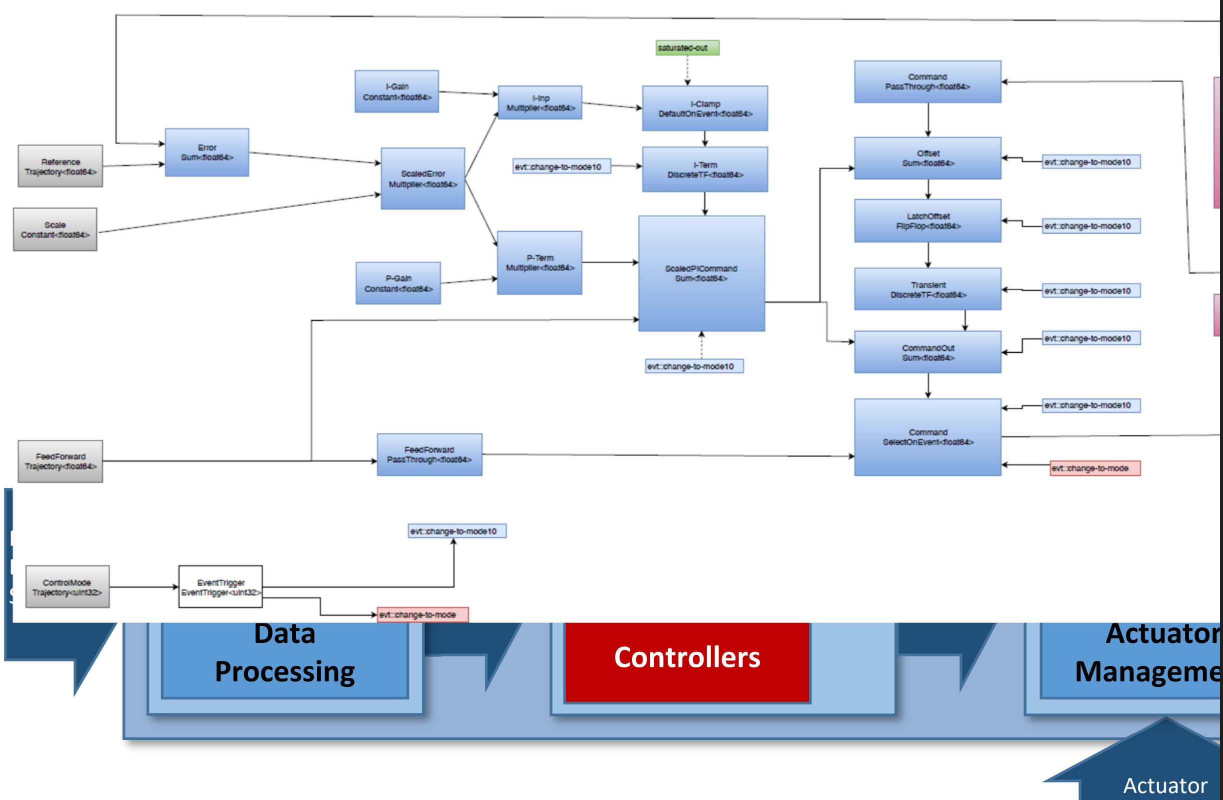
[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Basic sketch of the PCS architecture



[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Basic sketch of the PCS architecture

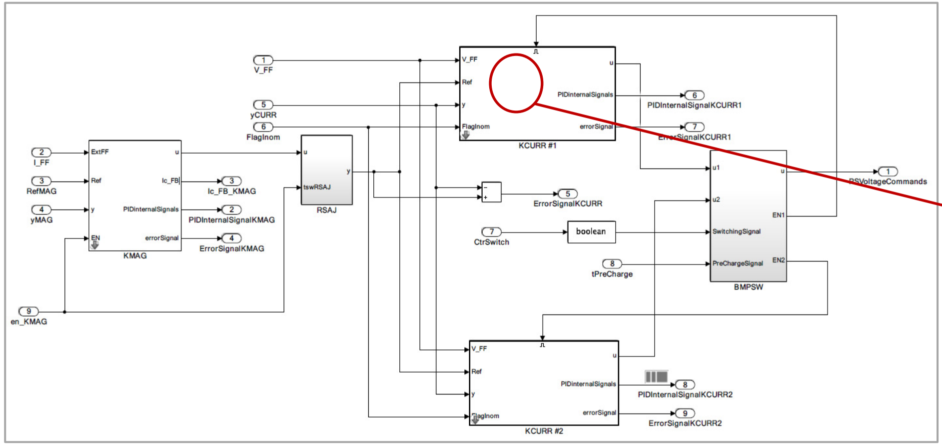



```xml
<FunctionBlock Name="ControlStack" xmlns:xsi="http://www.w3.org/2001/XMl
    <!-- Feedforward -->
    <FunctionBlock Name="Feedforward" Type="PassThrough<float64>">
        <InputPort Name="In" Signal="sig::Feedforward"/>
        <OutputPort Name="Out" Signal="local::FeedforwardCommand"/>
    </FunctionBlock>
    <!-- ScaledPI -->
    <FunctionBlock Name="ScaledPI">
        <FunctionBlock Name="Error" Type="Sum<float64>">
            <Parameter Name="Signs" Value="+-"/>
            <InputPort Name="In" Signal="sig::Reference"/>
            <InputPort Name="In" Signal="sig::Feedback"/>
            <OutputPort Name="Out" Signal="local::Error"/>
        </FunctionBlock>
        <FunctionBlock Name="ScaledError" Type="Multiplier<float64>">
            <InputPort Name="In" Signal="local::Error"/>
            <InputPort Name="In" Signal="sig::Scale"/>
            <OutputPort Name="Out" Signal="local::ScaledError"/>
        </FunctionBlock>
        <FunctionBlock Name="P-Gain" Type="Constant<float64>">
            <Parameter Name="Value" Value="10"/>
            <OutputPort Name="Out" Signal="local::P-Gain"/>
        </FunctionBlock>
        <FunctionBlock Name="P-Term" Type="Multiplier<float64>">
            <InputPort Name="In" Signal="local::ScaledError"/>
            <InputPort Name="In" Signal="local::P-Gain"/>
            <OutputPort Name="Out" Signal="local::P-Term"/>
        </FunctionBlock>
        <FunctionBlock Name="I-Gain" Type="Constant<float64>">
            <Parameter Name="Value" Value="1"/>
            <OutputPort Name="Out" Signal="local::I-Gain"/>
        </FunctionBlock>
        <FunctionBlock Name="I-Inp" Type="Multiplier<float64>">
            <InputPort Name="In" Signal="local::ScaledError"/>
            <InputPort Name="In" Signal="local::I-Gain"/>
            <OutputPort Name="Out" Signal="local::I-Inp"/>
        </FunctionBlock>
        <FunctionBlock Name="I-Clamp" Type="DefaultOnEvent<float64>">
            <Parameter Name="Mapping">
                <Item>false</Item>
            </Parameter>
            <InputPort Name="In" Signal="local::I-Inp"/>
```

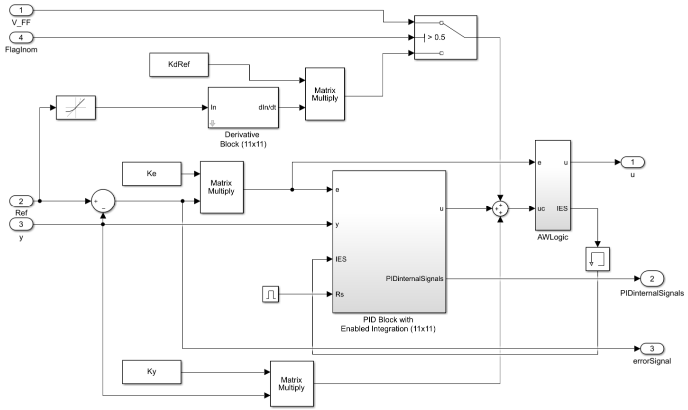
[1] PCS FDR presentation: PCS architecture design (3BX5LU)

PCS Prototyping : Controller model from the generated code

- Controller model from the Simulink platform can be converted to the code under Simulink constraints.
- A designer can devise a desired function only by changing parameters, while maintaining the same external interface to the other FB.
- Need an appropriate granularity in the controller model for conversion.



Unified Magnetic Controller (UMC) from PCS Simulation Platform



Coil Current Controller (KCURR)

PCS Prototyping : Controller model from the generated code

❑ How to interface in between Simulink and RTF

```c
/* External inputs (root inport signals with default storage) */
typedef struct {
  real_T V_FF[11];                    /* '<Root>/V_FF' */
  real_T I_CSPF_ref[11];              /* '<Root>/I_CSPF_ref' */
  real_T I_CSPF[11];                  /* '<Root>/I_CSPF' */
  int8_T FlagInom;                    /* '<Root>/FlagInom' */
  int8_T enable;                      /* '<Root>/enable' */
} ExtU_KCURRsim_T;

/* External outputs (root outports fed by signals with default st
typedef struct {
  real_T u[11];                       /* '<Root>/u' */
  real_T PIDinternalSignals[33];      /* '<Root>/PIDinternalSign
  real_T errorSignal[11];             /* '<Root>/errorSignal

/* Parameters (default storage) */
struct P_KCURRsim_T_ {
  real_T D[11];                       /* Variable: D
                                       * Referenced by:
                                       *   '<S20>/Constant2'
                                       *   '<S21>/Constant2'
                                       *   '<S22>/Constant2'
                                       *   '<S23>/Constant2'
                                       *   '<S24>/Constant2'
```

Example for converting to RTF configuration

```xml
<FunctionBlock Name="KCURR" Type="SimulinkBlock">

  <Parameter Name="LibraryPath" Value="KCURRsim-3/build/libKCURRsim.so"/>

  <Parameter Name="BlockName" Value="KCURRsim"/>

  <Parameter Name="KdRef" Value="[[0.759,0.2246,0.0636,0.0072,0.0037,0.1365
  <Parameter Name="Ke" Value="[[7.59,2.246,0.636,0.072,0.037,1.365,0.431,0.
  <Parameter Name="Ky" Value="[[0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0
  <Parameter Name="N" Value="[100,100,100,100,100,100,100,100,100,100,100]"
  <Parameter Name="Nd" Value="100"/>
  <Parameter Name="P" Value="[1,1,1,1,1,1,1,1,1,1,1]"/>
  <Parameter Name="D" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="I" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="Vlow" Value="[-1050,-1050,-2100,-1050,-1050,-1050,-3150,
  <Parameter Name="Vup" Value="[1050,1050,2100,1050,1050,1050,3150,3150,315

  <InputPort Name="V_FF" Signal="loaderBuffer1"/>
  <InputPort Name="I_CSPF_ref" Signal="loaderBuffer2"/>
  <InputPort Name="I_CSPF" Signal="loaderBuffer3"/>
  <InputPort Name="FlagInom" Signal="loaderLoader4"/>
  <InputPort Name="enable" Signal="loaderLoader5"/>
```

- ❖ Header file generated from KCURR model that defines the data structure for the interface to RTF
- ❖ Simulink wrapper FB loads the compiled library.
- ❖ Interface with structured data e.g., matrix and array is supported in the parameter attribute from the release of v2.2.4.

PCS Prototyping : Controller model from the generated code

❑ How to interface in between Simulink and RTF

```c
/* External inputs (root inport signals with default storage) */
typedef struct {
  real_T V_FF[11];                    /* '<Root>/V_FF' */
  real_T I_CSPF_ref[11];              /* '<Root>/I_CSPF_ref' */
  real_T I_CSPF[11];                  /* '<Root>/I_CSPF' */
  int8_T FlagInom;                    /* '<Root>/FlagInom' */
  int8_T enable;                      /* '<Root>/enable' */
} ExtU_KCURRsim_T;

/* External outputs (root outports fed by signals with default st
typedef struct {
  real_T u[11];                       /* '<Root>/u' */
  real_T PIDInternalSignals[33];      /* '<Root>/PIDInternalSi
  real_T errorSignal[11];             /* '<Root>/errorSignal
} ExtY_KCURRsim_T;

/* Parameters (default storage) */
struct P_KCURRsim_T_ {
  real_T D[11];                       /* Variable: D
                                       * Referenced by:
                                       *   '<S20>/Constant2'
                                       *   '<S21>/Constant2'
                                       *   '<S22>/Constant2'
                                       *   '<S23>/Constant2'
                                       *   '<S24>/Constant2'
```

Example for converting to RTF configuration

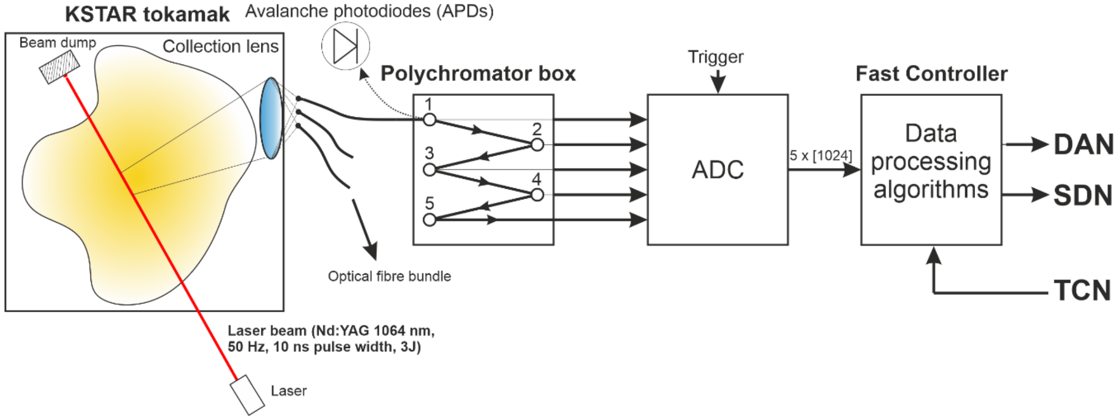
```xml
<FunctionBlock Name="KCURR" Type="SimulinkBlock">

  <Parameter Name="LibraryPath" Value="KCURRsim-3/build/libKCURRsim.so"/>

  <Parameter Name="BlockName" Value="KCURRsim"/>

  <Parameter Name="KdRef" Value="[[0.759,0.2246,0.0636,0.0072,0.0037,0.1365
  <Parameter Name="Ke" Value="[[7.59,2.246,0.636,0.072,0.037,1.365,0.431,0.
  <Parameter Name="Ky" Value="[[0,0,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,0
  <Parameter Name="N" Value="[100,100,100,100,100,100,100,100,100,100,100]"
  <Parameter Name="Nd" Value="100"/>
  <Parameter Name="P" Value="[1,1,1,1,1,1,1,1,1,1,1]"/>
  <Parameter Name="D" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="I" Value="[0,0,0,0,0,0,0,0,0,0,0]"/>
  <Parameter Name="Vlow" Value="[-1050,-1050,-2100,-1050,-1050,-1050,-3150,
  <Parameter Name="Vup" Value="[1050,1050,2100,1050,1050,1050,3150,3150,315

  <InputPort Name="V_FF" Signal="loaderBuffer1"/>
  <InputPort Name="I_CSPF_ref" Signal="loaderBuffer2"/>
  <InputPort Name="I_CSPF" Signal="loaderBuffer3"/>
  <InputPort Name="FlagInom" Signal="loaderLoader4"/>
  <InputPort Name="enable" Signal="loaderLoader5"/>
```

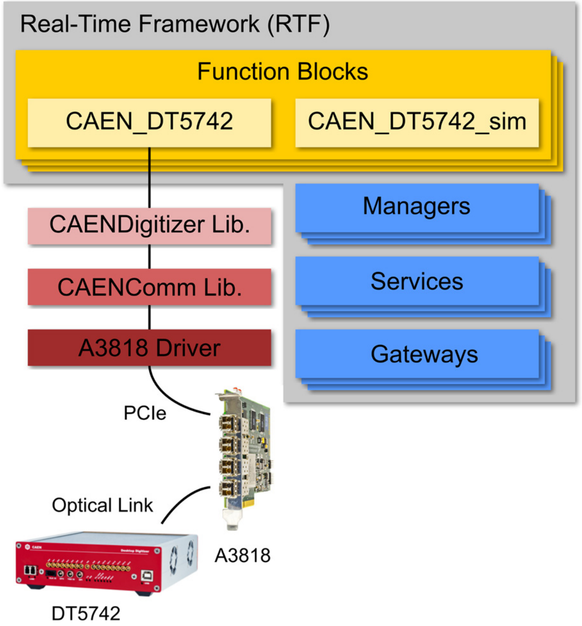
- ❖ Header file generated from KCURR model that defines the data structure for the interface to RTF

- ❖ Simulink wrapper FB loads the compiled library.
- ❖ Interface with structured data e.g., matrix and array is supported in the parameter attribute from the release of v2.2.4.

Plasma diagnostics: Edge Thomson Scattering diagnostics

- The Thomson Scattering diagnostics gives a reliable electron temperature and density profiles in magnetically confined plasma.

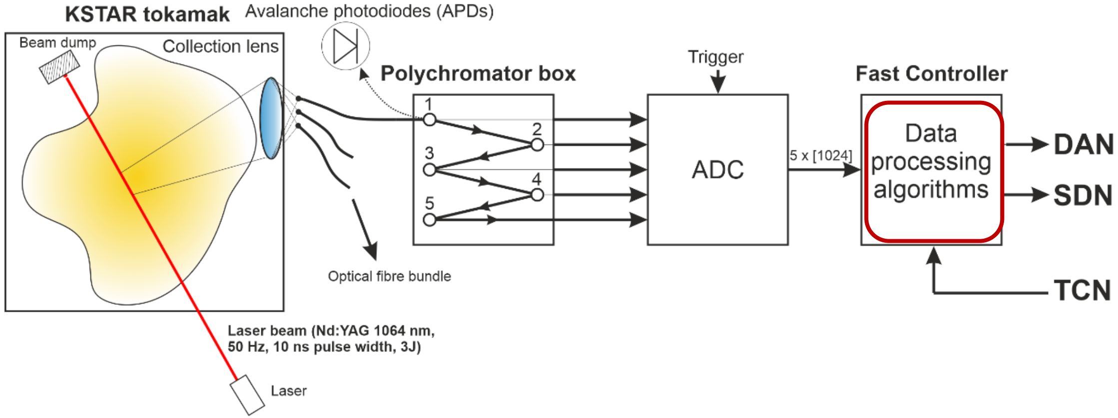


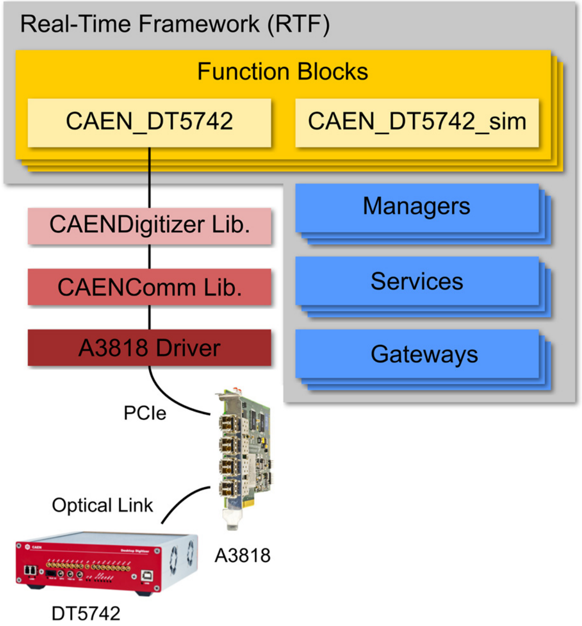


- The customized DAQ FB archives raw data through RTF transport layer
- The output links to the fitting FB to eliminate back scattered signal.
- Electron temperature is measured using lookup table where calibrated data is stored as per the wavelength from the polychromator signal.

Plasma diagnostics: Edge Thomson Scattering diagnostics

- The Thomson Scattering diagnostics gives a reliable electron temperature and density profiles in magnetically confined plasma.

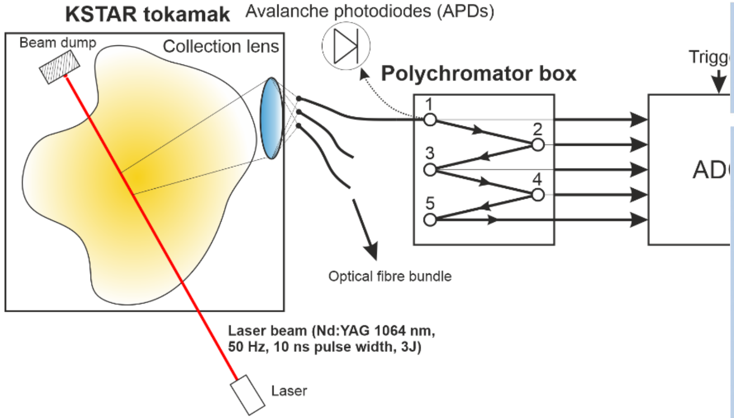


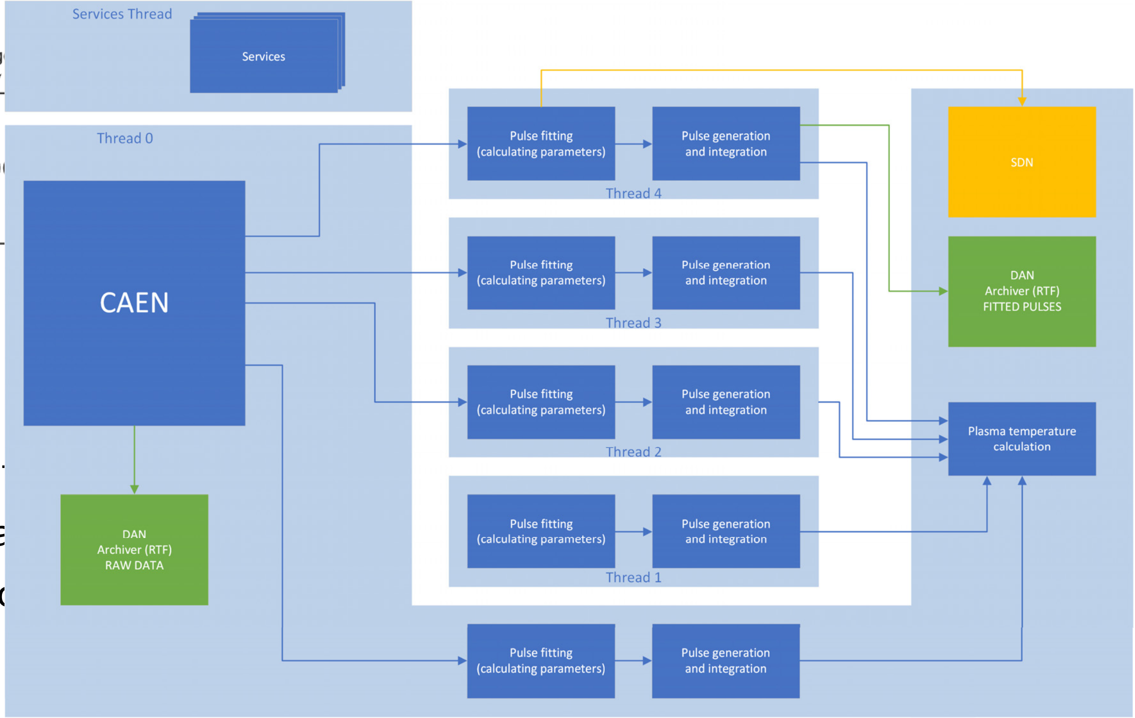


- The customized DAQ FB archives raw data through RTF transport layer
- The output links to the fitting FB to eliminate back scattered signal.
- Electron temperature is measured using lookup table where calibrated data is stored as per the wavelength from the polychromator signal.

Plasma diagnostics: Edge Thomson Scattering diagnostics

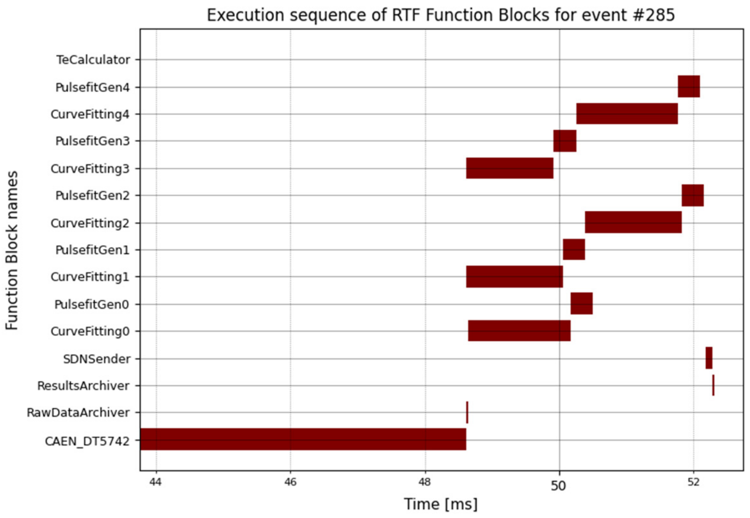
- The Thomson Scattering diagnostics gives a reliable electron temperature and density profiles in magnetically confined plasma.

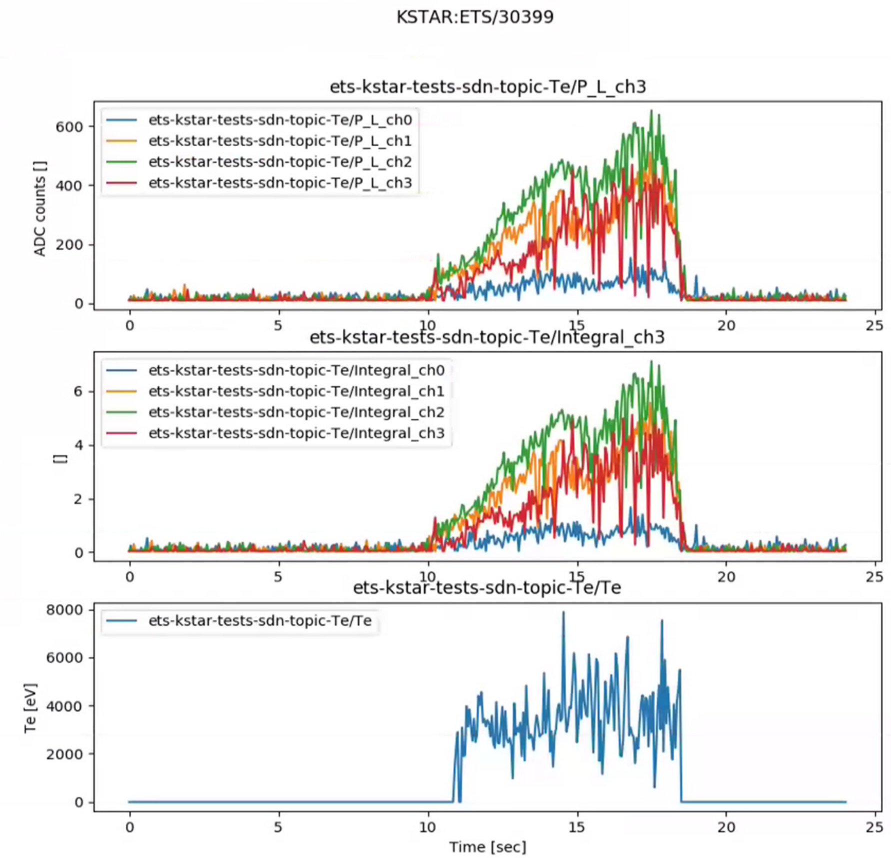




- The customized DAQ FB archives raw da
- The output links to the fitting FB to elimina
- Electron temperature is measured using l data is stored as per the wavelength from

Plasma diagnostics: Edge Thomson Scattering diagnostics

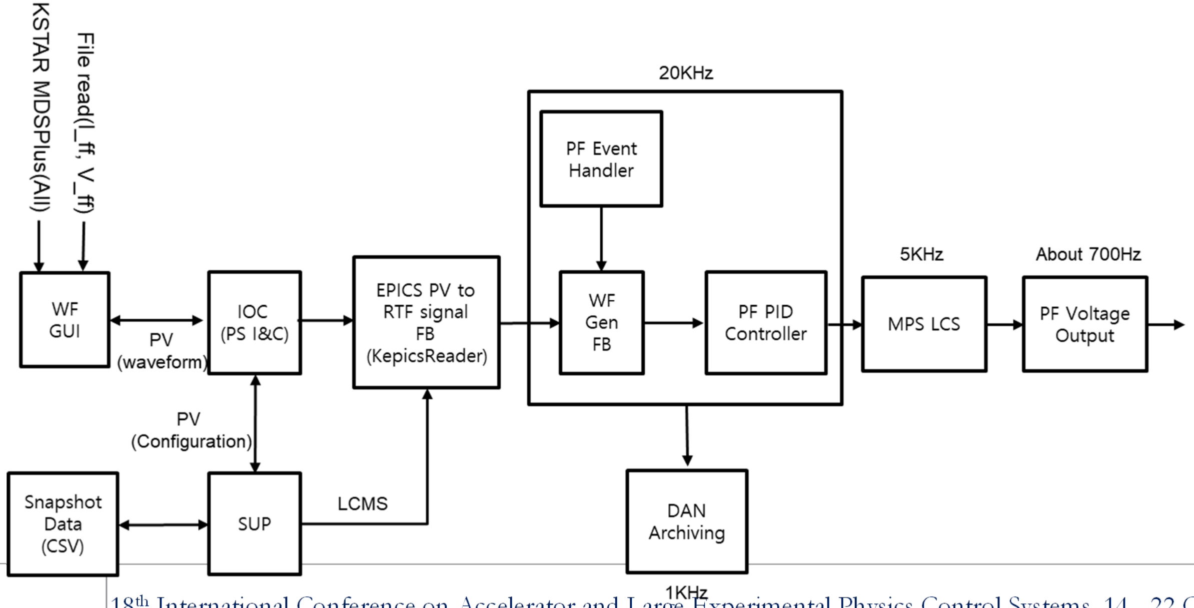




Function block	min	avg	max	Std.
Pulse fitting	0.52 ms	1.31 ms	2.10 ms	0.18 ms
Pulse generation	0.31 ms	0.32 ms	0.54 ms	0.01 ms
T_e calculation	0.01 ms	0.01 ms	0.02 ms	0.00 ms

Actuator control : Poloidal Field coil control

- ITER started implementing a real-life controller in order to evaluate both functional and non-functional behaviour of the PCS.
- 11 PF controllers were devised by complying with KSTAR native function model. Verified 20kHz control cycle in consecutive process pipeline such as exception handler, waveform generator, and PID function.



Conclusions

- The RTF is a flexible high-performance software platform that facilitates the development and deployment of complex real-time applications.
- It was designed to be portable and modular, enabling high reusability and maintainability of components constituting the real-time applications.
- Factory design pattern and rich function for multi-threaded program enables building application through configuration-driven process.
- Prototyping activities on some of the operating Tokamaks have demonstrated its applicability for the implementation of ITER real-time control systems.

Thank you for your attention!