9th International Workshop on Simulation for European Space Programmes
SESP 2006
6-8 November 2006 at ESTEC, Noordwijk, the Netherlands

Simulation of the Smart-1 Electric Propulsion System With a System Simulation Software Ecosimpro®

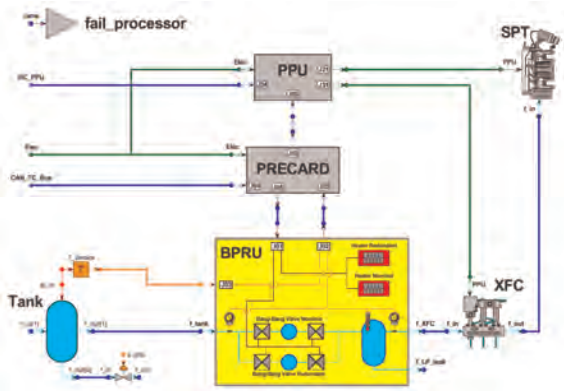
Jesus Alvarez[(1)], Ramon Perez-Varra[(1),] Christophe R. Koppel[(2)],

[(1)] *Empresarios Agrupado,*
Maggalanes 1, Madrid (Spain)
Emails: agu@iberespacio.es, rpv@ecosimpro.com

[(2)] *KopooS Consulting ind.*
57 rue d'Amterdam F-75008 (France)
Email: christophe.koppel@kopoos.com

ABSTRACT

SMART-1 is the first European spacecraft using Electric Primary Propulsion. A detailed simulation model of the Electric Propulsion System has been developed using EcosimPro® , a multi-disciplinary dynamic system simulation tool. The need of a simulation model was recognised early in the project, and the model was developed at the same time that the system was being designed.



The model has evolved with the project and it has been used during the complete life-cycle of the project: design, integration, testing, operations, and it was finally integrated into the real time simulator of the spacecraft at ESOC. Multiple benefits have been obtained from the model, like:

The check of the design changes for improving the behaviour of the system

The performance of fine tuning of the control units

The understanding of the behaviour of the system during normal operation and to found the recovery actions during the training when a failure is simulated

To enable the testing of the actual software of the spacecraft

To support to the operation of the satellite

The simulation model is described in the paper. One shall point out that the architecture of the simulation model is the same of the architecture of the real propulsion system because the simulation tool, EcosimPro®, supports Object Oriented Modelling. That also means that objects developed for this model can be (and have been) reused for other purpose. Comparison of model predictions and test results, and Specific examples of the Model Benefits are provided in the paper. A final part of the paper deals with the "in flight" data of the EPS compared to the EcosimPro® forecasts. It is shown that there are no discrepancies with respect to the simulation of the system. The focus is on validation of the tool and model with (real) data.

INTRODUCTION

IBERESPACIO (IE) –Spain– was in charge of the simulation of the Electric Propulsion System (EPS) developed by Snecma –France– for the ESA SMART-1 spacecraft. The work involved tasks of detailed simulation of different subparts of the EPS, software for the control of the whole subsystem and software to simulate the real TC (telecommands) and TM (telemetries) of the spacecraft.

The EPS is the primary propulsion system of SMART-1 spacecraft [1, 3, 5, 6]. It is composed of the following equipments (Fig.1):

- Thruster (model PPS® 1350),
- Power Processing Unit (PPU),

* Pressure regulation panel,
* Xenon tank, harness and filter unit (FU).

The thruster operates using xenon as a propellant. Xenon atoms are ionized, then accelerated by an electromagnetic field, neutralized by electrons to maintain charge balance and finally ejected, providing the thrust.
Two Xenon Flow Controllers (XFCs) regulate propellant flowrate.
The PPU transforms the electric power supplied by the spacecraft power bus (50 V DC) to generate the electromagnetic field and to control thruster operation.
Xenon feeding at a constant pressure is achieved through the Pressure Regulation Unit. This equipment uses Bang-Bang valves to keep downstream xenon pressure at 2 bars. Xenon is stored at high pressure upstream of Bang-Bang valves, in the Xenon Tank.
An electronic card (Pressure regulation Electronics, PRE) controls Bang-Bang valves actuation, using data from one low pressure transducer.
The Filter Unit (FU) is inserted between the thruster and the PPU to protect the latter from electrostatic discharges or EMI and to reduce plasma discharge current oscillations.
The harness interconnects the thruster with the PPU and the FU, carrying power and signals between them.

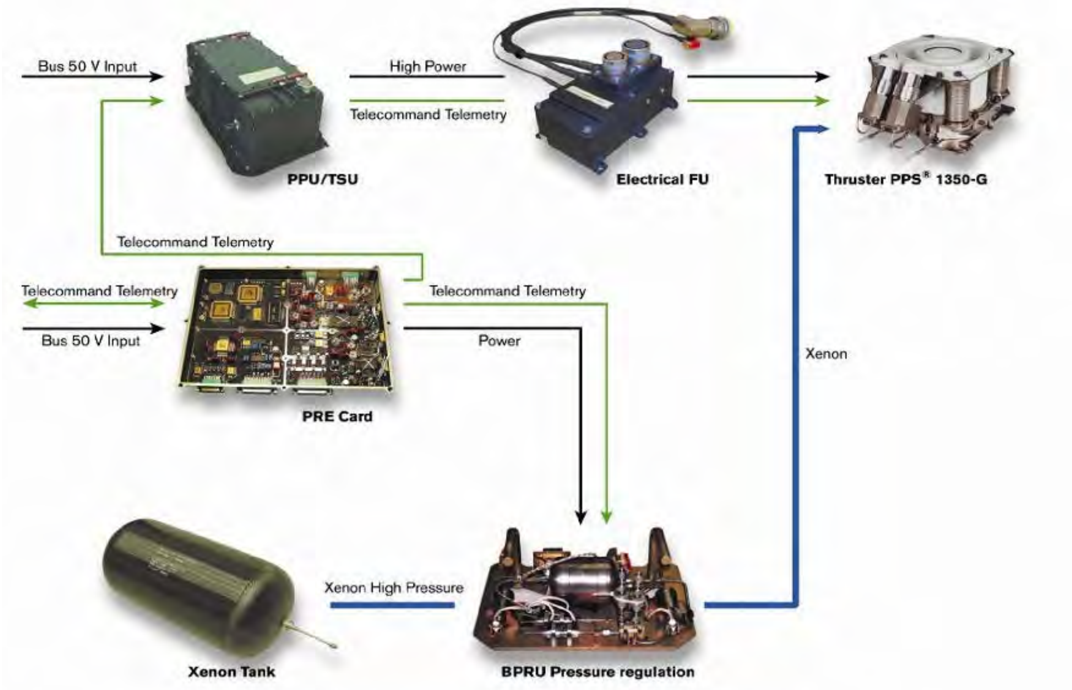


Fig. 1. Main components of the SMART-1 Electric propulsion system
two main inputs for the thruster with the xenon propellant and the electrical power

A description of the main components of the EPS and their model follows.

PPS® 1350 thruster.
The PPS® 1350 thruster is made up of the following major components :


Fig. 2. Thruster PPS® 1350 running in vacuum chamber

* Anode assembly, fig. 2, which includes magnetic winding, magnetic permeable path, discharge chamber with dielectric wall, propellant line voltage isolation, anode and plumbing for propellant distribution within the anode.
* Cathode-Neutraliser assembly, which includes two cathodes, each one having a heater, an emitter and an ignition device (ignitor).
* Two Xenon Flow Controllers (XFCs), fig. 3, each one including three isolation valves, three xenon filters and a mass flow controlled device, called thermothrottle.
Each cathode is associated to a XFC unit. Only one cathode (and its XFC) is used at the same time, while the other cathode remains in standby. Thruster start-up is achieved, after heating the selected cathode, by applying an ignition pulse train. Once in the steady state condition, electrons emitted by the cathode ionize xenon atoms in the discharge chamber. The resulting ions are then accelerated by the electromagnetic field and neutralized by additional electrons coming from the cathode.

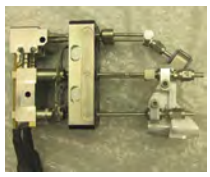


Fig. 3. XFC xenon flow controller of the Thruster PPS® 1350

* The XFC regulates xenon mass flowrate by adjusting current intensity through the thermothrottle. This device contains thermally constricting capillary tubes, so that an increase in current results in a decrease of xenon flow.

The PPS 1350 is qualified for a thrust of 88 mN at 1500 W discharge power. However, for the SMART-1 mission, power level required ranges from 463 W to 1190 W.

The thruster and XFC was described in the model by a set of equations obtained by correlations of test data: mainly the discharge current being a function of the xenon mass flow rate , the magnet current and the discharge voltage. Thrust is given by a similar function. The XFC uses in its model a first degree differential equation for taking into account its time constant.

Power processing Unit (PPU).
The PPU fully controls and supplies the thruster. Its functions include:

* PPS electrical supply (including all supplies and signals needed for anode and cathode assemblies operation).
* XFC electrical supply.
* Telecommand and telemetry interface.
* Power control (tuning of xenon flowrate and power limit).
* Thruster operation sequencing.

The PPU is connected at one side to satellite resources : S/C power supply 50 V DC bus and TM/TC buses. At the other side it is connected to the thruster elements: anode and cathode assemblies (via the filter unit) and the XFCs. The PPU is made up of three main functional units: The Controller (including components needed to TM/TC interface, to sequence thruster operation and to power control), the SPT electrical supply and the XFC electrical supply.

The PPU was described in the model by its logic, including the control loop and the Petri diagrams followed by the PPU when set in Automatic mode or Venting mode. Input and Output places (sub-states) and transition conditions of the Petri net are modelled with discrete statements of the EcosimPro® language: WHEN and IF_ELSE_ENDIF.
The TM/TC were strictly set by reference to the design documents: 31 TC , 48 TM, 4 direct TC and 3 direct TM were integrated in the model.

Pressure regulation.
Pressure regulation panel keeps xenon pressure at XFC inlet at approximately 2 bars (which is adjustable by telecommand), using the bang-bang valves concept. Each of the two redundant bang-bang assembly is composed of 2 normally closed electrical valves in series, with a small volume (~ 0.5 cm 3) in between. A volume of approximately 1 liter, called plenum, is placed downstream of bang.bang valves. This plenum is equipped with two redundant electrical heaters (to avoid xenon condensation). Two pressure transducers, one upstream of bang-bang valves ant the other downstream are used for system control, as well as two temperature probes (at the plenum and at the tank). The Pressure regulation is controlled through the sequential opening of BB valves: The opening of the upstream valve fill the small volume between both valves. Then upstream valve closes, isolating this small volume. After a pause, the downstream valve opens, transferring the limited mass of gas to the plenum. Then downstream valve is closed, returning to the initial state.
This intrinsic concept of Bang-Bang regulator introduce a very regular fluctuation of the “constant regulated pressure”: each time the plenum measured pressure becomes lower than the target pressure, the bang-bang valves are automatically activated and a small positive step in the pressure of the plenum volume occurs. This characteristic, as the heart of the system, is visible on about all the tele-measured functional parameters of the EPS. The telemetries available from that system are the main parameters:

A dedicated model was developed for the xenon pressure regulation, see further chapters.

Pressure Regulation Electronics (PRE).
The PRE card is inserted in the S/C on-board computer. Its main functions are :

* Pressure regulation panel configuration (bang-bang valve and heater selection)
* Bang-bang valves control
* Heater control
* Electrical supply for all pressure regulation components
* TM/TC interface
* Interface between CAN I/F and ML16/DS16 I/F (for the PPU)

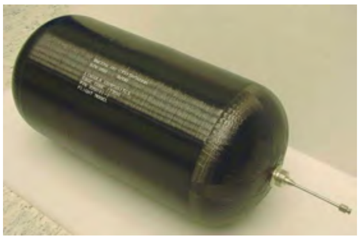


Fig. 4. Xenon Tank

The PRE was described in the model by its logic, including the loop and the Petri diagrams followed by the PRE when set in Automatic mode or Venting mode. The TM/TC were strictly set by reference to the design documents: 36 TC, 42 TM and 1 direct TM were integrated in the model.

Xenon tank

The xenon tank, fig. 4, has a nominal volume of 49.5 litres and a weight of less than 7.73 kg. It is a metal composite, cylindrical tank, with aluminium liner and carbon overwrap. For a BOL (beginning of life) capacity of 82 kg at a MEOP (maximum expected operation pressure) of 150 bars, tank temperature must be maintained below 50º C.

As for the xenon pressure regulation, a dedicated model was developed, see further chapters.

SUMMARY OF USER REQUIREMENTS OF THE "EPS SIMULATOR SOFTWARE"

The main User Requirements extracted from customer reference are listed below:

* The EPS Simulator Software will be performed through the use of EcosimPro® for the development of the Software
* The EPS Simulator Software will be representative of the EPS behaviour, as well as being able to simulate a number of predefined failure cases
* The EPS Simulator Software shall consist of the following parts:
  The performance model built using EcosimPro, which includes the PPU model as well as the xenon regulation and control, in order to provide the complete simulation of the EPS operation with control loops. In addition, the Ecosim model simulates the telcommand inputs and the telemetry outputs as well as the proper power and data handling protocols.
  A C++ wrapper of the EcosimPro model that will implement the interface between "SPEES" and the EPS Simulator Software.
  The EcosimPro model is translated into C++ code and compiled together with the C++ interface to make a dynamic library, the EPOS (EPS Simulator Software) library.
  The "SPEES" core will use RPC (Remote Procedure Call) to communicate with the EcosimPro model. The RPC server is linked with the EPOS library into an executable that is launched on the PC.
* The interface to the CAN (controller area network) controllers shall be handled by the onboard simulation software and delivered through a S/W interface
* The following parameters shall be made available to the S/C Simulator :
  The thrust vector and thrust torque (including the roll torque)
  The Precard power consumption and the PPU power comsuption
  The Xenon mass flow
  The specific impulse
  The discharge power

THE MODEL

A detailed simulation model of the Electric Propulsion System has been developed using EcosimPro® because that software was at that time the unique a multi-disciplinary dynamic system simulation tool. EcosimPro software was also selected for its reliability, its modular approach and its very valuable feature of symbolic treatment of the equations as well as the automatic resolution of the differential equations. The code written by Iberespacio (Spain) is thus understandable by any Engineer having rudimental basis in the language used. Moreover, the acceptance of the software was much facilitated by the great readability and its real time modification feature for adding a differential equation. This is of course also true, if any, for maintenance purpose.

The main advantages of using this specific kind of simulation tool were its ability to manage the systems and its components like objects. This was very powerful for the "low cost approach" of the EPS development. There was no need to develop multiple software for functional purpose and for simulation purpose. Thus lots of efforts and money were spared. In addition, the simulation results are much more representative of the real behaviour.

FIRST CHAPTER: THE FUNCTIONAL MODEL

The pressure regulation is generally not a difficult task when using "normal" gas like helium or nitrogen. But xenon gas is a much more complicate gas because its characteristics are much deviated from the perfect gas law. Its critical temperature is high (16°C), its critical pressure is 5.8 MPa and its critical volume specific mass is very high 1100 kg/m3 (larger than liquid water).

Because that was the first time that Snecma developed a xenon pressure regulation system for a flight satellite, and in order to maintain the low cost goal, it was decided to minimise the development tasks by using the software modelisation with real gas behaviour and with thermal aspects. No specific hardware was built for that development.

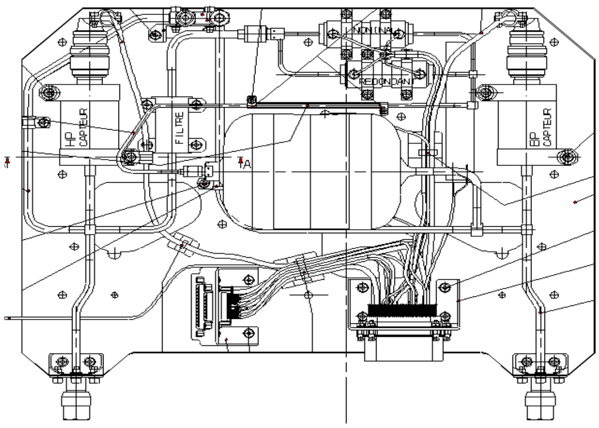


Fig. 5. BPRU (Bang-bang Pressure regulation unit) layout

However, the unique model of the pressure regulation unit (proto-flight model) was made available for being first used for a short characterisation test campaign, setup with real gas xenon stored in small size tanks (1 litre instead of 50 litres). This was undertaken before the acceptance test of the device.

The **main question** that the functional model had to solve was the assessment of the **number of bang-bang cycles** needed to achieve the nominal mission. The valves were nominally constrained by only 1 million of activation (off/on/off cycle).
The **second main goal** for the model was to characterise the digital regulation system in order to be able to **check in real time the health** of the system. This means that a **look-up table for computing the duration of the bang-bang cycle** was requested as output of the model.

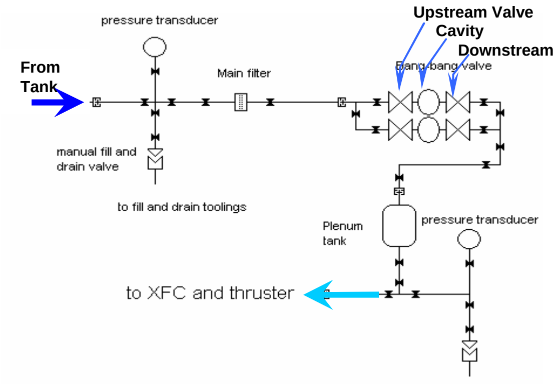


Fig. 6. BPRU sketch of the robust concept

Technical description of the Bang-Bang process

The technical description of the BPRU [2,4,8], fig. 5, follows on the next sketch, fig.6. The xenon is delivered to the BPRU thorough a mechanical connection. A filter provided by Sofrance, Snecma Group, assure the cleanliness of the xenon gas before entering into the Bang-Bang valves provided by Moog, USA. After the Bang-Bang valves, the gas is allowed to fill a plenum tank of one liter.

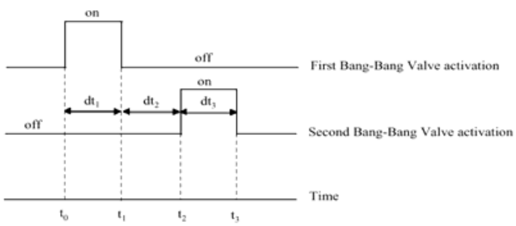


Fig. 7. Bang Bang Valve Cycle Timing

The nominal status of the valves is that there are all closed, even in automatic regulation mode.

When the outlet pressure of the BPRU decrease below the Preset Pressure value, the bang-bang valves are actuated in the following manner, fig.7:

- The upstream valve is opened
- The internal cavity between the two valves of the Bang-Bang assembly begins to be filled by the xenon gas from the tank
- Duration of the upstream opening is T1 seconds.
- This upstream valve is then closed.
- After a while of T2 seconds, the downstream valve is opened.
- The plenum tank pressure start to increase
- After T3 seconds, the downstream valve is then closed.
- The Bang-Bang valves are all closed and a new cycle can be executed on request.

One can see that never in the process, there exists a direct communication between the high pressure tank and the low pressure side of the BPRU: there is at least one valve fully closed between the two parts. Moreover most of the time all the valves are closed between the two parts (i.e. always two valves are closed). This particular feature makes that concept of pressurisation very robust, and not sensitive to any timing discrepancy (which is very important for other concept based on the Pulse Width Modulation (PWM)). Thus the simulation of that system was quite simplified by having chosen such a concept.

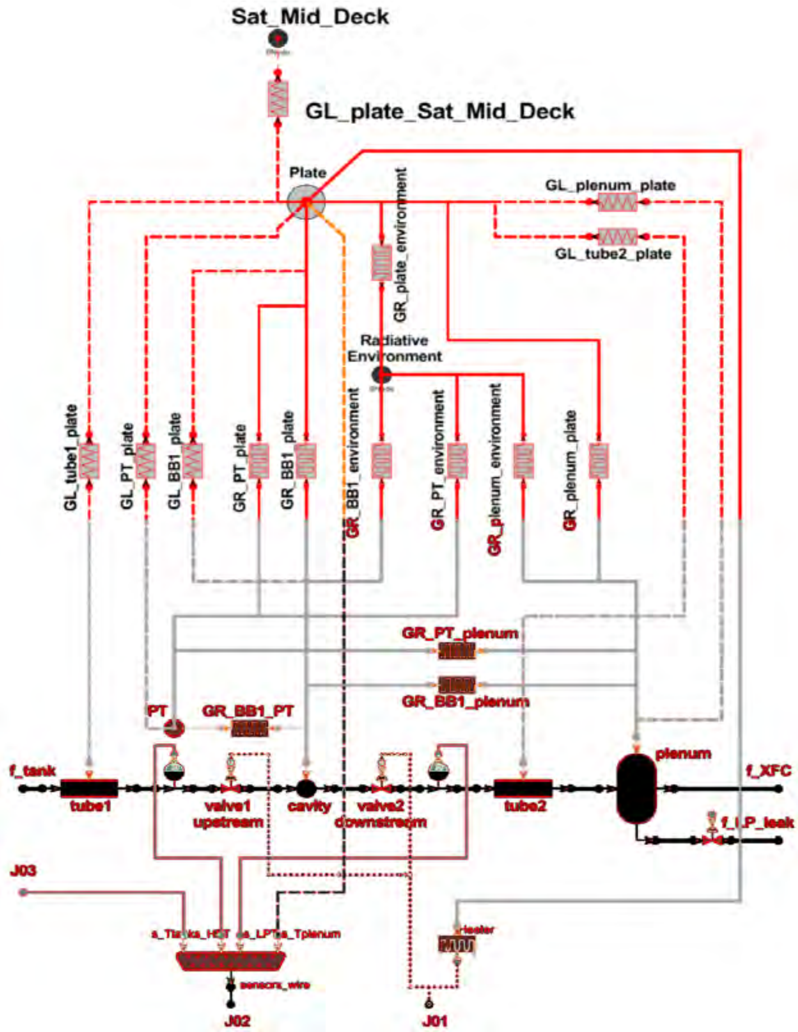


Fig. 8. BPRU Functional modelisation on EcosimPro®

Equations of the Bang-Bang process in the functional model

The model rely on the following :

* Management of the real gas characteristics of the xenon gas: this includes of course the state law, but also enthalpy, entropy, internal energy, viscosity, thermal conductivity, specific heats.
* Dynamic equations of the flow transfer from or into a volume (several differential equations),
* Orifice flow equation (sonic and subsonic),
* Tube with curvature flow equation
* Management of the multiple activation of the electro-fluid-mechanical elements (valves)
* Management of the thermal behaviour (dynamic thermal equation for conduction, forced convection and thermal radiation)

The sketch of the model, directly coming from the software is presented in fig. 8.

The model is divided in two parts: the fluid (gaseous) thermodynamic system based on a real gas xenon characteristic –in light blue color in the figure-- , and the thermal part conductive and irradiative in vacuum and weightlessness – in red color in the figure-. The two parts are properly interconnected together for a global modelisation of the whole pressurization system in its environment. The same model with some formal equations added was able to simulate the functional behavior with the natural convection effects on earth into the cavity and tanks as well as around the external skin of those mechanical parts.

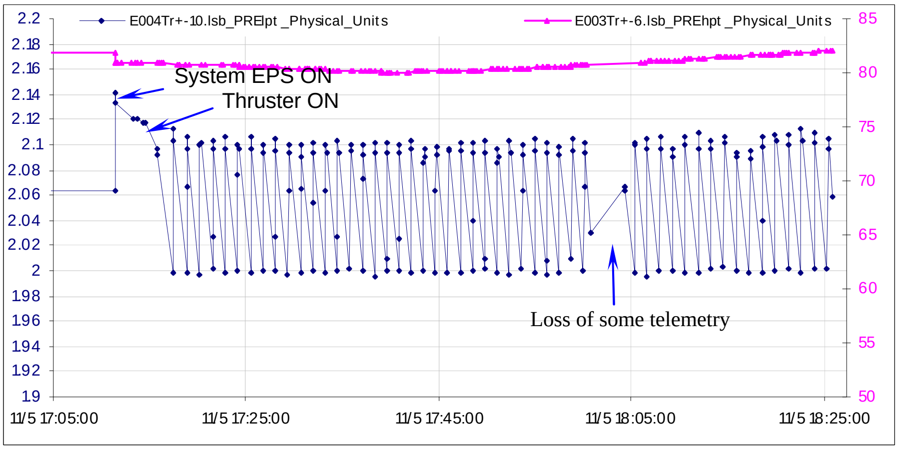


Figure 9. Behaviour of the pressure regulation, in flight, May 2004, duration 80 minutes: Pressure steps between 1.995 bar and 2.11 bar (ie +- 0.058 bar) occurs at each Bang-Bang cycle. *The High pressure around 82 bar shows slow variations.*

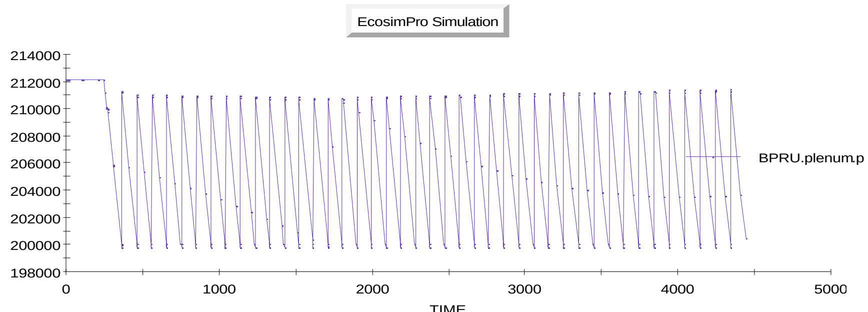


Figure 10. EcosimPro® simulation of the pressure steps with the inputs coming from the flight data: Behaviours are very similar with fig. 10, as expected (units Pa instead of bar).

Bang-bang cycles : Typical behaviours and in-flight measurements

The typical measurements in flight from the BPRU are presented in the figure 9. The regulation of the pressure is performed between 2.11 bar and 1.995 bar (ie +- 0.058 bar). The corresponding simulation output (performed in the same external conditions) presented fig. 10, shows a very similar behaviour.

One shall mention that the slight decrease of the pressure measurement immediately after the system has been powered ON is due to the warming time of the pressure transducer and its electronic. This was not simulated by the model because this was not considered important for the output of the model.

The loss of telemetries visible on fig. 9 is of course not simulated by the model.

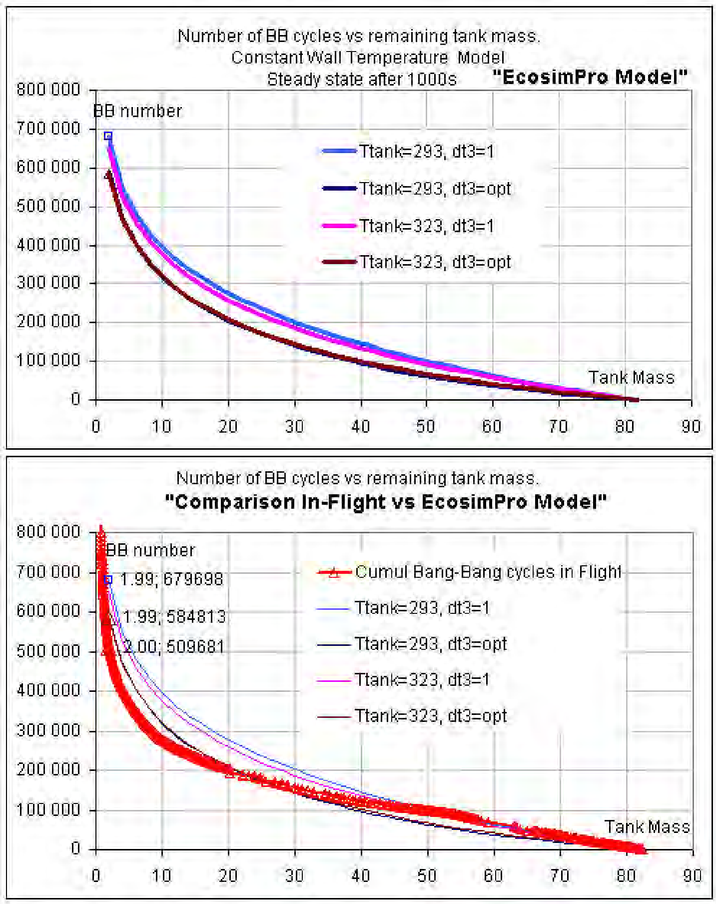


Fig. 11. EcosimPro® Model and Inflight results: Number of bang-bang cycles

The settings of the pressure regulation are mainly the timing parameters (duration of opening of the valves) and the target parameter of the regulation. The pressure step has been adjusted, during the satellite commissioning phase, by decreasing the duration for the downstream valve opening pulse: this is called the “shorter downstream valve opening strategy”. While the xenon mass consumption progress, it was decided to increase that same settings in February 2004 in order to recover the nominal settings to reduce the number of bang-bang cycles.

Number of bang-bang cycles

Among all the model outputs (temperatures, flows, pressures, timings), an important output to report here is the number of the valves activation forecasted by the model. The model predicts a range for the number of bang-bang cycles between 580 000 and 680 000 for the complete nominal use of xenon (remaining xenon mass 2 kg), and for various strategies, fig. 11.

The in-flight results are plotted on the second graph of fig. 11. For the same remaining xenon mass the number of bang-bang cycles measured in flight is slightly below than the one forecasted by the model (510 000 cycles). One shall add that the two strategies were followed in-flight : in a first part of the mission, the duration of the downstream valve opening was set to a shorter (0.5s) than optimal value. Thus we can conclude that the model provides us a rather pessimistic value. The behaviors of in-flight count and forecasted count are surprisingly very similar.

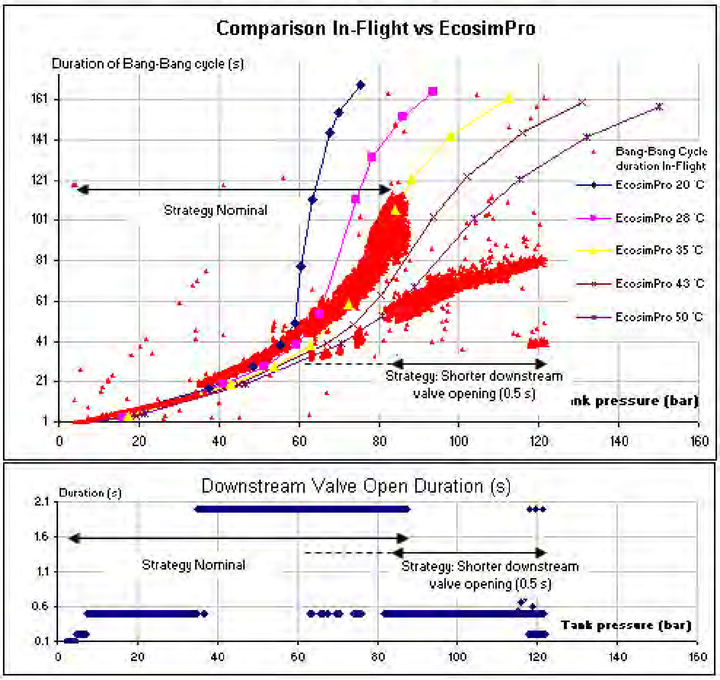


Fig. 12. EcosimPro Model and Inflight results: duration of the bang-bang cycle and Downstream valve opening pulse duration

Look-up table for the duration of the bang-bang cycle

In order to check the general behaviour of the pressurisation system (check of the leaktigthness and check of the full valve opening) the analysis of the duration of the bang-bang cycle was to be performed in-flight The system detects if the current value of the BB cycle duration stay in line with the look-up table computed by the simulation model. An out of limit from the table would means:

? If the BB cycle duration is too short with respect to the table: one of the valve poppet could jam

? If the BB cycle duration is too long: a weakness in the poppet/seat leak tightness of the downstream valve may occur.

Such check has been performed automatically by the software of the regulation after an adjustable period of inhibition in the automatic mode. The period of inhibition is needed because the first value depends on the initial state of the system (temperature, pressure into the plenum from the previous use). In case of positive check, the only action is to use the redundant branch of the Bang-Bang valves. This case never happens, the system behaves flawlessness..

The BB cycle duration is a function of the xenon temperature, of the xenon input pressure and of the strategies of the bang-bang valves opening duration. This output of the model that was integraded into the software of the EPS – PRE Card for the case of an optimal strategy of opening duration. The look-up table versus the xenon input pressure is plotted in fig. 12 with five curves at iso temperature from 20 to 50 °C.

In flight the temperature was regulated between 35 and 43 °C. The results in-flight are plotted on the same graph. The correlation between EcosimPro model and in-flight results is considered as extremely good, with of course the exception when the strategy of shorter downstream

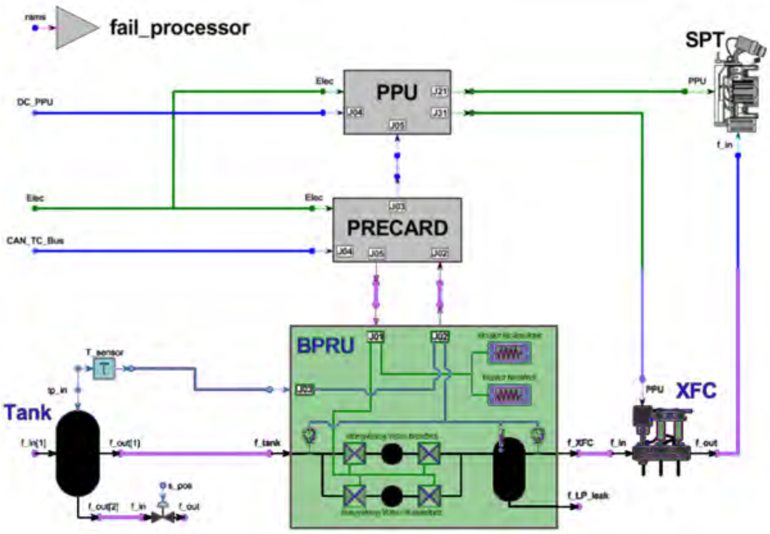


Fig. 13. EPOS an EPS simulation software which includes the functional model of the BPRU, and other components on EcosimPro®

valve opening pulse was followed. But even in that case, the shape of the in-flight results is compliant with the model results.

SECOND CHAPTER : INTEGRATION OF THE BPRU MODEL INTO THE EPS MODEL EPOS

EPOS is the acronym of Electric Propulsion Operation Software. The model is a simulation model of the whole EPS (including PPU, Thruster, Electronic Card, xenon tank, XFC and BPRU) that has been widely used by the team of ESOC Darmstadt [7] in order to learn how the satellite, and particularly the EPS, answer to the procedures for controlling the satellite. This software delivers a safe check of the procedures when updated.

The software chosen to produce EPOS was still EcosimPro for the same reasons as mentioned here before. The part relative to the BPRU simulation didn't require any additional work because the functional model was fully compliant with the architecture of EPOS as well as simulation time constrains, and hence the full BPRU EcosimPro model was included into EPOS. A sketch of EPOS is presented in fig. 13. The look of this sketch is the same as the system one, as shown on fig. 1.

EPOS includes the models describing all the subsystems and in addition, as required, a set of 78 failure case were included. One can see on fig. 13, that specific devices were added in the system for those failure cases: valves were added in the object component oriented software for example in order to be able to simulate leakages.

Finally, the EPOS model can be characterized by the following figures

Number of TM TC simulated: 165
Number of failure case: 78
Number of Petri diagram: 4
Number of loops: 1 PI (proportional integral) control loop, and 2 Petri loops.

Extension of SMART-1 mission forecasted by the model

The EPOS model was intensively used in the frame of the mission extension allowed to SMART-1 in order to set-up the final settings of the pressure regulation (BPRU). The presence of the full functional model of the BPRU into EPOS was greatly appreciated at that time: the real behaviour of the system could be simulated including the excitation of some health flags. EPOS was first used directly from the source model (see annex), on a simple PC station. Then the settings were included in the ESOC centre Satellite model for the overall check.
The results foreseen by the EPOS model were obtained rapidly by using a scale factor 1/50 in the characteristic dimensions of the Electric propulsion system (i.e. the tank of 50 litres became only 1 litre, and so on).
The following plot shows the events and behaviour of the system when the mission is being completed up to the "last drop" of usable xenon.

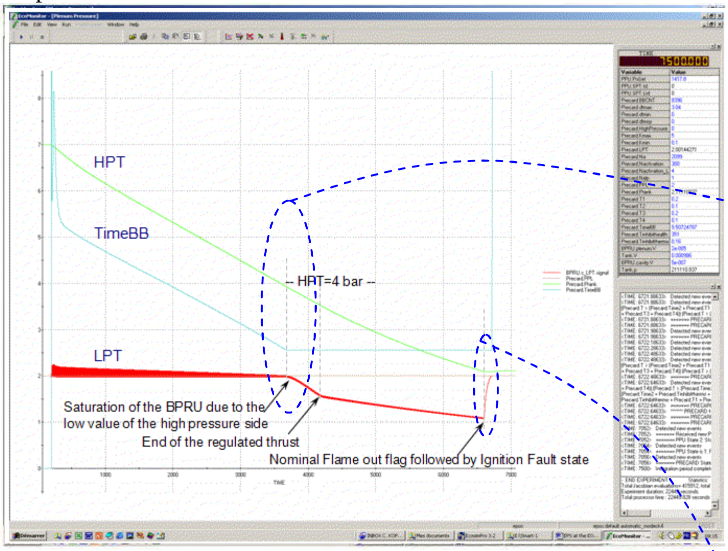


Fig. 14 : Simulation of the « End of xenon ». This characteristic event starts with the Saturation of the BPRU.

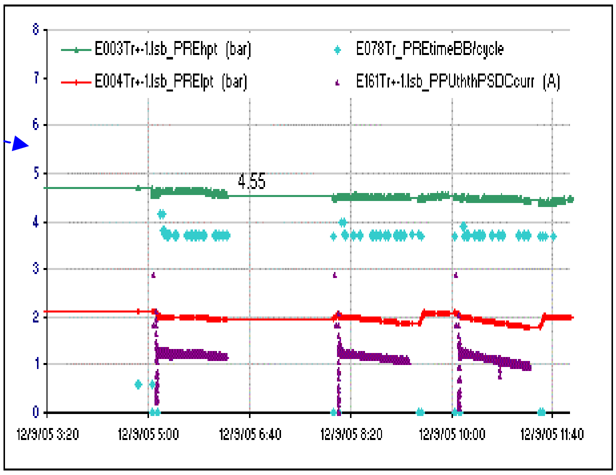


Fig. 15 : « End of xenon » real data in flight. Very similar behaviour to the one of the simulation (real time, real pulse duration here)

The Saturation of the BPRU occurs in the simulation with HPT at 3.9 bar (fig. 14) while in-flight, the raw HPT value recorded was 4.55 bar (fig. 15). Taking into account the shift of HPT (measured at the end of the mission) of +1.2 bar, the real HPT value was 3.35bar. This is a surprisingly remarkable achievement of the model. It forecast a lightly pessimistic value of only 0.55 bar above the real one.

The second point of comparison deal with the end of the

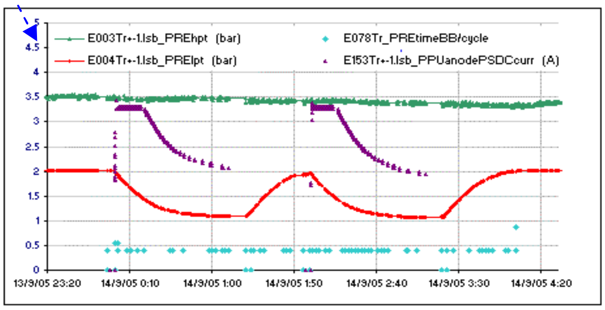


Fig. 16 : Re-pressurisations process real data in flight. Very similar behaviour to the one of the simulation (real time, real pulse duration here)

regulated thrust followed by the re-pressurisations process that occurs after the use of the thruster until a “flame out” flag occur (thruster turned off by the logic of the PPU when the discharge current decrease down the limit). In flight, fig. 16, the results are very similar compared to the simulation.

CONCLUSION

The simulation of the whole Electric propulsion system was based on the use of EcosimPro®. The main advantages of using this specific kind of simulation tool were its ability to manage the systems and its components like objects. There was no need to develop multiple software for functional purpose and for simulation purpose: the unique functional model has been integrated into the simulation model, thus sparing lots of efforts and money. As shown in the paper, the simulation results are much more representative of the real behaviour.

The comparison between forecasts from the simulation model and the real data of the hardware in flight has been performed and because there are no discrepancies, this is considered as very good performance: in particular the number of bang-bang cycles of the pressure regulation system, the predicted law for the duration of the bang-bang cycle, the pressure and the behaviour of the system after the thrust pulse at the end of the mission are fully similar.

The results obtained on the SMART-1 experience made the EcosimPro® software like a flight proven component. It is very important to know that we can be able to rely now on such powerful system model. Of course, the results obtained are primarily due to the very good quality of the equations used. But what is also required from system model, is the capabilities of the model for being integrated into the real time simulator of the spacecraft. This has been also fully demonstrated by EcosimPro® at ESOC Darmstadt.

References

[1] Saccoccia G., Estublier D., Racca G. " SMART-1: A Technology Demonstration Mission for Science Using Electric Propulsion," *AIAA-98-3330, 34th Joint Propulsion Conference*, July 1998, Cleveland, OH.

[2] Bushway E., Engelbrecht C., and Ganapathi G. “NSTAR Ion Engine Xenon Feed System: Introduction to system design and development,” IEPC 97-044, Aug. 1997.

[3] Koppel C., Lyszyk M., Valentian D., “PPS 1350 With Variable Power Feature for SMART-1,” *AIAA 2000-3427, 36th Joint Propulsion, Conference and Exhibit*, 17th-19th July 2000, Huntsville, Alabama.

[4] Ganapathi G and Engelbrecht C. “Performance of the Xenon Feed System on Deep Space One,” Journal of Spacecraft and Rockets, Vol. 37, N°3, Mat-June 2000.

[5] Estublier D., Koppel C., “SMART-1 End-to-end test: Final results and lesson learned,” *28th IEPC* 2003-0303, Toulouse, France.

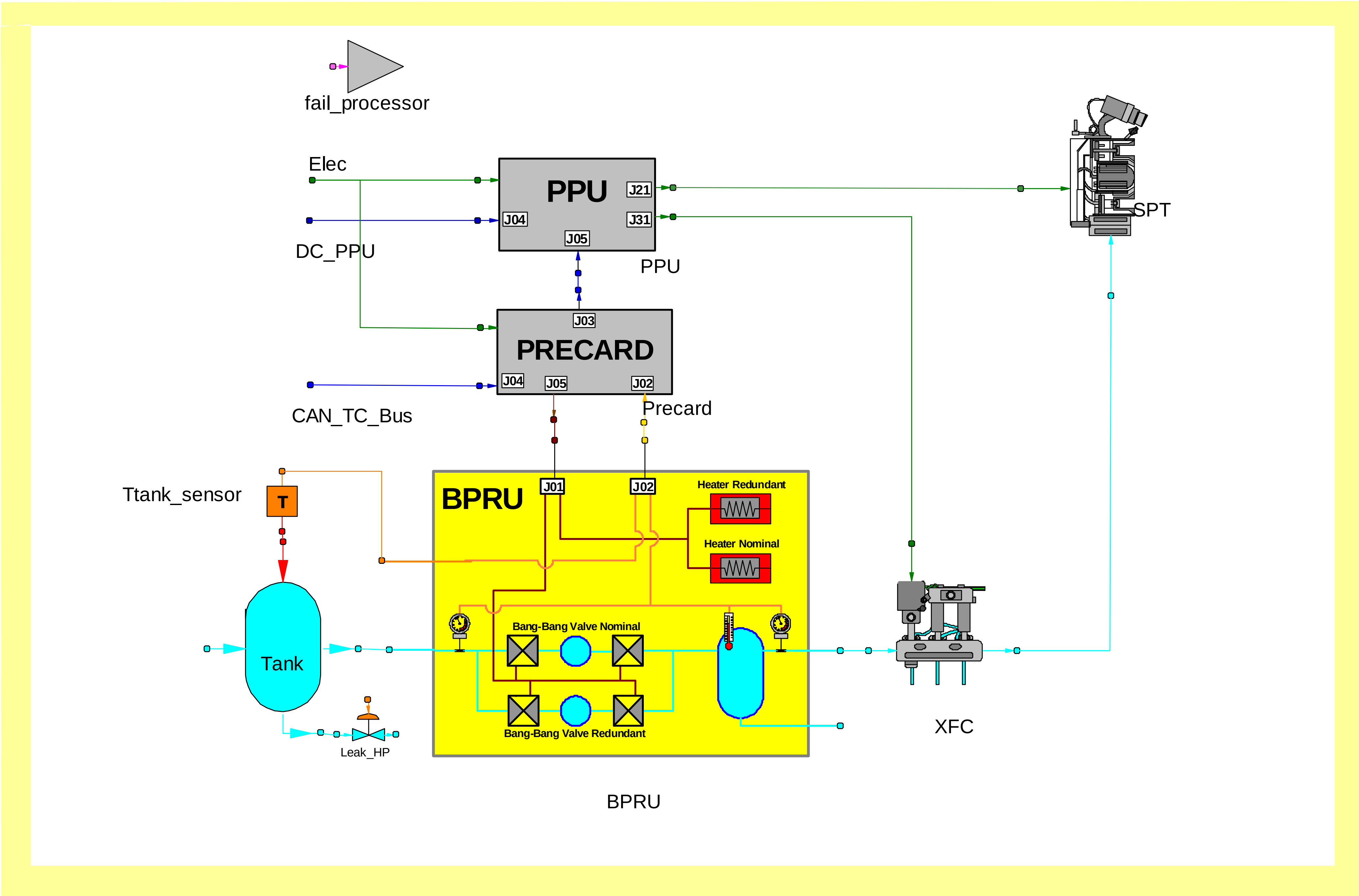
[6] Koppel C., Estublier D. “The Smart-1 Electric Propulsion Sub-System,” AIAA-2003-4545.

[7] Milligan D., Gestal D., Estublier D., Koppel C., “SMART1 Electric Propulsion Operations,” AIAA-2004-3436.

[8] Koppel C., Marchandise F., Estublier D. “Robust Pressure Regulation System for the SMART-1 Electric Propulsion Sub-System,” AIAA-2004-3977.

Annex document to the paper SESP2006-435470 Simulation of the Smart-1 Electric Propulsion System With a System Simulation Software Ecosimpro®

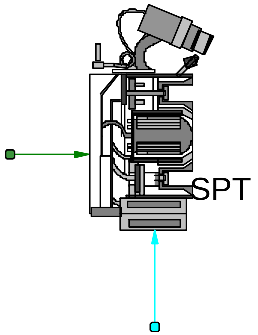
EPOS: Electric Propulsion Operation Software ***(including the functional model of the BPRU)***



Fundamental ports between components

```
-------------------------------------------------
--Fluid port type
-------------------------------------------------
PORT Fluid SINGLE
    REAL m   "mass flow (kg/s)"
    REAL ht  "total enthalpy (J/Kg)"
    REAL p   "pressure (bar)"
END PORT

--Port type to connect the PPU to the thruster
PORT PPU_SPT SINGLE
       REAL Ud
       REAL Id
       BOOLEAN AnodeSupply
       BOOLEAN MagnetSupply
       BOOLEAN HeaterSupply
       REAL Vignitor
END PORT

--Port type to connect the PPU to the Xenon flow controller
PORT PPU_XFC SINGLE
       REAL valves_A_pos
       REAL valves_B_pos
       BOOLEAN ThermothrottleSupply
       REAL Itt
END PORT

--Port Type for Failure Cases
--(They come from RAMS analysis)
PORT RAMS SINGLE IN
    EQUAL OUT INTEGER event = 0
END PORT

--Port type for CAN Telecommands & Telemetry
PORT CAN_TC_TM SINGLE
    INTEGER CAN_message_type = 0  --0= None, 1= LP_Command0, 2=
LP_Command1, 3= Poll
                                  --4= Poll Return --5 Pre TM
Return --6 PPU TM Return
    INTEGER TC_byte0 = 0
    INTEGER TC_byte1 = 0
    INTEGER TC_byte2 = 0
    INTEGER TC_byte3 = 0
    INTEGER LP_Data_0_11[12,8]
    INTEGER LP_Data_12_14[8]
    INTEGER LP_Data_15[8]
END PORT

--Port type for Direct Commands & Telemetry
PORT DR_TC SINGLE
    INTEGER  TC = 0  --0 = None, 1 = Direct main PPU On, 2 =
Direct main PPU Off,
                     --3 = Direct redundant PPU On, 4 = Direct
redundant PPU Off,
END PORT

--Port type to connect the BPRU Sensors to the PRECard
PORT PRE_BPRU_Sensors SINGLE
    REAL HPT          "High pressure signal (bar)"
    REAL LPT          "Low pressure signal (bar)"
    REAL Ttank        "Tank temperature signal (ºC)"
    REAL Tplenum      "Plenum temperature signal (ºC)"
END PORT

--Port type to connect the BPRU Actuators to the PRECard
PORT PRE_BPRU_Actuators SINGLE
    REAL V1_signal    "Upstream valve position V1 (0-1)"
    REAL V2_signal
    REAL V3_signal
    REAL V4_signal
    REAL Heater_Nom
    REAL Heater_Red
END PORT

--Electrical port type to calculate the total consumed power
PORT ElecPower
     SUM REAL power
END PORT
```

```
--------------VISUAL SOURCE SAFE INFO------------------------------
-- $Archive: /ECOSIM LIBRARIES/EPOS/WORK/EPOS_SPT_Compo.el $
-- $Author: Rpv $
-- $Revision: 9 $
-- $JustDate: 20/06/02 $
-------------------------------------------------------------------
--Component representing the Plasma thruster using performance
--curves provided by Snecma
-------------------------------------------------------------------
USE MATH


COMPONENT SPT

PORTS
       IN   PPU_SPT PPU      "Electrical connections to the PPU through the filter unit"
       IN   Fluid   f_in     "Fluid flow connection for Xe"

DATA

       REAL r_TAP         = 0    "Spherical coordinate r of the thrust application point (m)"
       REAL theta_TAP = 0    "Spherical coordinate theta of the thrust application point (rad)"
       REAL phi_TAP   = 0    "Spherical coordinate phi of the thrust application point (rad)"


       REAL theta_dir =  MATH.PI/2  "Spherical coordinate theta or elevation angle of thrust vector direction (rad)"
       REAL phi_dir   = -MATH.PI/2  "Spherical coordinate phi or azimuth angle of thrust vector direction (rad)"

     REAL   NeededHeatingTime = 160 "Needed heating time (s)"
     REAL   Tswirl_nom = 3.2e-5  "Nominal value of swirl torque (Nw*m)"
     REAL   Tswirl_min = 1.21e-5 "Minimun value of swirl torque (Nw*m)"
     REAL   Tswirl_max = 5.21e-5 "Maximum value of swirl torque (Nw*m)"

DECLS
       CONST REAL go     = 9.80665          "Gravity acceleration (m^2/s)"
       CONST REAL q      = 1.609e-19        "Electrical charge of one electron (C)"
       CONST REAL MXe    = 0.13129/6.022e23 "Mass of one Xe atom (kg)"
     CONST REAL Uo       = 40                "Minimum anode voltage for producing positive thrust (V)"

       INTEGER  NeededIgnitionPulses = 1     "Number of ignition pulses needed for ignition"

       REAL m                                "Xenon mass flow (kg/s)"
       REAL Ud                               "Discharge voltage (V)"
       REAL Id                               "Actual discharge current (A)"
       REAL Isp                              "Specific impulse (s)"
       REAL Pd                               "Discharge power (W)"
       REAL F                                "Thrust (Nw)"
```



```
    REAL rx                                   "X position of thrust aplication point (m)"
    REAL ry                                   "Y position of thrust aplication point (m)"
    REAL rz                                   "Z position of thrust aplication point(m)"

    REAL Fx                                   "Component X of thrust vector (N)"
    REAL Fy                                   "Component Y of thrust vector (N)"
    REAL Fz                               "Component Z of thrust vector (N)"

  REAL T_swirl                              "Swirl Torque (N m)"
    REAL Tx                                   "Component X of thrust torque (N m)"
    REAL Ty                                   "Component Y of thrust torque (N m)"
    REAL Tz                                   "Component Z of thrust torque (N m)"


  REAL Th_start = 1e40                      "Time for beginning of heating (s)"

  INTEGER IgnitionPulseCount = 0

    BOOLEAN AlreadyHeated = FALSE
    BOOLEAN AlreadyIgnited = FALSE
```

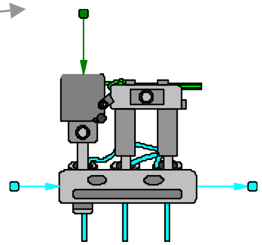
DISCRETE

```
    WHEN (PPU.Vignitor > 30) THEN
        IgnitionPulseCount = IgnitionPulseCount + 1
    END WHEN

    WHEN (AlreadyHeated AND (0.98e6*m > 1)AND \
             IgnitionPulseCount >= NeededIgnitionPulses) THEN
        AlreadyIgnited = TRUE AFTER 0.003  --It is assumed that the ignitiont time is
                                           --less than the pulse duration
    END WHEN

    WHEN (PPU.HeaterSupply) THEN
          Th_start = TIME
    END WHEN

    WHEN (NOT PPU.HeaterSupply) THEN
          Th_start = 1e40
    END WHEN

    WHEN (TIME - Th_start > NeededHeatingTime) THEN
          AlreadyHeated = TRUE
    END WHEN
    --
  --  WHEN (AlreadyIgnited AND (NOT PPU.AnodeSupply OR m < 0.01e-6)) THEN
    WHEN (NOT PPU.AnodeSupply OR m < 0.01e-6) THEN
          AlreadyHeated = FALSE AFTER 0
          AlreadyIgnited = FALSE AFTER 0
          IgnitionPulseCount = 0 AFTER 0
    END WHEN
    --Failure Cases
    WHEN (ActFail[Fail_Flame_after_000_ignition_attempts]) THEN
         NeededIgnitionPulses = 0
    END WHEN
    WHEN (ActFail[Fail_Flame_after_002_ignition_attempts]) THEN
         NeededIgnitionPulses = 2
    END WHEN
```

```
        WHEN (ActFail[Fail_Flame_after_089_ignition_attempts]) THEN
             NeededIgnitionPulses = 89
        END WHEN
        WHEN (ActFail[Fail_Flame_after_200_ignition_attempts]) THEN
             NeededIgnitionPulses = 200
        END WHEN
        WHEN (NOT ActFail[Fail_Flame_after_200_ignition_attempts] AND \
              NOT ActFail[Fail_Flame_after_002_ignition_attempts] AND \
              NOT ActFail[Fail_Flame_after_089_ignition_attempts] AND \
              NOT ActFail[Fail_Flame_after_000_ignition_attempts]) THEN
             NeededIgnitionPulses = 1
        END WHEN
CONTINUOUS
        -- port equations -------------------
       m = f_in.m
        PPU.Ud = Ud
        PPU.Id = Id
        ------------ vectores posicion, empuje, momento -------------
        rx = r_TAP * sin(theta_TAP)*cos(phi_TAP)
        ry = r_TAP * sin(theta_TAP)*sin(phi_TAP)
        rz = r_TAP * cos(theta_TAP)

        Fx = F*sin(theta_dir)*cos(phi_dir)
        Fy = F*sin(theta_dir)*sin(phi_dir)
        Fz = F*cos(theta_dir)

        Tx = ry*Fz - rz*Fy + T_swirl*sin(theta_dir)*cos(phi_dir)
        Ty = rz*Fx - rx*Fz + T_swirl*sin(theta_dir)*sin(phi_dir)
        Tz = rx*Fy - ry*Fx + T_swirl*cos(theta_dir)
        ------------------------------------------------------------------------------------------
        ------------ ecs motor --------------------------------------------------------------------

        F   = ZONE (PPU.AnodeSupply AND (0.98e6*m > 1) AND (Ud > Uo) AND AlreadyIgnited) 0.845 * ssqrt(2*q*(Ud - Uo)/MXe) * \
                                                                            m  * (1 - exp(-m*1e6/1.791))
                 OTHERS                                                            0
        Isp = ZONE (PPU.AnodeSupply AND (0.98e6*m > 1) AND AlreadyIgnited)    F/((m+1e-10)*go)
                 OTHERS                                                            0
IMPL(Id)      Id = ZONE (PPU.AnodeSupply AND (0.98e6*m > 1) AND AlreadyIgnited)     (m*1e6 - 0.837)*(7.10/max(Ud, 175) + 0.918)
                 OTHERS                                                            0

       T_swirl = ZONE (NOT(PPU.AnodeSupply AND (0.98e6*m > 1) AND (Ud > Uo) AND AlreadyIgnited))
                        0.
                 ZONE (ActFail[Fail_Zero_Torque])
                        0
                 ZONE (ActFail[Fail_Mini_Swirl_Torque] AND NOT ActFail[Fail_Wrong_Swirl_Orientation])
                          Tswirl_min
                 ZONE (ActFail[Fail_Mini_Swirl_Torque] AND ActFail[Fail_Wrong_Swirl_Orientation])
                       -Tswirl_min
```

```
                ZONE (ActFail[Fail_Max_Swirl_Torque] AND NOT ActFail[Fail_Wrong_Swirl_Orientation])
                         Tswirl_max
                ZONE (ActFail[Fail_Max_Swirl_Torque] AND ActFail[Fail_Wrong_Swirl_Orientation])
                      -Tswirl_max
                ZONE (NOT ActFail[Fail_Wrong_Swirl_Orientation])
                      Tswirl_nom
                ZONE (ActFail[Fail_Wrong_Swirl_Orientation])
                     -Tswirl_nom
                OTHERS
                     0
      --Discharge Power
      Pd = Id * Ud
       -----------------------------------------------------------------------------------------
      f_in.p = 1e-9
END COMPONENT
```

```
--------------VISUAL SOURCE SAFE INFO------------------------------
-- $Archive: /ECOSIM LIBRARIES/EPOS/WORK/EPOS_XFC_Compo.el $
-- $Author: Rpv $
-- $Revision: 4 $
-- $JustDate: 26/07/02 $
-------------------------------------------------------------------
--Component representing the Xe flow controller
-------------------------------------------------------------------
USE CONTROL
COMPONENT XFC
PORTS
       IN PPU_XFC PPU
       IN  EPOS.Fluid f_in
      OUT EPOS.Fluid f_out
DATA
      REAL tau_tt = 2. "Time constant for the thermothrottle (s)"
DECLS
       REAL PXe    "Xenon pressure (bar)"
       REAL m "Mass flow (kg/s)"
       REAL m_ss   "Steady State Mass flow (kg/s)"
       REAL Itt    "Thermothrottle current (A)"
INIT
      m = 0.
CONTINUOUS
      PXe = f_in.p * 1e-5
       m_ss  =  (PPU.valves_A_pos + PPU.valves_B_pos) * \
                     1e-6* 2 * PXe**1.73 / (1. + 11.94 * Itt**8.17)**0.1
      m' = (m_ss - m) / tau_tt
      --Conservation of mass
      f_in.m = m
      f_out.m = m
      --Conservation of energy
      f_in.ht = f_out.ht
       -- port equations ------
       Itt = PPU.Itt
       ------------------------
END COMPONENT
```

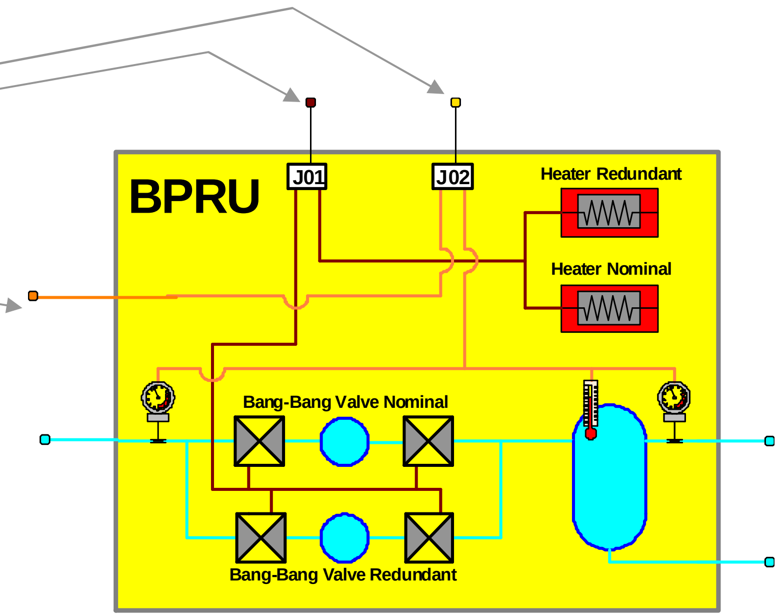


XFC

```
-- Generated automatically with EcoDiagram 1.1.8, 28/6/2005 17:36:58
COMPONENT BPRU
```

PORTS

```
    OUT EPOS.PRE_BPRU_Sensors J02
    IN EPOS.PRE_BPRU_Actuators J01
    IN CONTROL.analog_signal J03
    IN EPOS.Fluid f_tank
    OUT EPOS.Fluid f_XFC
    OUT EPOS.Fluid f_LP_leak

 TOPOLOGY
   EPOS.Tube(
      Nodes = 1,      -- Non default value.
      Wall_Energy =  TRUE ) tube1(
      fluid = Xe,
      D = 0.00177,    -- Non default value.
      l = 0.21,       -- Non default value.
      rug = 5e-005,
      m_wall = 0.08, -- Non default value.
      cp_wall = 580, -- Non default value.
      L_D = 0,
      f_L_D = 0,
      po = 7550000,  -- Non default value.
      To = 20,
      T_wall_o = 20)

   EPOS.Volume2(
      Wall_Energy =  TRUE ) cavity(
      fluid = Xe,
      V = 5e-007,     -- Non default value.
      A_cross = 7.616e-005, -- Non default value.
      A_wall = 0.0003046,   -- Non default value.
      m_wall = 0.21, -- Non default value.
      cp_wall = 550, -- Non default value.
      Mo = 0,
      po = 200000,    -- Non default value.
      To = 20,
      T_wall_o = 20)

   EPOS.Tube(
      Nodes = 1,      -- Non default value.
      Wall_Energy =  TRUE ) tube2(
      fluid = Xe,
      D = 0.00177,    -- Non default value.
```



BPRU

```
      l = 0.391,      -- Non default value.
      rug = 5e-005,
      m_wall = 0.0165,      -- Non default value.
      cp_wall = 550, -- Non default value.
      L_D = 0,
      f_L_D = 0,
      po = 200000,    -- Non default value.
      To = 20,
      T_wall_o = 20)

   THERMAL.Heater heater

   THERMAL.BNode Sat_Mid_Deck(
      Label = "Node Label",
      qi = 0,
      T = 20   -- Non default value.
   )
```

```
THERMAL.DNode Plate(
   Label = "Node Label",
   qi = 0,
   To = 20,
   Boundary =  FALSE ,
   C = 1056.15    -- Non default value.
)

THERMAL.DNode PT(
   Label = "Node Label",
   qi = 0,
   To = 20,
   Boundary =  FALSE ,
   C = 278.4      -- Non default value.
)

THERMAL.GL GL_tube1_plate(
   cond = 0)

THERMAL.GL GL_BB1_plate(
   cond = 0.13    -- Non default value.
)

THERMAL.GL GL_tube2_plate(
   cond = 0.32    -- Non default value.
)

THERMAL.GL GL_plenum_plate(
   cond = 0.254   -- Non default value.
)

THERMAL.GL GL_PT_plate(
   cond = 0.13    -- Non default value.
)

THERMAL.GL GL_plate_Sat_Mid_Deck(
   cond = 63.76   -- Non default value.
)

THERMAL.GR GR_PT_environment(
   REF = 0.0013256       -- Non default value.
)

THERMAL.GR GR_BB1_environment(
   REF = 0.0008291       -- Non default value.
)
```

```
THERMAL.GR GR_plenum_environment(
   REF = 0.0029838       -- Non default value.
)

THERMAL.GR GR_plate_environment(
   REF = 0.024276 -- Non default value.
)

THERMAL.GR GR_PT_Plate(
   REF = 0.0002809       -- Non default value.
)

THERMAL.GR GR_BB1_Plate(
   REF = 0.00019031      -- Non default value.
)

THERMAL.GR GR_plenum_plate(
   REF = 0.00044884      -- Non default value.
)

THERMAL.GR GR_BB1_PT(
   REF = 1.7367e-006     -- Non default value.
)

THERMAL.GR GR_PT_plenum(
   REF = 7.5747e-006     -- Non default value.
)

THERMAL.GR GR_BB1_plenum(
   REF = 6.3238e-006     -- Non default value.
)

EPOS.Valve valve1(
   fluid = Xe,
   Ao = 1.365e-008,      -- Non default value.
   zeta = 0.7,    -- Non default value.
   dp_lam = 2000)

EPOS.Valve valve2(
   fluid = Xe,
   Ao = 1.365e-008,      -- Non default value.
   zeta = 0.7,    -- Non default value.
   dp_lam = 2000)

EPOS.Valve Leak_LP(
   fluid = Xe,
   Ao = 6.65289647e-011, -- Non default value.
```

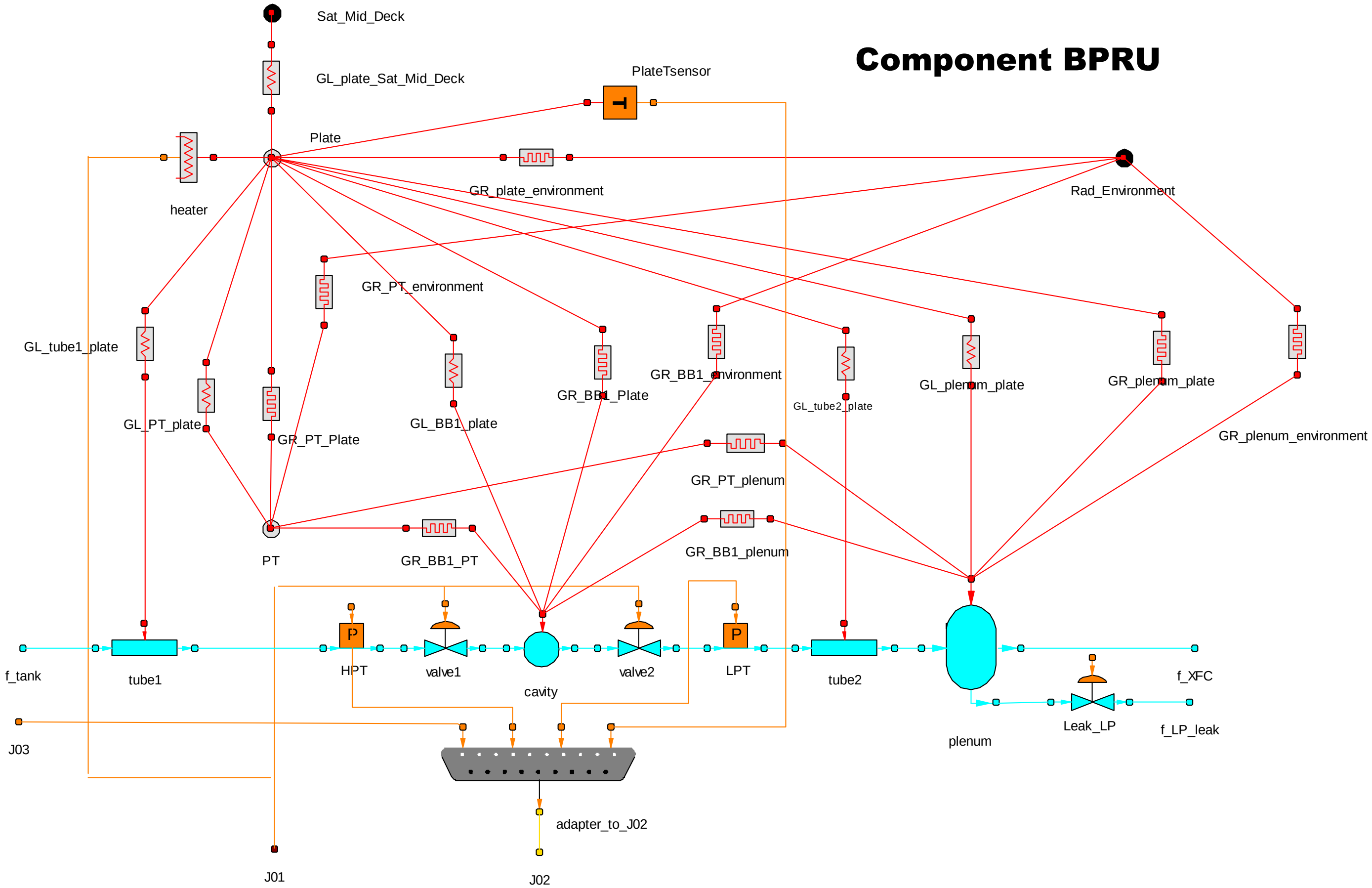
```
   zeta = 1,       -- Non default value.
   dp_lam = 2000)

EPOS.PSensor HPT(
   gain = 1e-005, -- Non default value.
   bias = 0)

EPOS.PSensor LPT(
   gain = 1e-005, -- Non default value.
   bias = 0)

EPOS.Volume3(
   Wall_Energy =  TRUE ) plenum(
   fluid = Xe,
   V = 0.001,
   A_cross = 0.01209,
   A_wall = 0.04836,
   m_wall = 0.54, -- Non default value.
   cp_wall = 580, -- Non default value.
   Mo = 0,
   po = 200000,   -- Non default value.
   To = 20,
   T_wall_o = 20)

THERMAL.BNode Rad_Environment(
   Label = "Node Label",
   qi = 0,
   T = 20  -- Non default value.
)

THERMAL.T_sensor PlateTsensor(
   gain = 1,
   bias = 0)

EPOS.Adapter_signals_to_BPRU_Sensor adapter_to_J02


CONNECT GL_BB1_plate.tp_in TO cavity.tp_in
CONNECT GL_plate_Sat_Mid_Deck.tp_out TO Sat_Mid_Deck.tp_in
CONNECT GL_tube2_plate.tp_in TO tube2.tp_in
CONNECT GR_BB1_PT.tp_in TO cavity.tp_in
CONNECT GR_BB1_Plate.tp_in TO cavity.tp_in
CONNECT GR_BB1_environment.tp_in TO cavity.tp_in
CONNECT f_tank TO tube1.f_in
CONNECT cavity.f_in[1] TO valve1.f_out
CONNECT cavity.f_out[1] TO valve2.f_in
CONNECT Leak_LP.f_out TO f_LP_leak
CONNECT GR_BB1_plenum.tp_in TO cavity.tp_in
CONNECT HPT.f_in TO tube1.f_out
CONNECT HPT.f_out TO valve1.f_in
CONNECT LPT.f_in TO valve2.f_out
CONNECT LPT.f_out TO tube2.f_in
CONNECT Leak_LP.f_in TO plenum.f_out[2]
CONNECT f_XFC TO plenum.f_out[1]
CONNECT GR_BB1_plenum.tp_out TO plenum.tp_in
CONNECT plenum.f_in[1] TO tube2.f_out
CONNECT GL_tube1_plate.tp_out TO Plate.tp_in
CONNECT GL_tube1_plate.tp_in TO tube1.tp_in
CONNECT GR_BB1_PT.tp_out TO PT.tp_in
CONNECT GL_BB1_plate.tp_out TO Plate.tp_in
CONNECT GL_PT_plate.tp_out TO Plate.tp_in
CONNECT GR_BB1_Plate.tp_out TO Plate.tp_in
CONNECT GL_PT_plate.tp_in TO PT.tp_in
CONNECT GR_PT_Plate.tp_out TO Plate.tp_in
CONNECT GL_plenum_plate.tp_out TO Plate.tp_in
CONNECT GL_plate_Sat_Mid_Deck.tp_in TO Plate.tp_in
CONNECT Plate.tp_in TO heater.tp_out
CONNECT GR_PT_Plate.tp_in TO PT.tp_in
CONNECT GR_PT_environment.tp_in TO PT.tp_in
CONNECT GL_plenum_plate.tp_in TO plenum.tp_in
CONNECT GR_plenum_environment.tp_in TO plenum.tp_in
CONNECT GR_PT_plenum.tp_in TO PT.tp_in
CONNECT GL_tube2_plate.tp_out TO Plate.tp_in
CONNECT GR_PT_plenum.tp_out TO plenum.tp_in
CONNECT GR_plenum_plate.tp_out TO Plate.tp_in
CONNECT GR_plenum_plate.tp_in TO plenum.tp_in
CONNECT GR_plate_environment.tp_in TO Plate.tp_in
CONNECT GR_plenum_environment.tp_out TO Rad_Environment.tp_in
CONNECT GR_plate_environment.tp_out TO Rad_Environment.tp_in
CONNECT GR_PT_environment.tp_out TO Rad_Environment.tp_in
CONNECT GR_BB1_environment.tp_out TO Rad_Environment.tp_in
CONNECT HPT.s_measure TO adapter_to_J02.s_HPT
CONNECT LPT.s_measure TO adapter_to_J02.s_LPT
CONNECT J03 TO adapter_to_J02.s_Ttank
CONNECT PlateTsensor.s_measure TO adapter_to_J02.s_Tplenum
CONNECT J02 TO adapter_to_J02.sensors_wire
CONNECT Plate.tp_in TO PlateTsensor.tp_in
```

DISCRETE

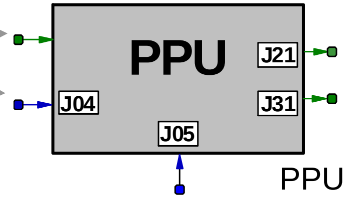
```
WHEN (ActFail[Fail_Leak_LP_0250_g_per_year])  THEN
    Leak_LP.s_pos.signal = 250 / 2250
    END WHEN
WHEN (ActFail[Fail_Leak_LP_1000_g_per_year])  THEN
```

```
        Leak_LP.s_pos.signal = 1000 / 2250
        END WHEN
    WHEN (ActFail[Fail_Leak_LP_2250_g_per_year])  THEN
        Leak_LP.s_pos.signal = 1
        END WHEN
    WHEN (NOT ActFail[Fail_Leak_LP_2250_g_per_year] AND NOT
ActFail[Fail_Leak_LP_1000_g_per_year] AND NOT
ActFail[Fail_Leak_LP_0250_g_per_year])  THEN
        Leak_LP.s_pos.signal = 0
        END WHEN

CONTINUOUS
    heater.s_power.signal = J01.Heater_Nom + J01.Heater_Red
    valve1.s_pos.signal = J01.V1_signal + J01.V3_signal
    valve2.s_pos.signal = J01.V2_signal + J01.V4_signal

END COMPONENT
```

Component BPRU



```
PARTS of COMPONENT PPU --(BOOLEAN REAL_TIME = TRUE)
    PORTS
        IN   ElecPower   Elec  "Port for the requested electrical power"
        IN   DR_TC       J04   "Port for direct commands"
        IN   CAN_TC_TM   J05   "Port for the CAN Telecommands/Telemetry"
        OUT  PPU_SPT     J21   "Port for the electrical connections to SPT"
        OUT  PPU_XFC     J31   "Port for the electrical connections to the SFC"
    DATA
        REAL a = 123.      "Impedance for discharge current/voltage characteristics (Ohm)"

        REAL Idcrit = 1.35 "Current limit to consider the thruster ignited (A)"
        REAL Vacrit = 160. "Voltage limit to consider a fault condition on discharge behaviour (V)"

        REAL Plow = 649.3   "Maximum Power specified for the ignition of the thruster (W)"
........................................................
........................................................
........................................................
        REAL c_eta_PPU = 0       "Quadratic coefficient for calculation of PPU efficiency"

    DECLS
........................................................
........................................................
........................................................
    INIT
        FOR (i IN PPU_STATES)
            Bv[i] = FALSE
        END FOR

        Bv[PPU_State_1] = TRUE
        --Initialisation of dynamic variables
        Eaux = 0
        Pnom = 0
        IttLoop = 0
    DISCRETE
        --PROCESS DIRECT TELECOMMANDS
        WHEN(J04.TC != 0 AND PowerON) THEN
            IF (J04.TC == 1) THEN
                --Main PPU On
                IF (PPU_ON == FALSE) THEN
                    PPU_ON = TRUE
                    ppu_unit = Main_PPU
                END IF
            ELSEIF (J04.TC == 2 AND ppu_unit == Main_PPU) THEN
........................................................
........................................................
........................................................
        WHEN (NOT ActFail[Fail_Thermothrottle_voltage_TM_not_available]) THEN
```



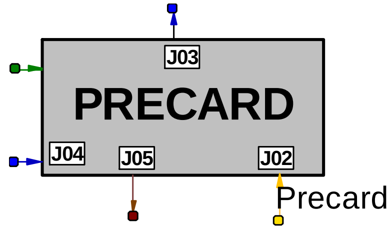
```
    IF (Error_byte == 1*16 + 7) THEN
        Error_byte = 0
    END IF
END WHEN
```

CONTINUOUS

```
dT1 = max(TIME - dT1start, 0.)
T = max(TIME - Tstart, 0.)
vT = max(TIME - vTstart, 0.)
T_PPU_ON = max(TIME - TpowerStart, 0.)

Pnom' = ZONE (PnSet > Pnom + 0.01 AND vT > Tlow)  6.625 / 0.4
        ZONE (PnSet < Pnom - 0.01 AND vT > Tlow) -6.625 / 0.4
        OTHERS 0.
E=  IdREF - Id
IdREF = ZONE (NOT Discharge_I_TC AND AnodeSupply)   Vd_vs_Power(Pnom)/100 - 0.1
        ZONE (Discharge_I_TC AND AnodeSupply)       IdSet
        OTHERS                                      0.
Im= ZONE (Magnet_I_TC AND MagnetSupply)         ImSet
    ZONE (NOT Magnet_I_TC AND MagnetSupply)     Im_vs_Power(Pnom)
    OTHERS                                     0
Va = ZONE (NOT AnodeSupply)
         0
     ZONE (AnodeSupply AND Id < Vd_vs_Power(Pnom) / 100. )
         Vd_vs_Power(Pnom)
     ZONE (AnodeSupply AND Id < Vd_vs_Power(Pnom) * (1./100. +  1./a))
         Vd_vs_Power(Pnom)  - a * (Id - Vd_vs_Power(Pnom)/100.)
     OTHERS
         0
IttLoop'= ZONE (ThermothrottleSupply AND LoopON AND Id > Idcrit)     -K*Eaux'-(K-L)/0.1*E
          OTHERS                                                0
 Itt = ZONE (NOT(ActFail[Fail_of_Thermothrottle_heater]) AND \
            ThermothrottleSupply AND LoopON AND Id > Idcrit)   max(IttLoop, 0)
        ZONE (NOT(ActFail[Fail_of_Thermothrottle_heater]) AND \
            ThermothrottleSupply)                               ItSet
       ZONE (ActFail[Fail_of_Thermothrottle_heater])
            0
      OTHERS
            0
Eaux' = (E-Eaux)/t_filter

 Ih =  ZONE (HeaterSupply) min((TIME-Thstart)*0.3/0.1, IhSet )
       OTHERS                            0
J21.Ud = Va
J21.Id = Id
J21.AnodeSupply = AnodeSupply
J21.MagnetSupply = MagnetSupply
```

```
        J21.HeaterSupply = HeaterSupply
        J21.Vignitor = Vignitor
        J31.Itt = Itt
        --Voltages
        Vh = Ih * Rh
        Vtt = Itt * Rtt
        Vm =  Im * Rm
        --Power
         Pa =        Va * Id
         Pheater = Ih**2 * Rh
         Pignitor = ZONE (IgnitionPulse) 34.6
                   OTHERS 0
         Pmagnet = (Im + Id)* Im * Rm
         Pvalves =  ZONE(ValveSupply) 14.6
                  OTHERS 0
        Pthrottle = Itt**2 * Rtt
        eta_PPU = a_eta_PPU + b_eta_PPU*Pa + c_eta_PPU*Pa**2
        PPU_PowerComsuption = ZONE (PPU_ON AND NOT ActFail[Fail_Excesive_PPU_power_1_10_times] \
                                          AND NOT ActFail[Fail_Excesive_PPU_power_1_25_times] \
                                              AND NOT ActFail[Fail_Excesive_PPU_power_1_50_times] \
                                          AND NOT ActFail[Fail_Excesive_PPU_power_2_00_times])
                                     (Pa + Pheater + Pignitor + Pmagnet + Pvalves + Pthrottle) / eta_PPU
                            ZONE (PPU_ON AND ActFail[Fail_Excesive_PPU_power_1_10_times])
                              1.10 * (Pa + Pheater + Pignitor + Pmagnet + Pvalves + Pthrottle) / eta_PPU
                            ZONE (PPU_ON AND ActFail[Fail_Excesive_PPU_power_1_25_times])
                              1.25 * (Pa + Pheater + Pignitor + Pmagnet + Pvalves + Pthrottle) / eta_PPU
                                 ZONE (PPU_ON AND ActFail[Fail_Excesive_PPU_power_1_50_times])
                              1.50 * (Pa + Pheater + Pignitor + Pmagnet + Pvalves + Pthrottle) / eta_PPU
                                 ZONE (PPU_ON AND ActFail[Fail_Excesive_PPU_power_2_00_times])
                              2.00 * (Pa + Pheater + Pignitor + Pmagnet + Pvalves + Pthrottle) / eta_PPU
                            ZONE (NOT PPU_ON) 0
                            OTHERS             0
        Elec.power = PPU_PowerComsuption
        mainCurrent = PPU_PowerComsuption / 50.
        Id_out_regulation = ZONE(Id > 3.5)  1.
                            ZONE(Id > 1.75) 0.
                            OTHERS          1.
        Va_LT_Vacrit = ZONE (Va < Vacrit AND NOT ActFail[Fail_Anode_voltage_TM_not_available]) 1
                     OTHERS  0.
        Id_GT_Idcrit = ZONE (Id > Idcrit) 1
                     OTHERS  0.
        CRP_EGRN_DC_voltage = ZONE (Va > Vacrit) -20
                     OTHERS  0.
        --Oscillation current 1mA when Va < Vacrit
        FFU_oscillation_current = ZONE (Va < Vacrit) 0.001
                     OTHERS  0.
END COMPONENT
```

```
PARTS of COMPONENT PRECARD

PORTS
    IN  CAN_TC_TM          J04            "Input port for CAN TC"
    OUT CAN_TC_TM          J03            "Output port for CAN TC to PPU"
    IN  PRE_BPRU_Sensors   J02            "Input for BPRU sensors"
    OUT PRE_BPRU_Actuators J05            "Output to BPRU Valves & Heaters"
    IN  ElecPower          Elec           "Electrical power to the Precard (W)"
```




```
 DATA
    REAL HeaterPower = 3         "Electrical Power to BPRU heater when ON (W)"
DECLS
    --Enumerative vars
    ENUM PRE_TC_Name tc_name
    ENUM PRE_TM_Name tm_name
    ENUM TCOMMAND_TYPE tc_type
    --Parameters commanded by telecommands
    REAL T1 = 1     RANGE  0., 5.               "Opening time of first BB valve (s)"
    REAL T2 = 0.1   RANGE  0., 200.             "Inhibition time between first and second BB valve (s)"
    REAL T3 = 1     RANGE  0., 5.               "Opening time of second BB valve (s)"
........................................................
........................................................
........................................................
    REAL V4_closed_pos  --Most closed position for valve V4

INIT
    EncodePreCardData(3, PRE_DATA_PPL, dummy1, dummy2, dummy3)
    FOR (i IN PRE_STATES)
        Bv[i] = FALSE
    END FOR
    Bv[PRE_State_1] = TRUE
DISCRETE
    --TC Processing
    WHEN (J04.CAN_message_type == CAN_LP_Command0) THEN
          IF (PowerON) THEN
            IF (DEBUG_EPOS) THEN
                PRINT("======= LP Command 0 received by PreCard: \t byte0=$J04.TC_byte0 \
                           \t byte1=$J04.TC_byte1 \t byte2=$J04.TC_byte2 =======")
            END IF
........................................................
........................................................
........................................................
    --Transition 654 TO 656
    WHEN (Bv[PRE_State_654] AND (HighPressure==1 OR \
            NOT (Nalp < Nactivation_LP)) AND \
            (T > Time2 + T1 + T2 + T3 + T4)) THEN
        Bv[PRE_State_654] = FALSE AFTER 0
        Bv[PRE_State_656] = TRUE AFTER 0
```

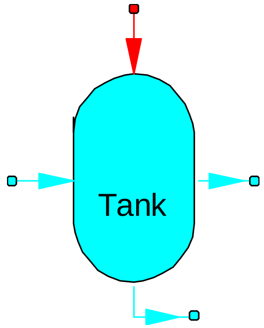
```
    END WHEN
    --Transition 654 TO 655
   --WHEN (Bv[PRE_State_654] AND NOT(HighPressure==1) AND             (Nalp < Nactivation_LP) AND            (T > Time2 +( T1 + T2 + T3 + T4)*(Nalp+1) )) THEN
-- ADDED CKOPPEL  JUNE 2005
     WHEN (Bv[PRE_State_654] AND NOT(HighPressure==1) AND \
             (Nalp < Nactivation_LP) AND \
            (T > Time2 + T1 + T2 + T3 + T4) AND BBCycleFullCompleted ) THEN        --  ADDED CKOPPEL  JUNE 2005
        Bv[PRE_State_654] = FALSE AFTER 0
        Bv[PRE_State_655] = TRUE AFTER 0
    END WHEN
    --Transition 655 TO 654
    WHEN (Bv[PRE_State_655]) THEN
       Bv[PRE_State_655] = FALSE AFTER 0
       Bv[PRE_State_654] = TRUE AFTER 0
       T_GT_Time2_T4 = FALSE AFTER 0
    END WHEN
........................................................
........................................................
........................................................
    --Desactivation of Critical RAM Failure
    WHEN (NOT(ActFail[Fail_PRE_Critical_RAM])) THEN
        Verify_Transmit(TRUE, b10, PreCard_mode, health_bits, Exception_count, ERRVECT)
    END WHEN
```

CONTINUOUS

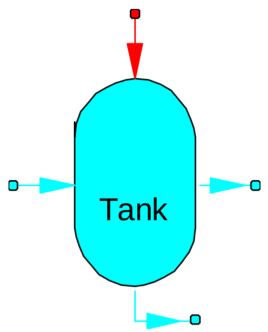
```
        T = max(TIME - Tstart, 0)
        TpowerON = max(TIME - TstartPowerON, 0)
        Ptank =  ZONE (ActFail[Fail_of_HPT_min_val])     PRE_DATA_VALUE[PRE_TM_HPT, MIN_RANGE] --Failure of HPT sensor: Minimum value
                 ZONE (ActFail[Fail_of_HPT_max_val])     PRE_DATA_VALUE[PRE_TM_HPT, MAX_RANGE] --Failure of HPT sensor: Maximum value
                 ZONE (ActFail[Fail_of_HPT_last_val])    HPT_last                              --Failure of HPT sensor: Last value
                 ZONE (ActFail[Fail_of_HPT_plus_1_5])    J02.HPT + 1.5
                 ZONE (ActFail[Fail_of_HPT_minus_1_5])   J02.HPT - 1.5
                 OTHERS                                  J02.HPT                               --Normal operation of HPT sensor
        LPT   =  ZONE (ActFail[Fail_of_LPT_min_val])     PRE_DATA_VALUE[PRE_TM_LPT, MIN_RANGE] --Failure of LPT sensor: Minimum value
                 ZONE (ActFail[Fail_of_LPT_max_val])     PRE_DATA_VALUE[PRE_TM_LPT, MAX_RANGE] --Failure of LPT sensor: Maximum value
                 ZONE (ActFail[Fail_of_LPT_last_val])    LPT_last                        --Failure of LPT sensor: Last value
                 OTHERS                                  J02.LPT                          --Normal operation of LPT sensor

        Ttank =  ZONE (ActFail[Fail_of_Ttank_min_val])   PRE_DATA_VALUE[PRE_TM_Ttank, MIN_RANGE] --Failure of Ttank sensor: Minimum value
                 ZONE (ActFail[Fail_of_Ttank_max_val])   PRE_DATA_VALUE[PRE_TM_Ttank, MAX_RANGE] --Failure of Ttank sensor: Maximum value
                 ZONE (ActFail[Fail_of_Ttank_last_val])  Ttank_last                          --Failure of Ttank sensor: Last value
                 OTHERS                                 J02.Ttank                            --Normal operation of Ttank sensor
        Tplenum= ZONE (ActFail[Fail_of_Tplenum_min_val])  PRE_DATA_VALUE[PRE_TM_Tplenum, MIN_RANGE] --Failure of Tplenum sensor: Minimum value
                 ZONE (ActFail[Fail_of_Tplenum_max_val])  PRE_DATA_VALUE[PRE_TM_Tplenum, MAX_RANGE] --Failure of Tplenum sensor: Maximum value
                 ZONE (ActFail[Fail_of_Tplenum_last_val]) Tplenum_last                          --Failure of Tplenum sensor: Last value
                 OTHERS                                  J02.Tplenum                           --Normal operation of Tplenum sensor
        V1_closed_pos = ZONE (ActFail[Fail_V1_only_closes_till_01_pc])  1.e-2
                        ZONE (ActFail[Fail_V1_only_closes_till_05_pc])  5.e-2
```

```
                  ZONE (ActFail[Fail_V1_only_closes_till_10_pc]) 10.e-2
                  ZONE (ActFail[Fail_V1_only_closes_till_50_pc]) 50.e-2
                  OTHERS                                        0         --Normal value
   V2_closed_pos = ZONE (ActFail[Fail_V2_only_closes_till_01_pc])  1.e-2
                  ZONE (ActFail[Fail_V2_only_closes_till_05_pc])  5.e-2
                  ZONE (ActFail[Fail_V2_only_closes_till_10_pc]) 10.e-2
                  ZONE (ActFail[Fail_V2_only_closes_till_50_pc]) 50.e-2
                  OTHERS                                        0         --Normal value
   V3_closed_pos = ZONE (ActFail[Fail_V3_only_closes_till_01_pc])  1.e-2
                  ZONE (ActFail[Fail_V3_only_closes_till_05_pc])  5.e-2
                  ZONE (ActFail[Fail_V3_only_closes_till_10_pc]) 10.e-2
                  ZONE (ActFail[Fail_V3_only_closes_till_50_pc]) 50.e-2
                  OTHERS                                        0         --Normal value
   V4_closed_pos = ZONE (ActFail[Fail_V4_only_closes_till_01_pc])  1.e-2
                  ZONE (ActFail[Fail_V4_only_closes_till_05_pc])  5.e-2
                  ZONE (ActFail[Fail_V4_only_closes_till_10_pc]) 10.e-2
                  ZONE (ActFail[Fail_V4_only_closes_till_50_pc]) 50.e-2
                  OTHERS                                        0         --Normal value
   J05.V1_signal = ZONE(VVB[1]==1 AND BBP==1) 1.
                  ZONE(VVB[1]==0 AND BBP==1) V1_closed_pos
                  OTHERS 0
   J05.V2_signal = ZONE(VVB[2]==1 AND BBP==1) 1
                  ZONE(VVB[2]==0 AND BBP==1) V2_closed_pos
                  OTHERS 0
   J05.V3_signal = ZONE(VVB[1]==1 AND BBP==2) 1.
                  ZONE(VVB[1]==0 AND BBP==2)  V3_closed_pos
                  OTHERS 0
   J05.V4_signal = ZONE(VVB[2]==1 AND BBP==2) 1
                  ZONE(VVB[2]==0 AND BBP==2) V4_closed_pos
                  OTHERS 0
   J05.Heater_Nom = ZONE(HACT==1 AND HP==1 AND NOT(ActFail[Fail_BPRU_Nominal_Heater])) HeaterPower
                   OTHERS 0.
   J05.Heater_Red = ZONE(HACT==1 AND HP==2 AND NOT(ActFail[Fail_BPRU_Redundant_Heater])) HeaterPower
                   OTHERS 0.
   Elec.power =     ZONE (PowerON) 6
                   OTHERS 0
END COMPONENT
```

```
------------------------------------------------------------
-- Abstract component to represent a volume with a variable
-- number of fluid ports
------------------------------------------------------------
ABSTRACT COMPONENT Volume
        (
        INTEGER nf_in = 1            "Number of fluid inlets",
        INTEGER nf_out = 1           "Number of fluid outlets",
        BOOLEAN Wall_Energy = TRUE  "Flag to consider or not the energy equation of the wall"
        )

    PORTS
        IN  Fluid f_in[nf_in]
        OUT Fluid f_out[nf_out]
        IN   thermal  tp_in

    DATA
        ENUM FluidName fluid = Xe      "Working fluid"
        REAL V =       0.001      "Volume (m^3)"
........................................................
........................................................
........................................................
    TOPOLOGY
        PATH f_in TO f_out
    INIT
        ier1 = 0
        ier2 = 0
        ASSERT (po == 0. OR Mo ==0.) FATAL \
            "po & Mo cannot be used simultaneously to initialize a Volumen"
        IF (Mo > 0) THEN
            --Initialisation of dynamic variables with mass and temperature
            M = Mo
            rho = Mo / V
            U = M * Fl_u_vs_rhoT(FluidCode[fluid], rho, To, ipx, ipp, ier2)
        ELSE
            --Initialisation of dynamic variables with pressure and temperature
            rho =   Fl_rho_vs_pt(FluidCode[fluid], po, To,ipx, ipp, ier2)
            u =     Fl_u_vs_pt(FluidCode[fluid], po, To, ipx, ipp, ier2)
            M = V * rho
            U = M * u
        END IF
        T_wall = To
    CONTINUOUS
```



```
        D_tank = sqrt(4 * A_cross / MATH.PI)
        h_film = (cond / D_tank) *  Nusselt_function(Hfilm_tank, D_tank, cond, visc,\
                     0.8, SUM(i IN 1, nf_in;f_in[i].m), SUM(i IN 1, nf_out;f_out[i].m),
                     rho, T, T_wall)
        q_wet  = h_film * A_wall * (T_wall-T)
        EXPAND (Wall_Energy)
            T_wall'= (tp_in.q - q_wet) / cp_wall / m_wall
        EXPAND_BLOCK (NOT Wall_Energy)
            T_wall = T_wall_o
        END EXPAND_BLOCK
    --  Ecuaciones de conservacion del fluido
        M' = SUM (i IN 1, nf_in; f_in[i].m)  - SUM(i IN 1, nf_out; f_out[i].m)
        U' = SUM (i IN 1, nf_in; f_in[i].m * f_in[i].ht) - \
                   SUM(i IN 1, nf_out;f_out[i].m * f_out[i].ht) + q_wet
        rho = M / V
        u   = U / M
   --   Ecuaciones de estado
        Fl_state_vs_rhou(FluidCode[fluid], rho, u, p, T, x, cond, visc, ipx, ipp, ier1)
        EXPAND (i IN 1, nf_in)
            f_in[i].p  = p
        EXPAND_BLOCK (i IN 1, nf_out)
            f_out[i].p  = p
            f_out[i].ht = u + p/rho
        END EXPAND_BLOCK
    --  Thermal port
        tp_in.T = T_wall
END COMPONENT


-----------------------------------------------------------
-- Volume with 1 inlet & 2 outlets
-----------------------------------------------------------
COMPONENT Volume3 IS_A Volume
    DECLS
        CLOSE nf_in = 1
        CLOSE nf_out = 2
END COMPONENT
```



```
COMPONENT Fail_Processor
    PORTS
        IN RAMS rams
    DECLS
        INTEGER icode
    DISCRETE
        WHEN (rams.event != 0) THEN
            IF (rams.event > 0 ) THEN
                IF (rams.event >= 1 AND rams.event <= NumFails) THEN
                    IF (NOT ActFail[rams.event]) THEN
                        --Disactivate previous failures of the same group
                        FOR (j IN 1,NumFails EXCEPT rams.event)
                            IF (ActFail[j] == TRUE AND \
                                FailGroup[j] == FailGroup[rams.event]) THEN
                                icode = j
                                ActFail[j] = FALSE
                            END IF
                        END FOR


                        ActFail[rams.event] = TRUE
                    END IF
                END IF
            ELSE
                IF (-rams.event >=1 AND -rams.event <= NumFails) THEN
                     --Disactivate previous failure with the same code
                    IF (ActFail[-rams.event]) THEN
                        icode = - rams.event
                        ActFail[icode] = FALSE
                    END IF
                END IF
            END IF
            rams.event = 0
       END WHEN
END COMPONENT
```