Investigating the effects of designing industrial control software using push and poll strategies

Investigating the Effects of Designing Industrial Control Software using Push and Poll Strategies

J.F. Groote[1], A.A.H. Osaiweran[1], M.T.W. Schuts[2], and J.H. Wesselius[2]
[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Philips Healthcare, BU Interventional X-ray, Best, The Netherlands
{j.f.groote, a.a.h.osaiweran}@tue.nl, {mathijs.schuts, jacco.wesselius}@philips.com

Abstract

In this paper we apply a number of design guidelines for circumventing the state space explosion problem from [J.F. Groote, T.W.D.M. Kouters, and A.A.H. Osaiweran, Specification guidelines to avoid the state space explosion problem, 2011] to the design and formal verification of a real industrial case, namely a controller of a power distribution unit of X-ray machines developed at Philips Healthcare. Through this work we investigate whether these guidelines are effective in designing practical applications. We provide a number of alternative designs that mainly incorporate pushing and polling strategies, taking into account a number of these guidelines. Using the pushing strategy components notify one another when information becomes available while using polling components ask for information only when it is needed. We find that designs that use a pushing strategy and do not apply such guidelines typically lead to the generation of substantially more states. All demonstrated designs formally refine a single predefined external specification that captures the desired external behavior of the system. Moreover, all designs are deadlock free and do not exhibit any illegal interactions. This confirms our hypothesis that the design guidelines are really effective in practical contexts.

1 Introduction

Due to the increasing complexity of industrial control software, establishing the behavioral correctness is a challenging task. During the construction of large industrial software systems, errors are regarded as inevitable and some are often hard to analyze or even to reproduce, due to the concurrent nature of interacting components. Hence, techniques for automatic detection of flaws are widely encouraged, to assist developers building their software rapidly and correctly.

Behavioral verification tools, mainly using model checking technology, can be used for verifying the discrete behavior of complex industrial software designs [13, 3, 4]. They assist correctness verification of designs of complex systems prior to their actual implementation. In a number of reported industrial cases [12, 16], design errors have been discovered, which were hard to find using conventional testing, due to the concurrent nature of the components. Since model checking tools provide high-level automation compared to other verification techniques such as theorem proving, they quickly become more popular and attractive in industry.

Model checking tools require a model or specification that precisely describes the behavior of concurrent components to be verified. The model can thoroughly be investigated to prove that the components always satisfy certain requirements. The tools perform enumerative, systematic exploration of all (or part of) possible execution scenarios of the molded system. The set of the execution scenarios are often characterized by an LTS (label transition system or state space) which contains states and transitions labeled by actions performed by the components.

But the behavioral verification is limited by the state space explosion problem, which arises when the verified components include a huge number of states that cannot fit into memory, de-

spite the use of clever verification algorithms and powerful computers. Although model checking technologies nowadays available can potentially handle billions of states, they still suffer from this problem. For some practical cases developers have to wait hours or days for outcomes resulting from the tools when verifying even a single property of their systems.

In [7, 8] we have proposed a number of guidelines to tackle the state space explosion problem but in a different manner, namely by designing software components such that they can be easily verified. In this paper we apply a number of these guidelines on the design and the formal verification of a practical industrial case, namely a controller of a power distribution unit (PDU) [10, 11, 9], used for controlling the electrical power, of X-ray machines developed at Philips Healthcare. Through this we want to know whether the guidelines are effective in practical context. To accomplish this, we propose a number of alternative designs to achieve the required functionality of the controller. We found that the designs that do not use the guidelines have substantially more states.

We start by describing a single desired external behavior of the controller. Then, we provide two main designs, where the first uses a pushing strategy and the second uses a polling strategy. By pushing we mean that components of a system share their information with others when the information is available, while polling means that components poll (or ask) information from others only when it is needed. As will be demonstrated shortly, other guidelines such as the restrict use of data and the use of global synchronous communication have been further applied and substantially helped reducing the state space. All design alternatives refine the external behavior of the controller and provide the intended behavior of the system.

Throughout this article we use mCRL2 [13, 6] for formal specification and state space generation. Hence, we assume a basic knowledge of the description language and the tool set. Additionally, we use the refinement concept to prove formal refinement of designs against the external behavior. For this we use mCRL2, CADP [3] and CSP/FDR2 [15, 4].

The results of this work confirms that different design styles can reduce the number of the generated states of the modeled systems and that the guidelines are effective in practical applications.

This paper is organized as follows. In Section 2 we bring a list of guidelines used for designing and verifying the PDU controller from [7, 8]. Section 3 gives an overview of the context of the PDU controller. The strategies and tactics used to accomplish the tasks of modeling and verifying the controller are described in Section 4. The external behavior of the controller is detailed in Section 5. The designs of the controller using the pushing strategy are demonstrated in Section 6, while the designs implementing the poll strategy are described in Section 7. In Section 8 we give some statistical data, comparing the push and poll variants and the used tools.

2 Overview of the used guidelines

In this section we give a concise description of the guidelines [7, 8] that we used in this paper.

1. **Information polling**. This guideline advises to let processes ask for information, whenever it is required. The alternative is to share information with other components, whenever the information becomes available. Although, this latter strategy clearly increases the number of states of a system, it appears to prevail over information polling in most specifications that we have seen.

2. **Global synchronous communication**. If more parties communicate with each other, it can be that a component 1 communicates with a component 2, and subsequently, component 2 informs a component 3. This requires two consecutive communications and therefore two state transitions. By using multi-actions it is possible to let component 1 communicate with component 2 that synchronously communicates with a component 3. This only requires one transition. By synchronizing communication over different components, the number of states of the overall system can substantially be reduced.

3. **Avoid parallelism among components**. If components operate in parallel, the state space grows exponentially in the number of components. By sequentializing the behavior

of these components, the size of the total state space is only the sum of the sizes of the state spaces of the individual components. In this latter case state spaces are small and easy to analyze, whereas in the former case analysis might be quite hard. Sequentializing the behavior can for instance be done by introducing an arbiter, or by letting a process higher up in the process hierarchy to allow only one sub-process to operate at any time.

4. **Restrict the use of data**. The use of data in a specification is a main cause for state-space explosion. Therefore, it is advisable to avoid using data whenever possible. If data is essential, try to categorize it, and only store the categories. For example, instead of storing a height in millimeters, store *too_low*, *right_height* and *too_high*. Finally, take care that data is only stored in one way. E.g., storing the names of the files that are open in an unordered buffer is a waste. The buffer can be ordered without losing information.

5. **Specify the external behavior of sets of sub-components**. If the behavior of sets of components are composed, the external behavior tends to be overly complex. In particular the state space is often larger than needed. A technique to keep this behavior small is to separately specify the expected external behavior first. Subsequently, the behaviors of the components are designed such that they meet this external behavior.

The following two guidelines are not used in this work, but it is worth mentioning them here for the sake of completeness.

1. **Confluence and determinacy**. When parallel behavior cannot be avoided, it is useful to model such that the behavior is τ-confluent. In this case τ-prioritisation can be applied when generating the state space, substantially reducing the size of the state space. Modeling a system such that it is τ-confluent is not easy. A good strategy is to strive for determinacy of behavior. This means that the 'output' behavior of a system must completely be determined by the 'input'. This is guaranteed whenever an internal action (e.g. receiving or sending a message from/to another component) can be done in a state of a single component, then no other action can be done in that state.

2. **Compositional design and reduction**. If a system is composed out of more components, it can be fruitful to combine them in a stepwise manner, and reduce each set of composed components using an appropriate behavioral equivalence. This works well if the composed components do not have different interfaces that communicate via not yet composed components. So typically, this method does not work when the components communicate in a ring topology, but it works very nicely when the components are organized as a tree.

3 The context of the PDU controller

We start by illustrating the context of the PDU controller. Philips healthcare, at Best, the Netherlands is developing a family of highly sophisticated, computerized X-Ray systems. The systems include a distributed architecture in the sense that clinical applications, required for establishing X-Ray examinations, are deployed on a cluster of PCs and devices. These components require an efficient and reliable source of power control.

In order to efficiently control the flow of power and to systematically start-up and shutdown the PCs and the devices in an orderly fashion, the system utilizes a Power Distribution Unit (PDU), see Figure 1. All PCs and devices are attached to the PDU. The clinical user has no means of powering on/off the components of the system separately without using the PDU. The clinical user can only initiate start-up and shutdown requests by pushing a number of buttons on a user console attached to the PDU. Upon pressing these buttons the PDU controls the flow of power to the components.

The PDU console provides two buttons: *PowerOn*, and *PowerOff*. The PDU includes an optional *EmergencyOff* button which can be used to cut down any source of power to the system,

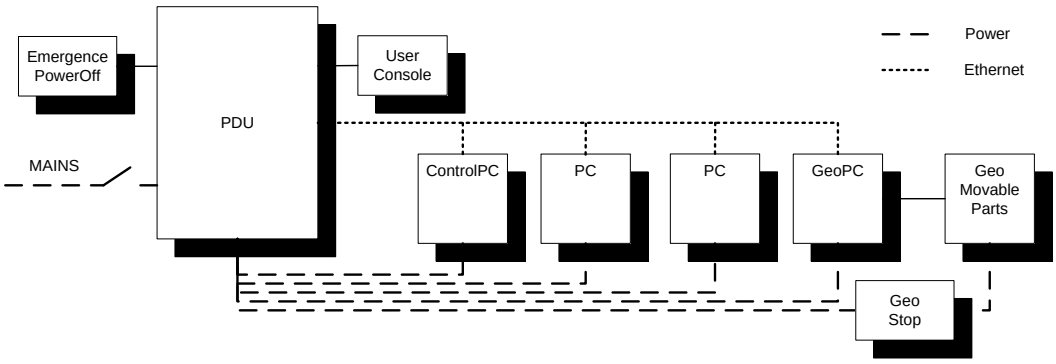

Figure 1: Power, network, and device distribution

in case of calamities. This optional button may be installed on request or when mandatory by legislation.

To manage the start-up and shutdown behavior of the system, the PDU employs two networks. The first network is the power network which supplies the attached PCs and devices with the necessary power. The flow of power to the components is controlled by the PDU by switching a number of power taps on and off. The second network is an Ethernet network, by which the PDU can communicate with the PCs and the devices, through a number of dedicated signals.

The PDU taps are classified into switchable and permanent taps. The switchable taps can be switched on/off by the PDU. The permanent taps are powered when the system is off in the perception of the clinical user, but not in the perception of the PDU (the system is in standby from perspective of the PDU). This allows the attached components to be available for batch processing, maintenance and remote accesses purposes. The permanent taps can be switched off when forced by the clinical users (e.g., by pressing the *EmergencyOff* button).

PCs and devices The PCs and devices depicted in Figure 1 almost expose the same start-up and shutdown behavior, but there is a small difference between the GeoPC (Geometrical PC) and the ControlPC from others, see the state machines in Figure 2.

Initially, a PC is in the *Off* state. When it is supplied with power, it transits to the *StartingUp* state where the Operating System (OS) boots up and then the clinical applications are started. After the OS and the applications are up-and-running, the PC transits to the *Operational* state.

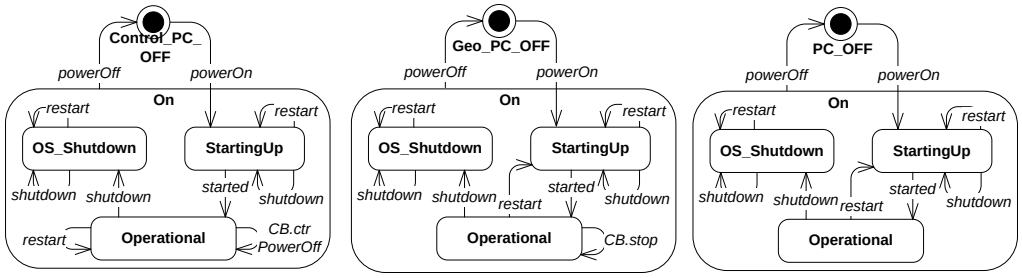

Figure 2: The external behavior of all PCs

The applications of a PC are restarted upon receiving a *restart* message from the PDU in *Operational* state. Additionally, when a *shutdown* message is received from the PDU, the PC

stops all running applications and shuts down the OS.

The GeoPC and the ControlPC include additional behavior. The main function of the GeoPC is controlling a number of motorized movable segments such as the table where patients lay on and the stands holding X-Ray generators and detectors. On multiple places, the system is equipped with *Stop* buttons which can be pressed by the users to stop any motorized movement in case of dangerous situations. Upon pressing these buttons the GeoPC sends a *CB.stop* signal requesting the PDU to switch off the taps to the movable segments. This is visualized in the state machine of GeoPC in Figure 2.

The ControlPC can send a *CB.controlPowerOff* signal, demanding and forcing the PDU to systematically power off the entire system, including the ControlPC itself. The ControlPC is attached to a permanent tap while all other PCs are connected to switchable taps.

Behavior of the Power Distribution Unit The PDU includes a controller that implements the state machine of Figure 3. We assume that the PDU, all PCs and devices are well functioning; therefore, all error scenarios and recovery operations are excluded from the state machine.

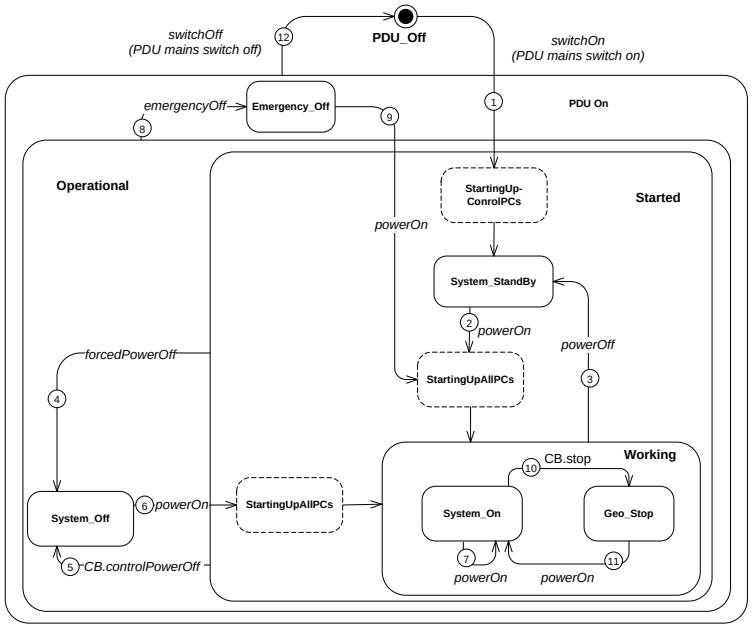

Figure 3: The high-level behaviour of the PDU [11]

The state machine in Figure 3 distinguishes the following six stable states as described in Table 1, and the two transiting stated as described in Table 2.

The *PowerOn*, *PowerOff*, and *EmergencyOff* buttons on the user console plus the *Stop* button of the movable segments, leads to commands that are processed by the PDU controller. Depending on the current state and the supplied command, the controller sends messages to the PCs over the Ethernet network and/or switches the taps on and off if required.

The state machine includes eight distinct events in total. The *PDUswitchOn* and *PDUswitchOff* indicate switching the mains disconnector switch on and off, respectively. The *powerOff* event indicates that the user presses the *PowerOff* button for less than 3 seconds, while the *forcedPowerOff* event denotes that the user presses the same button more than 10 seconds. Both *powerOn* and *emergencyOff* represent pressing the *PowerOn* and *EmergencyOff* buttons, respectively. *ControlPowerOff* and *stop* events indicate receiving callback signals from both the ControlPC and the GeoPC, where the first requests the PDU to power off the complete system and the second demands the PDU to immediately cut down the power to the movable segments.

State	Property
PDU_Off	The mains disconnector switch is open which means that the PDU is powerless. All PCs and devices are off.
System_Standby	The permanent power taps are powered. All switchable taps are powerless. ControlPC is on.
System_On	All permanent and switchable taps are powered. All PCs and devices are on.
GEO_Stop	Similar to the *System_On* state, only the movable parts are powerless. All PCs and devices are on. Motorized movements are disabled.
System_Off	All permanent and switchable taps are powerless. All PCs and devices are off.
Emergency_Off	All permanent and switchable taps are powerless. All PCs and devices are off.

Table 1: The stable states of the PDU state machine [11]

State	Property
StartingUpControlPC	The permanent power taps are powered. The ControlPC is starting up.
StartingUpAllPCs	All permanent and switchable taps are powered. Not all PCs or devices are fully operational.

Table 2: The transitioning states of the PDU state machine [11]

Table 3 summarizes the required tasks for each transition of the state machine. For example, when the system is in the *System_Off* state and the user presses the *PowerOn* button, all permanent and switchable taps are switched on, and therefore all PCs and devices start-up. Eventually, all PCs and devices are started-up and the system can potentially move to the *System_On* state.

In the *System_On* state, if the user again presses the *PowerOn* button, the PDU broadcasts a *restart* message over the Ethernet network. Consequently, the PCs and devices shall restart their applications. But, if the user presses the *PowerOff* button for less than 3 seconds, the PDU broadcasts a *shutdown* message over the Ethernet network. Upon receiving the message by the PCs, they gradually shutdown their applications and then their OS. When all PCs and devices are shutdown, the taps will be made powerless by the PDU.

Beside the above mentioned events we introduce a number of indication callback events that reflect the status (or modes) of the system:

- the *startingUp* event informs external users that the system is in the process of starting up its components,
- the *systemStandby* event notifies the user that the system is in the *System_StandBy* state,
- the *off* event tells the users that the entire system is off,
- the *systemOn* event informs the user that the system is up-and-running and fully operational,
- and the *geoStop* event indicates the user that all motorized movements are disabled.

4 Strategy and tactics

The conceptual structure of the specification of the PDU controller is depicted in Figure 4. The external behavior of the PDU and the PCs are depicted as ovals. The design of the PDU controller

Transition	Activity
1	Boot PDU; the PDU switches on all permanent power taps; the ControlPC is starting up.
2	The PDU switches on all switchable taps, one by one to avoid a big inrush current; all devices are starting up.
3	The PDU broadcasts a "shutdown" message to shutdown all control devices except the ControlPC; the PDU switches off all switchable taps when power load is below a threshold or when the timer expires.
4	The PDU immediately switches off all power taps.
5	The PDU broadcasts a "shutdown" message to shutdown all control devices including the ControlPC; the PDU switches off all taps when power load is below threshold or when the timer expires.
6	The PDU switches on all taps, one by one to avoid a big inrush current; all devices are starting up.
7	The PDU broadcasts a "restart" message; the applications of all control devices are restarted.
8	Disconnect the PDU internal power bus.
9	The PDU switches on all taps, one by one to avoid a big inrush current; all devices are starting up.
10	The PDU switches off the power taps that supply motor drives of movable parts.
11	The PDU switch on the power taps that supply motor drives of movable parts.
12	The PDU is switched off; all taps are switched off.

Table 3: The activities required for each transition of the PDU state machine [11]

is shown as a square shape. The communication channels with the direction of information flow are depicted using arrows.

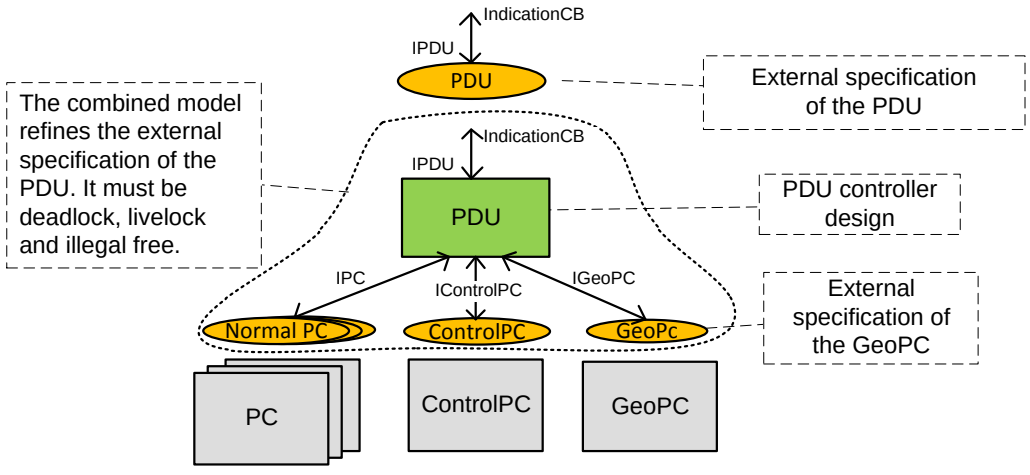

Figure 4: Conceptual structure of the specification of the PDU controller

The figure shows the structure of a combined model that includes the parallel composition of the PDU controller and the external specification of the PCs, highlighting the communication channels used for exchanging information among the components. Each design alternative of the PDU controller has a different combined model. To construct these models we have followed a number of steps, summarized below.

Modeling the external behavior of the PDU First, we modeled the desired external behavior of the PDU with respect to the external users of the system. This specification includes all external commands issued by the user console plus all indication callback signals sent to the user. The specification is identical for all design alternatives, and is used as a guide for implementing the alternative designs we are comparing. This external behavior excludes any internal interaction with the PCs.

Describing the external behavior of the PCs The external behavior of each PC is described with respect to the interaction required with the PDU. The description excludes any activities performed internally by the PC.

Constructing alternative designs for the PDU controller We design the PDU controller in two manners, namely a design where PCs 'push' their information to the PDU when the information is available, and another design where the PDU 'polls' information from the PCs whenever it is required.

For each design manner there are a number of alternatives that assist further reducing the state space. All design alternatives adhere to the external specification, and provide the external users of the system with the expected behavior.

Modeling conventions In the specification of all models any action pre-fixed by the letter 'r' denotes the receiving party of a communication whereas actions pre-fixed by 's' denote the sending party. The result of a communication is denoted by an action without any pre-fixed letter.

Specification completeness In every state of the external behavior of the PCs we assign illegal responses to the stimuli if they are not expected in a state. The same response is assigned to callbacks received from the PCs in the specification of the PDU design for detecting unexpected callbacks. During the behavioral verification we search for the occurrences of such an event plus deadlock and livelock scenarios.

Refining the external behavior Each design alternative is checked against the external specification using a number of refinement models: weak-trace [2], Failures [15], Failures-divergences [15], observational [14], safety [1], Tau* [5] and branching-bisimulation [17]. The reason of choosing refinement over equivalence check is that checking equivalence may tend to be overly complex. It may require that both the implementation and the external behavior to strictly have the same structure, so the external specification might be forced to be adjusted to satisfy the structure of the design. This is what we are trying to avoid here.

Instead of using equivalence checks we prove refinement of designs by means of inclusion (or preorder) checks. Precisely, we prove that the behavior of a design is included in the behavior of the external specification. Upon the success of the check we know that the design always exposes expected behavior to the external world under the refinement model being used, i.e., no extra unexpected behavior would result from the concrete implementation of the design crossing the external boundary.

We believe that specifying the external behavior of a system prior to its implementation assists constructing the system better, but does not guarantee building the internal behavior of the system correctly. Checking correctness of internal behavior of systems can be accomplished by other means such as searching for deadlocks, livelocks, illegal interactions and verifying properties on systems.

The details of the steps performed throughout this case are addressed in the subsequent sections.

5 The external specification of the PDU controller

We started our modeling activities by considering the fifth guideline. The external specification of the PDU controller in the mCRL2 language is listed below. It precisely describes the external

behavior of the PDU, with respect to the external users, reflecting the internal modes of the system using states and visible indication callbacks, matching the state machine of Figure 3. It includes all user commands as input stimuli, and all user indication callbacks as responses to the external world. It excludes all internal interactions such as internal system messages and powering on/off the PCs.

```
proc ExtSpec(s:State)=
    (s==PDU_Off) ->
        ( IPDU(PDUswitchOn) . IndicationCB(startingUp) . ExtSpec(StartingUpCrPC) ) +
    (s==System_StandBy) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn). IndicationCB(startingUp) . ExtSpec(StartingUpAllPCs) +
          IPDU(powerOff) . ExtSpec(System_StandBy) +
          IPDU(forcedPowerOff) . IndicationCB(off) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . IndicationCB(off) . ExtSpec(Emergency_Off) +
          int . IndicationCB(off).ExtSpec(System_Off) ) +
    (s==System_On) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . IndicationCB(startingUp) . ExtSpec(StartingUpAllPCs) +
          IPDU(powerOff) . IndicationCB(systemStandby) . ExtSpec(System_StandBy) +
          IPDU(forcedPowerOff) . IndicationCB(off) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . IndicationCB(off) . ExtSpec(Emergency_Off) +
          int . IndicationCB(off) . ExtSpec(System_Off) +
          int . IndicationCB(geoStop) . ExtSpec(Geo_Stop) ) +
    (s==StartingUpAllPCs) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . ExtSpec(StartingUpAllPCs) +
          IPDU(powerOff) . IndicationCB(systemStandby) . ExtSpec(System_StandBy) +
          IPDU(powerOff) . ExtSpec(StartingUpCrPC) +
          IPDU(forcedPowerOff) . IndicationCB(off) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . IndicationCB(off) . ExtSpec(Emergency_Off) +
          int . IndicationCB(off) . ExtSpec(System_Off) +
          int . IndicationCB(systemOn) . ExtSpec(System_On) +
          int . ExtSpec(StartingUpAllPCs) +
          int . IndicationCB(geoStop) . ExtSpec(Geo_Stop) ) +
    (s==Geo_Stop) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . IndicationCB(systemOn) . ExtSpec(System_On) +
          IPDU(powerOff).IndicationCB(systemStandby).ExtSpec(System_StandBy) +
          IPDU(forcedPowerOff) . IndicationCB(off) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . IndicationCB(off) . ExtSpec(Emergency_Off) +
          int . IndicationCB(off) . ExtSpec(System_Off) ) +
    (s==System_Off) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . IndicationCB(startingUp) . ExtSpec(StartingUpAllPCs)+
          IPDU(powerOff) . ExtSpec(System_Off) +
          IPDU(forcedPowerOff) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . ExtSpec(Emergency_Off) ) +
    (s==Emergency_Off) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . IndicationCB(startingUp) . ExtSpec(StartingUpAllPCs) +
          IPDU(powerOff) . ExtSpec(Emergency_Off) +
          IPDU(forcedPowerOff) . ExtSpec(Emergency_Off) +
          IPDU(emergencyOff) . ExtSpec(Emergency_Off) ) +
    (s==StartingUpCrPC) ->
        ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
          IPDU(powerOn) . ExtSpec(StartingUpCrPC) +
          IPDU(powerOff) . ExtSpec(StartingUpCrPC) +
          IPDU(forcedPowerOff) . IndicationCB(off) . ExtSpec(System_Off) +
          IPDU(emergencyOff) . IndicationCB(off) . ExtSpec(Emergency_Off) +
          int . IndicationCB(systemStandby) . ExtSpec(System_StandBy) ) ;
```

To briefly explain the model we choose the *System_On* state as an example. The state includes seven summands in total. It precisely describes that when the PDU is in the *System_On* state, it can receive any external command from the users. This is indicated by the first five summands. Upon receiving an external command the PDU may send indication callback signals and then transits to a next state. For example, when the PDU receives the *powerOff* command, it sends the *systemStandby* indication to the external users and then transits to the *System_StandBy* state.

The last two summands of the state represent the cases where external users can receive indications that the system is off or transiting to the *Geo_Stop* state, due to some internal interactions with the PDU. Both *int* events represent detailed activities performed by the concrete implementation of the PDU. For example, *int.IndicationCB(off)* represents the following internal activities:

1. The user of the ControlPC has requested the PDU to power off the entire system via the internal *controlPowerOff* callback event.

2. The PDU treats the signal by sending the *shutdown* message around to all devices.
3. The PDU switches all taps off.
4. The PDU sends the *IndicationCB(off)* signal to the external world.
5. The PDU transits to the *System_Off* state.

The same technique had been applied to all states of the PDU, matching the original state machine of Figure 3. The complete specification of the model is listed in A.

When the specification of the model was completed, it was checked for absence of deadlocks and livelocks. The corresponding LTS had been generated, and used at later stages for the refinement check against the concrete designs of the PDU using mCRL2 and CADP.

6 Implementing the PDU controller using the push strategy

In this variant, the design of the controller utilizes a pushing strategy, in the sense that all PCs share information with the PDU controller upon changes in their internal states. This is illustrated in the sequence diagram in Figure 5. For instance, when the PDU is in the *System_On* state and the *Stop* button is pressed, the GeoPC notifies the PDU controller by sending the *stop* callback event. The same applies to the *controlPowerOff* callback from the ControlPC. Furthermore, when the PCs are powered on by the PDU, the PDU waits for callbacks from the PCs indicating that they are ready and fully operational.

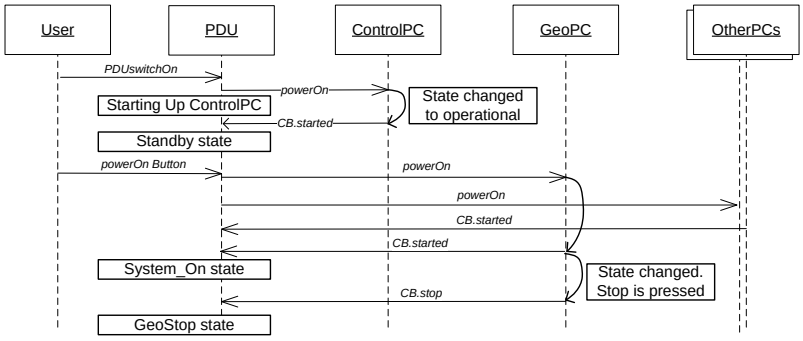

Figure 5: Example of a scenario where pushing is used

6.1 The external behavior of the PCs

In this section we introduce the external specification of the ControlPC that describes the external behavior with respect to the PDU controller. Similarly, the specification of the remaining PCs is straightforward and therefore omitted from the text, but it is available in B. The specification of the PCs are identical for all push design variants.

The specification of the ControlPC is straightforward. It includes five states. In any state the ControlPC can receive a number of legal and illegal stimuli events.

Note that, when the ControlPC is in the *StartingUp* state, it can send (or push) the callback event *sICR_PC_CB(started)* to the PDU and then transits to the *Operational* state. Similarly, when the ControlPC is in the *Operational* state, it can send the *sICR_PC_CB(controlPowerOff)* callback event to the PDU, as a request to power off the entire system.

```
proc ControlPC(s:PCState) =
(
 (s==PC_Off) ->
 ( rICR_PC(powerOn) . ControlPC(StartingUp) +
   rICR_PC(powerOff) . Illegal . delta +
   rICR_PC_Broadcast(restart) . Illegal . delta +
   rICR_PC_Broadcast(shutdown) . Illegal . delta) +
 (s==Operational) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   sICR_PC_CB(controlPowerOff) . ControlPC(WaitingShutdown) +
   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown))+
 (s==WaitingShutdown) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
 )+
 (s==OS_Shutdown) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   rICR_PC_Broadcast(restart) . Illegal . delta +
   rICR_PC_Broadcast(shutdown) . Illegal . delta )+
 (s==StartingUp) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   rICR_PC_Broadcast(restart) . Illegal . delta +
   rICR_PC_Broadcast(shutdown) . Illegal . delta +
   sICR_PC_CB(started) . ControlPC(Operational)
 ));
```

6.2 The design of the PDU controller

There are mainly four alternative models for the PDU designs that incorporate the push strategy. The details of each of them are introduced below.

The asynchronous PDU controller In this variant the PDU controller communicates with the PCs synchronously and sequentially one-by-one, but the PCs communicate with the PDU asynchronously. The PDU includes a queue to store incoming callback events from the PCs.

The first issue we encountered when verifying this variant was the queue size and the large number of interleaving caused by the queue and the external commands. The PCs can quickly send callback events to the queue leading to filling-up a queue of any arbitrary size. External commands can arrive while there are still unprocessed callbacks in the queue, hence verification was initially not doable.

Therefore, we had to limit the behavior of the PCs such that having more than one similar callback at a time in the queue is prohibited. Furthermore, we give any callback event a priority to be processed by the PDU over any external user command, so the queue has to be emptied first.

Below we introduce a part of the design specification, demonstrating only the *PDU_Off* stable state and the *StartingUp_CR_PC* transiting state. The entire specification can be found in Appendix B.

```
proc PDU_State_Machine(s:PDUState,geopcOn,crpcOn,geoPressed:Bool,startedPc:Nat) =
(
 (s==PDU_Off) ->
 ( rIPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
                        PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
   rICR_PC_CB(controlPowerOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rIGeoPC_CB(stop) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rICR_PC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rIGeoPC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   sum id:Pos . rIPC_CB(id,started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc)
 )+
 (s==StartingUp_CR_PC) ->
 ( rIPDU(PDUswitchOff) . sICR_PC(powerOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rIPDU(powerOn) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
   rIPDU(powerOff) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                                  PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rIPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
                                  PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
   rICR_PC_CB(controlPowerOff) . Illegal . delta +
   rICR_PC_CB(started) . IndicationCB(systemStandby) .
                                  PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
   rIGeoPC_CB(stop) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
   rIGeoPC_CB(started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
   sum id:Pos . rIPC_CB(id,started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc)
 )+

.... (the specification of the rest of states follows)
);
```

The first summand of the *PDU_Off* state specifies that when the PDU is switched on, it powers on the ControlPC, sends an indication to the user that the system is starting-up and transits to

the *StartingUp_CR_PC* state. The other remaining summands of the state specify that any (late) callback events received in the state are consumed.

When the PDU is in the *StartingUp_CR_PC*, it can process a number of external commands and internal callbacks. All callbacks are ignored except those originating from the ControlPC. When the PDU knows that the ControlPC is ready and operational by receiving the *rICR_PC_CB(started)* from the queue, it sends an indication to the user that the system is in the *SystemStandby* state. Receiving a request to power off the system from the ControlPC is illegal since the ControlPC has to start first, see the summand that corresponds to the *rICR_PC_CB(controlPowerOff)* stimulus event.

The specification of the PDU controller includes a number of data parameters used for remembering the status of the PCs. For example, the *startedPc* data parameter is of type natural number and is used to count how many normal PCs have started.

The asynchronous PDU controller with global synchronous communication The model of this variant is almost identical to the previous model, except that the fourth guideline is used. We noticed that powering on/off the PCs can be modeled using multi-actions. That is, instead of modeling this behavior by sending the *powerOn* or *powerOff* events to the PCs sequentially, all PCs engage into one big action, denoting that the event occurs at the same time for all PCs.

To clarify the concept, consider the following examples. The following Handler process communicates with the PDU (via the *rcommandhandler* and srelease actions) and the PCs (via the *sIPC* action), where all communications are done sequentially until completion. This process is used in the specification of the asynchronous push model addressed earlier.

```
proc Handler =
    sum c:Command . rcommandhandler(c) | sIPC(1,c) .
                                    sIPC(2,c) . sIPC(3,c) .sIPC(4,c) . srelease | sIPC(5,c) .  Handler
```

Obviously, this process results in five successive states, with the possibility of interleaving with other processes.

On the other hand, the following Handler process describes the use of multi-actions, used for this design variant. All communications with PCs are done in one step.

```
proc Handler =
    sum c:Command . rcommandhandler(c) | sIPC(1,c) | sIPC(2,c) | sIPC(3,c) | sIPC(4,c) | sIPC(5,c) . Handler
```

Clearly, this process results in a single state.

Since the number of states are reduced to a single state, the entire state space can also be reduced, taking into account the reduced interleaving. The complete specification of this model is listed in C.

The synchronous PDU controller In this variant all interactions between the PDU and the PCs are synchronous. In contrast with the previous variants, the PDU does not include any queue, and all received callbacks from the PCs are processed synchronously. Still, all PCs inform (or push) the PDU upon the changes of their states, but in a synchronous manner.

The specification of this variant is listed in D. The specification is similar to the asynchronous variant except that the queue placed between the PDU and the PCs is removed.

The synchronous PDU controller with global synchronous communication Here, the model of synchronous PDU controller above is adapted, such that powering on/off PCs is accomplished by multi-actions. The detail of using multi-actions is previously described for the asynchronous controller with global synchronous communication variant, and hence is omitted here. The complete specification of this variant is introduced in E.

7 Implementing the PDU controller using the poll strategy

In this section we present a model that describes the implementation of the PDU controller using a polling strategy. We used the first guideline to accomplish this model. The PDU controller polls the PCs to acquire their states. Figure 6 visualizes an example of polling used for designing the controller.

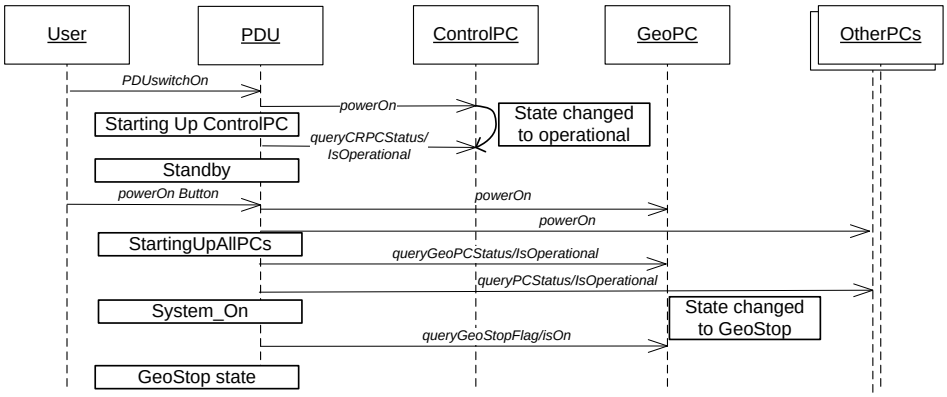

Figure 6: Example of a scenario where polling is used

7.1 The external behavior of the PCs

Before describing the design of the controller, we first need to describe the external behavior of the PCs. Below, a fragment of the mCRL2 specification related to the external behavior of the ControlPC is described. The specification of the GeoPC and the normal PCs are straightforward and almost identical to this specification.

```
proc ControlPC(s:PCState) =
(
 (s==PC_Off) ->
 ( rICR_PC(powerOn) . ControlPC(PC_On) +
   rICR_PC(powerOff) . Illegal . delta +
   rICR_PC(queryCRPCStatus) . Illegal . delta +
   rICR_PC(queryCRPCPowerOffFlag) . Illegal . delta +
   rICR_PC_Broadcast(restart) . Illegal . delta +
   rICR_PC_Broadcast(shutdown) . Illegal . delta
 )  +
 (s==PC_On) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsOperational) . ControlPC(PC_On) +
   rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsNotOperational) . ControlPC(PC_On) +
   rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOn) . ControlPC(PC_On) +
   rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(PC_On)+
   rICR_PC_Broadcast(restart) . ControlPC(PC_On) +
   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
 )  +
 (s==OS_Shutdown) ->
 ( rICR_PC(powerOn) . Illegal . delta +
   rICR_PC(powerOff) . ControlPC(PC_Off) +
   rICR_PC(queryCRPCStatus) . Illegal . delta +
   rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(OS_Shutdown) +
   rICR_PC_Broadcast(restart) . ControlPC(OS_Shutdown) +
   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
 )
);
```

As can be seen from the specification, the ControlPC has three main states:

- *PC_Off*: the PC is off, which means that the tap is switched off;

- *PC_On*: the PC is on, which means that the tap is switched on, but the applications on the PC can be operational or not; and
- *OS_Shutdown*: the tap is on but the OS and the applications are shut down.

Per state it is defined which calls are allowed to be issued by the design of the PDU controller, and which of them are illegal. Every query (or poll) method has a return value that is immediately sent back to the PDU.

When the ControlPC is powered on, it can receive a number of signals by polling. The ControlPC non-deterministically replies to these signals indicating its current state: for example, observe the summands with *queryCRPCStatus* and *queryCRPCPowerOffFlag* calls which non-deterministically return a value in the *PC_On* state.

7.2 The design of the PDU controller

The PDU controller design has to adhere to the external specification of the PDU on the one hand, and to correctly use the specifications of the PCs on the other hand. To implement a polling mechanism, the PDU utilizes internal timers to stimulate the PDU to poll status of the PCs in certain states. As we will see shortly, the fourth guideline is employed to abstract from concrete data values of the timer. For example, we abstract from the progress of timer values in milliseconds by a single event denoting the expiration of the time.

Moreover, the third guideline is used for modeling the start-up behavior of the system. Compared to the the push model the PDU sequentially polls information about the state of the PCs, when it is needed. The PDU does not expect any spontaneous information to be pushed by the PCs. The complete specification of this variant can be found in F.

Below we introduce a fragment of the controller design specification, related to *PDU_Off*, *StartingUpCrPC* and *WaitingCRPCReply* states.

```
proc PDU_State_Machine(s:PDUState,cRPCstarted,geoPCstarted,geoPressed:Bool,state:PDUState) =
(
 (s==PDU_Off) ->
 ( IPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
   PDU_State_Machine(StartingUpCrPC,false,false,false,none)
 ) +
 (s==StartingUpCrPC) ->
 ( IPDU(PDUswitchOff) . sICR_PC(powerOff) .
                  PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
   IPDU(powerOn) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
   IPDU(powerOff) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
   IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                    PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
   IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
              PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
   IPDUTimer(pollPC) . sICR_PC(queryCRPCStatus) .
               PDU_State_Machine(WaitingCRPCReply,cRPCstarted,geoPCstarted,geoPressed,state)
 ) +
  ...
 (s==WaitingCRPCReply) ->
 ( rICR_PCrVal(IsOperational) . IndicationCB(systemStandby) .
             PDU_State_Machine(System_Standby,true,geoPCstarted,geoPressed,state) +
   rICR_PCrVal(IsNotOperational) . PDU_State_Machine(StartingUpCrPC,false,geoPCstarted,geoPressed,state)
 ) +
  ...
);
```

The fragment describes that when the system is switched on in the *PDU_Off* state, the ControlPC is powered on, the user gets an indication that the system is starting up, and the PDU transits to the *StartingUpCrPC* state. As can be inferred from the specification, the *StartingUpCrPC* state is used to not only monitor the progress of starting up the ControlPC, but also to react upon the external requests from users.

Then, when the PDU is stimulated by the timer via the *pollPC* signal, the PDU requests the state of the ControlPC by sending the *queryCRPCStatus* signal and transits to the *WaitingCRPCReply* state, waiting a response from the ControlPC. As specified in the external behavior of the ControlPC, either *IsOperational* or *IsNotOperational* signals are returned to the PDU. Depending on the return value, the PDU transits back to *StartingUpCrPC* (and hence can query the status

of the ControlPC again), or gives an indication that the system is in standby and transits to the *System_Standby* state.

Similarly, when the system is in the *System_Standby* stable state and the *PowerOn* button is pressed, the PDU transits to the *StartingUpAllPCs* state where all other PCs are checked, in the same manner of checking the status of the ControlPC described above.

```
...
(s==WaitingPC1statusReply) ->
( rIPCrVal(1,IsOperational) . sIPC(2,queryPCstatus) .
        PDU_State_Machine(WaitingPC2statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
  rIPCrVal(1,IsNotOperational) .
         PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
) +
(s==WaitingPC5statusReply) ->
( (!geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(systemOn) .
               PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,state) +
  (geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(geoStop) .
               PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,geoPressed,state) +
   rIPCrVal(5,IsNotOperational) .
            PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
...
```

That is, the first PC is checked if it is operational or not. If the first PC is not operational, the system can transit back to the *StartingUpAllPCs* state; see for example the specification of the *WaitingPC1statusReply* state above. If the first PC is operational, then the second PC is checked, and so on until all PCs are operational. When the last PC is operational, an indication is sent to the user, and then the PDU moves to the *System_On* state, see the *WaitingPC5statusReply* state.

During starting up of all PCs, the PDU queries the GeoPC and the ControlPC to check the status of whether any of the *Stop* buttons has been pressed or if the user needs to power off the entire system. If these flags are on, on the respective PCs, the PDU immediately switches off the taps supply the movable part or starts to power off the entire system. The PDU remembers the status of the *Stop* button, and therefore, when the last PC is operational, the PDU transits to *System_On* or *Geo_Stop* stable states.

The poll controller with global synchronous communication In combination with the first guideline, we use guideline 2 to model the instantaneous powering on or off the PCs. The same global synchronous communication concept used for the *Handler* process of the push model is also used here. We refer to G for the entire specification of this model.

8 Results of the experiments

After the specification of all models were created using the mCRL2 description language, we started the verification tasks. We used the mCRL2 tool set (July 2011 release) for performing verification and state space generation on a Unix-based server machine (4×2.5 Ghz processor and 46 GB RAM). The generated state spaces of all models were further analyzed using CADP (June 2011 beta release) for checking deadlocks, livelocks, illegals and proving refinements of designs against the external specification.

Table 4 depicts the activities performed throughout this work together with the tools used to accomplish each of them. The '✓' mark indicates a feature supported by the tool and being used in this work, '−' denotes that the feature is supported by the tool but is not being used in this work, and '×' indicates that the tool does not support the feature. As can be seen from the table, the formal specification using CADP is skipped since we used the mCRL2 for state space generation. The state space was analyzed later using both mCRL2 and CADP. We also translated the mCRL2 models to CSP and used FDR for state space generation. FDR was used to verify refinements under traces, failures and failures-divergence models, of which the last two are not supported by both mCRL2 and CADP. When the state space of each model has been generated, branching bisimulation reduction was applied after all internal events not visible on the external specification are hidden, to facilitate the verification and refinement tasks.

Activity	mCRL2	CADP	CSP/FDR
Formal specification	✓	–	✓
State space generation	✓	–	✓
Branching Bisimulation Reduction	✓	–	×
Checking deadlocks and illegals	✓	✓	✓
Checking livelocks	–	–	✓
Checking Weak-traces	✓	✓	✓
Checking Failures (-Divergence)	×	×	✓
Checking Observational	×	✓	×
Checking Safety	×	✓	×
Checking Tau*	×	✓	×
Checking Branching	×	✓	×

Table 4: List of performed tasks plus the tools used to realize them.

The three tools were used for searching for occurrences of deadlocks and illegals. All tools provided the same result, namely all models are deadlock and illegal free. After we hid all events except those exposed in the external specification, we checked for the occurrences of livelocks. Checking livelocks merely was accomplished using FDR (FDR2 2.91 academic use release). The reason of choosing FDR over other tools is that FDR provides readable, easy to analyze, counterexamples in case livelocks exist. The mCRL2 for example can report a sequence that leads to a cycle of *tau* events, but one can hardly deduce the corresponding original actions that form the cycle. The same applies for CADP.

We encountered a similar issue when trying to prove refinement of designs against the external behavior using both mCRL2 and CADP. The tools can easily find counterexamples when a refinement check is violated, under the refinement models they support. But, the generated counterexamples were hard to read since all original internal actions were permanently replaced by the hidden action *tau*. By using mCRL2 and CADP, we spent extra time analyzing the counterexamples and to 'guess' the correct original events correspond to the hidden events by matching the sequence of *tau's* on the original system. This indeed caused more efforts and time to be spent for modeling and verification since we did not efficiently know whether the design or the external specification was incorrect. Notable is that knowing the original actions correspond to the hidden action *tau* when checking refinements was straightforward in CSP/FDR.

However, when we attempted to verify an initial model of the push design using FDR, the tool quickly crashed during the compilation phase. The reason is that the model initially implements a *list* to store the started PCs, see the *startedPc* data parameter in the push model introduced earlier. The controller needs this list during the start-up of the system in order to know that all PCs are fully operational before moving to the *System_On* state. It seems that having such a list in our model caused FDR to crash, and thus when replacing the list by a counter, the issue was solved indeed. Notable is that mCRL2 dealt with both types of push models that include either a list or a counter of started PCs effectively. The last four refinement checks were performed using CADP, which was the only tool supporting them.

In table 5 we summarize the end result of checking refinements of designs, under a number of refinement models. The table is self-explainable. All designs refine the external specification under all refinement models, which means that all designs provide the expected behavior to external users of the system according to the predefined external specification. The only exception is the refinement of the poll models under the failures-divergences model, which fails due to the presence of a livelock.

The livelock exists in the poll models since internal *tau* loops can easily be formed, see Figure 7 for a livelock scenario. For example, in case the ControlPC is not operational, the PDU controller will query it again. This can continue forever, unless the ControlPC becomes operational. But, since the external users can still issue external commands even if the ControlPC is not operational, we consider this livelock to be rather benign, and indeed the livelock represents a desired and

Model	Weak-traces	Failures	Fail. Diverg.	observational	safety	Tau*	branching
Async. Push	✓	✓	✓	✓	✓	✓	✓
Async. Push Global Sync.	✓	✓	✓	✓	✓	✓	✓
Sync Push	✓	✓	✓	✓	✓	✓	✓
Sync Push Global sync.	✓	✓	✓	✓	✓	✓	✓
Poll	✓	✓	×	✓	✓	✓	✓
Poll Global sync.	✓	✓	×	✓	✓	✓	✓

Table 5: Results of checking refinements of designs against the external behavior

expected behavior.[1]

Note that all designs are deadlock, livelock and illegal free except the poll designs, which are not livelock free due to the above mentioned reason.

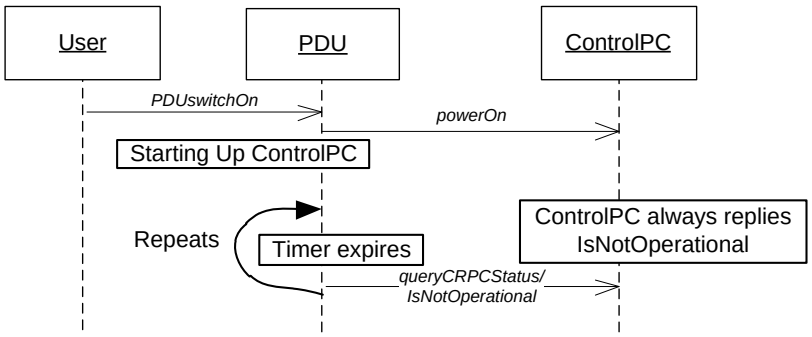

Figure 7: A divergence example

Model	States	Transitions	BB		BBDP	
External specification	15	53	13	46	13	47
Async. Push	78,088,550	122,354,296	47	173	47	173
Async. Push Global Sync.	44,866,381	75,945,810	47	173	47	173
Push sync	6,318	8,486	23	111	23	111
Push sync global sync	3,832	6,000	23	111	23	111
Poll	953	1,367	14	54	14	60
Poll global sync	608	1,022	14	54	14	60

Table 6: State spaces of all models

The last table sums up the statistical data related to the size of generated state spaces. The second and third columns shows the number of generated states and transitions for the entire state spaces. The branching-bisimulation (BB) columns depict the number of resulting states and transitions after the branching-bisimulation reduction was applied on the original state space, while those resulting from branching-bisimulation compression with divergence preserving (BBDP) are depicted in the last columns.

[1]In fact there are additional services deployed on a number of PCs for monitoring the status of PCs. If they detect that there is some PC has failed to start, they try to start it again using its baseboard management control (BMC) via its intelligent platform management interface (IPMI), through the Ethernet network. The PDU team is not responsible of implementing these services.

The difference between the number of transitions of the poll models after compression using BB and BBDP indicates that the poll model includes divergences. A divergence scenario of the poll model was discussed earlier.

As can be seen from the table, the poll variants appear to be better than others, with only 953 and 608 states. They show also fewer states after compression. This favorably compares to the asynchronous push model which includes 78,088,550 states. Therefore, it seems that extending the asynchronous push model further with extra details may limit the verification process, unless the design of the PDU controller is decomposed into a number of smaller components verified in isolation, or on-the-fly reduction techniques are used for circumventing a foreseen state space explosion.

Finally, the above results indicate that different design styles can substantially influence the number of states of the modeled systems. This confirms that these design styles are effective in practice. Although more experiments need to be done with these different design styles, we are strengthened in our believe that these styles are very important and designers should be actively aware of such strategies if they want to design verifiable systems.

References

[1] A. Bouajjani, J.C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for branching time semantics. In *Proceedings of the 18th international colloquium on Automata, languages and programming*, pages 76–92, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[2] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31:560–599, June 1984.

[3] CADP homepage. *http://www.inrialpes.fr/vasy/cadp/, 2011.*

[4] FDR homepage. *http://www.fsel.com, 2011.*

[5] J.C. Fernandez and L. Mounier. On the fly verification of behavioural equivalences and preorders. In *Proceedings of the 3rd International Workshop on Computer Aided Verification*, CAV '91, pages 181–191, London, UK, 1992. Springer-Verlag.

[6] J.F. Groote, J. Keiren, A. Mathijssen, B. Ploeger, F. Stappers, C. Tankink, Y. Usenko, M. v. Weerdenburg, W. Wesselink, T. Willemse, and J. v. d. Wulp. The mcrl2 toolset. In *Proceedings of the International Workshop on Advanced Software Development Tools and Techniques (WASDeTT 2008)*, 2008.

[7] J.F. Groote, T.W.D.M. Kouters, and A.A.H. Osaiweran. Specification guidelines to avoid the state space explosion problem. CS-Report 10-14, Eindhoven University of Technology, 2010.

[8] J.F. Groote, T.W.D.M. Kouters, and A.A.H. Osaiweran. Specifcation guidelines to avoid the state space explosion problem. In *Proceedings of the 4th IPM international Conference, FSEN 2011*, page (IN PRESS), Tehran, Iran, 2011. Springer-Verlag, Berlin, Germany.

[9] R. Kleihorst. Feature design - foundation power distribution subsystem, internal Philips document. 2010.

[10] R. Kleihorst. Power distribution unit - concept specification, internal Philips document, xdy036-080190. 2010.

[11] M. Loos. Feature design - startup/shutdown, internal Philips document, v0.6. 2010.

[12] A. Mathijssen and A.J. Pretorius. Verified design of an automated parking garage. In *Proceedings of the 11th international workshop, FMICS 2006 and 5th international workshop, PDMC conference on Formal methods: Applications and technology*, FMICS'06/PDMC'06, pages 165–180, Springer–Verlag, Berlin, Heidelberg, 2007.

[13] mCRL2 toolset homepage. *http://www.mcrl2.org/, 2011.*

[14] R. Milner. *Communication and concurrency.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[15] A.W. Roscoe. *The theory and practice of concurrency.* Prentice Hall, 1998.

[16] M. Van Eekelen, S. Ten Hoedt, R. Schreurs, and Y. S. Usenko. Analysis of a session-layer protocol in mcrl2: verification of a real-life industrial implementation. In *Proceedings of the 12th international conference on Formal methods for industrial critical systems*, FMICS'07, pages 182–199, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] R.J. van Glabbeek and P.W. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43:555–600, May 1996.

A External behaviour

```
sort State = struct PDU_Off | System_StandBy | System_On | Emergency_Off | System_Off | Geo_Stop |
                                      Off | StartingUp | StartingUpCrPC | StartingUpAllPCs;
Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff | emergencyOff;
IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;
act IPDU:Command;
act  IndicationCB: IndicationMsg;

proc ExtSpec(s:State)=
        (s==PDU_Off) ->
                ( IPDU(PDUswitchOn) .
                  IndicationCB(startingUp) . ExtSpec(StartingUpCrPC) ) +
        (s==System_StandBy) ->
                ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
                  IPDU(powerOn).
                  IndicationCB(startingUp)  . ExtSpec(StartingUpAllPCs)+
                  IPDU(powerOff) . ExtSpec(System_StandBy) +
                  IPDU(forcedPowerOff)  .
                  IndicationCB(off)  . ExtSpec(System_Off) +
                  IPDU(emergencyOff)  .
                  IndicationCB(off)  . ExtSpec(Emergency_Off) +
                  tau . IndicationCB(off)  . ExtSpec(System_Off) ) +
        (s==System_On) ->
                ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
                  IPDU(powerOn)  .
                  IndicationCB(startingUp)  . ExtSpec(StartingUpAllPCs)+
                  IPDU(powerOff)  .
                  IndicationCB(systemStandby)  . ExtSpec(System_StandBy) +
                  IPDU(forcedPowerOff)  .
                  IndicationCB(off)  . ExtSpec(System_Off) +
                  IPDU(emergencyOff)  .
                  IndicationCB(off) . ExtSpec(Emergency_Off) +
                  tau . IndicationCB(off)  . ExtSpec(System_Off) +
                  tau . IndicationCB(geoStop)  . ExtSpec(Geo_Stop)  ) +
        (s==StartingUpAllPCs) ->
                ( IPDU(PDUswitchOff)  . ExtSpec(PDU_Off) +
                  IPDU(powerOn) . ExtSpec(StartingUpAllPCs) +
          IPDU(powerOff)  . IndicationCB(systemStandby) . ExtSpec(System_StandBy)+
                  IPDU(powerOff)  . ExtSpec(StartingUpCrPC)+
                     IPDU(forcedPowerOff) .  IndicationCB(off)  . ExtSpec(System_Off) +
                  IPDU(emergencyOff) .  IndicationCB(off) . ExtSpec(Emergency_Off) +
                  tau . IndicationCB(off)  . ExtSpec(System_Off) +
                  tau . IndicationCB(systemOn) . ExtSpec(System_On) +
                  tau . ExtSpec(StartingUpAllPCs) +
                  tau . IndicationCB(geoStop) . ExtSpec(Geo_Stop) ) +
        (s==Geo_Stop) ->
                ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
                  IPDU(powerOn) . IndicationCB(systemOn)  . ExtSpec(System_On) +
                  IPDU(powerOff).IndicationCB(systemStandby).ExtSpec(System_StandBy) +
                  IPDU(forcedPowerOff)  .  IndicationCB(off)  . ExtSpec(System_Off) +
                  IPDU(emergencyOff)  .  IndicationCB(off)  . ExtSpec(Emergency_Off) +
                  tau . IndicationCB(off)  . ExtSpec(System_Off) ) +
        (s==System_Off) ->
                ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
                  IPDU(powerOn) . IndicationCB(startingUp) . ExtSpec(StartingUpAllPCs)+
                  IPDU(powerOff) . ExtSpec(System_Off) +
                  IPDU(forcedPowerOff) . ExtSpec(System_Off) +
                  IPDU(emergencyOff)  . ExtSpec(Emergency_Off)  ) +
        (s==Emergency_Off) ->
                ( IPDU(PDUswitchOff) . ExtSpec(PDU_Off) +
                  IPDU(powerOn) . IndicationCB(startingUp)  . ExtSpec(StartingUpAllPCs) +
                  IPDU(powerOff)    . ExtSpec(Emergency_Off) +
                  IPDU(forcedPowerOff) . ExtSpec(Emergency_Off) +
                  IPDU(emergencyOff)  . ExtSpec(Emergency_Off)  ) +
        (s==StartingUpCrPC) ->
                ( IPDU(PDUswitchOff)  . ExtSpec(PDU_Off) +
```

```
                  IPDU(powerOn)  . ExtSpec(StartingUpCrPC)+
                  IPDU(powerOff) . ExtSpec(StartingUpCrPC) +
                  IPDU(forcedPowerOff)  .  IndicationCB(off)  . ExtSpec(System_Off)+
                  IPDU(emergencyOff)  .  IndicationCB(off) . ExtSpec(Emergency_Off)+
                  tau . IndicationCB(systemStandby) . ExtSpec(System_StandBy) ) ;

init hide({},ExtSpec(PDU_Off));
```

B Asynchronous push model

```

sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff
                                                 | emergencyOff | onPressed;
Bmsg = struct restart | shutdown ;
PCsCallbacks = struct controlPowerOff | stop | started ;
PCState = struct PC_Off | Operational | WaitingShutdown | StartingUp | OS_Shutdown | StopPressed;
PDUState = struct PDU_Off | StartingUp_CR_PC | StartingUpAllPcs | SystemStandby | System_On
                                                 | Emergency_Off | System_Off | Geo_Stop ;
IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;
PCs = struct CRPC | GeoPC | NormalPC ;
sort  CBMsg = struct msg(pc:PCs,id:Pos,cb:PCsCallbacks);

act Illegal,
srelease,rrelease,release;
sIPDU, rIPDU, IPDU,
sICR_PC,rICR_PC,ICR_PC,
sIGeoPC,rIGeoPC,IGeoPC,
 scommandhandler, rcommandhandler, commandhandler : Command;
sIPC,rIPC,IPC : Pos # Command;
sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
sMsghandler, rMsghandler, Msghandler: Bmsg;
sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;
rICR_PC_CB, sICR_PC_CBout, ICR_PC_CBout, sICR_PC_CB,   rICR_PC_CBin, ICR_PC_CBin,
rIGeoPC_CB, sIGeoPC_CBout,  IGeoPC_CBout , sIGeoPC_CB, rIGeoPC_CBin , IGeoPC_CBin: PCsCallbacks;

rIPC_CB,   sIPC_CBout,    IPC_CBout,    sIPC_CB,      rIPC_CBin,    IPC_CBin : Pos # PCsCallbacks;

IndicationCB : IndicationMsg;

proc ControlPC(s:PCState) = (
(s==PC_Off) ->
( rICR_PC(powerOn) . ControlPC(StartingUp) +
 rICR_PC(powerOff) . Illegal . delta +
 rICR_PC_Broadcast(restart) . Illegal . delta +
 rICR_PC_Broadcast(shutdown) . Illegal . delta
) +
(s==Operational) ->
( rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 sICR_PC_CB(controlPowerOff) . ControlPC(WaitingShutdown) +
 rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
)+
(s==WaitingShutdown) ->
( rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
)+
(s==OS_Shutdown) -> (
 rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . Illegal . delta +
 rICR_PC_Broadcast(shutdown) . Illegal . delta
)+
```

```
(s==StartingUp) ->
( rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . Illegal . delta +
 rICR_PC_Broadcast(shutdown) . Illegal . delta +
 sICR_PC_CB(started) . ControlPC(Operational)
)
);


proc GeoPC(s:PCState) = (
(s==PC_Off) ->
( rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC(powerOn) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC(powerOff) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==Operational) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . GeoPC(Operational) + % it can happen if the restart is issued before
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 sIGeoPC_CB(stop) . GeoPC(StopPressed)
)+
(s==StopPressed) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . GeoPC(Operational)
)+
(s==OS_Shutdown) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(StartingUp) +
 rIGeoPC_Broadcast(restart) . Illegal . delta +
 sIGeoPC_CB(started) . GeoPC(Operational)
)
);

proc NormalPC(id:Pos,s:PCState) = (
(s==PC_Off) ->
( rIPC(id,powerOn) . NormalPC(id,StartingUp) +
 rIPC(id,powerOff) . Illegal . delta +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==Operational) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
 rIPC_Broadcast(id,restart) . NormalPC(id,StartingUp)
)+
(s==OS_Shutdown) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
```

```
124 )+
125 (s==StartingUp) ->
126 ( rIPC(id,powerOn) . Illegal . delta +
127  rIPC(id,powerOff) . NormalPC(id,PC_Off) +
128  rIPC_Broadcast(id,shutdown) . NormalPC(id,StartingUp) +
129  rIPC_Broadcast(id,restart) . Illegal . delta +
130  sIPC_CB(id,started) . NormalPC(id,Operational)
131 ) );
132
133 % the PDU design
134
135 proc PDU_State_Machine(s:PDUState,geopcOn,crpcOn,geoPressed:Bool,startedPc:Nat) = (
136 (s==PDU_Off) ->
137 ( rIPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
138                       PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
139  rICR_PC_CB(controlPowerOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
140  rIGeoPC_CB(stop) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
141  rICR_PC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
142  rIGeoPC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
143  sum id:Pos . rIPC_CB(id,started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc)
144 )+
145 (s==StartingUp_CR_PC) ->
146 (
147  rIPDU(PDUswitchOff) . sICR_PC(powerOff) .
148                        PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
149  rIPDU(powerOn) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
150  rIPDU(powerOff) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
151  rIPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
152                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
153  rIPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
154                             PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
155  rICR_PC_CB(controlPowerOff) . Illegal . delta +
156  rICR_PC_CB(started) . IndicationCB(systemStandby) .
157                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
158  rIGeoPC_CB(stop) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
159  rIGeoPC_CB(started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
160  sum id:Pos . rIPC_CB(id,started) .
161                        PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc)
162 )+
163 (s==SystemStandby) ->
164 (
165  rIPDU(PDUswitchOff) . sICR_PC(powerOff) .
166                           PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
167  rIPDU(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
168              IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,true,false,0) +
169  rIPDU(powerOff) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
170  rIPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
171                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
172  rIPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
173                           PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
174  rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sICR_PC(powerOff) .
175            IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
176  rIGeoPC_CB(stop) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
177  rICR_PC_CB(started) . Illegal . delta +
178  rIGeoPC_CB(started) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
179  sum id:Pos . rIPC_CB(id,started) .
180                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc)
181 )+
182 (s==System_On) ->
183 (
184  rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
185                        rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
186  rIPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) . sMsghandler(restart) .
187      rrelease . IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
188  rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease
189              sIGeoPC(powerOff) . scommandhandler(powerOff) .  rrelease . IndicationCB(systemStandby).
190                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
```

```
191   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
192     rrelease . IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
193   rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .  rrelease .
194             IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
195   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
196       sMsghandler(shutdown) .  rrelease . sICR_PC(powerOff) . sIGeoPC(powerOff) .
197       scommandhandler(powerOff) . rrelease . IndicationCB(off) .
198       PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
199   rIGeoPC_CB(stop) . IndicationCB(geoStop) .
200                      PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
201   rICR_PC_CB(started) . Illegal . delta +
202   rIGeoPC_CB(started) . Illegal . delta +
203   sum id:Pos . rIPC_CB(id,started) . Illegal . delta
204  )+
205  (s==Emergency_Off) ->
206  (
207   rIPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
208   rIPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
209                 IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
210   rIPDU(powerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
211   rIPDU(forcedPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
212   rIPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
213   rICR_PC_CB(controlPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
214   rIGeoPC_CB(stop) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
215   rICR_PC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
216   rIGeoPC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
217   sum id:Pos . rIPC_CB(id,started) .
218                        PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc)
219  )+
220  (s==System_Off) ->
221  (
222   rIPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
223   rIPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
224              IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
225   rIPDU(powerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
226   rIPDU(forcedPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
227   rIPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
228   rICR_PC_CB(controlPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
229   rIGeoPC_CB(stop) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
230   rICR_PC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
231   rIGeoPC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
232   sum id:Pos . rIPC_CB(id,started) .
233                               PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc)
234  )+
235  (s==Geo_Stop) ->
236  (
237   rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
238                       rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
239   rIPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
240                                 PDU_State_Machine(System_On,geopcOn,crpcOn,false,startedPc) +
241   rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
242        sIGeoPC(powerOff) .scommandhandler(powerOff) . rrelease . IndicationCB(systemStandby) .
243                  PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
244   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
245     rrelease . IndicationCB(off) .
246             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
247   rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
248          rrelease . IndicationCB(off) .
249                        PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
250   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
251      sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) . sIGeoPC(powerOff) .
252               scommandhandler(powerOff) . rrelease . IndicationCB(off) .
253                          PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
254   rIGeoPC_CB(stop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
255   rICR_PC_CB(started) . Illegal . delta +
256   rIGeoPC_CB(started) . Illegal . delta +
257   sum id:Pos . rIPC_CB(id,started) . Illegal . delta
```

```
)+
(s==StartingUpAllPcs) ->
(
 rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
                     rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIPDU(powerOn) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
(crpcOn ) -> rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
       rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
       IndicationCB(systemStandby) . PDU_State_Machine(SystemStandby,false,crpcOn,false,0) +
(! crpcOn ) -> rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
            rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
                                  PDU_State_Machine(StartingUp_CR_PC,false,crpcOn,false,0) +
 rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
     rrelease . IndicationCB(off) .
                         PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
      rrelease . IndicationCB(off) .
          PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 (crpcOn ) -> rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) .
          sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
          sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
          IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 (! crpcOn ) -> rICR_PC_CB(controlPowerOff) .
        PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
 ((geopcOn && startedPc==5 && geoPressed)) -> rICR_PC_CB(started) . IndicationCB(geoStop) .
                          PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
 ((geopcOn && startedPc==5 && ! geoPressed)) -> rICR_PC_CB(started) . IndicationCB(systemOn) .
                               PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
 ((! geopcOn || startedPc<5)) -> rICR_PC_CB(started) .
                          PDU_State_Machine(StartingUpAllPcs,geopcOn,true,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,true,startedPc)+
 (! geopcOn && (startedPc<5 || ! crpcOn)) -> rIGeoPC_CB(started) .
                           PDU_State_Machine(StartingUpAllPcs,true,crpcOn,geoPressed,startedPc) +
 (crpcOn && startedPc==5 && ! geoPressed) -> rIGeoPC_CB(started) . IndicationCB(systemOn) .
                               PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
 (crpcOn && startedPc==5 && geoPressed) -> rIGeoPC_CB(started) . IndicationCB(geoStop) .
                          PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
 (startedPc<4) -> sum id:Pos . rIPC_CB(id,started) .
                     PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1) +
 (startedPc==4 && geopcOn && crpcOn && ! geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
       IndicationCB(systemOn) . PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
 (startedPc==4 && geopcOn && crpcOn && geoPressed) -> sum id:Pos .
       rIPC_CB(id,started) . IndicationCB(geoStop) .
                                  PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
 ((startedPc==4) && (geopcOn==false || ! crpcOn)) -> sum id:Pos . rIPC_CB(id,started) .
                        PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1)
)
);

 proc CallBackQueue(l:List(CBMsg))=
   sum id:Pos,cbm:PCsCallbacks . rIPC_CBin(id,cbm) . CallBackQueue(msg(NormalPC,id,cbm) |> l) +
   sum cbm:PCsCallbacks . rICR_PC_CBin(cbm) . CallBackQueue(msg(CRPC,1,cbm) |> l) +
   sum cbm:PCsCallbacks . rIGeoPC_CBin(cbm) . CallBackQueue(msg(GeoPC,1,cbm) |> l) +
   (l!=[] && pc(rhead(l))==NormalPC) ->
                     sIPC_CBout(id(rhead(l)),cb(rhead(l))). CallBackQueue(rtail(l)) +
   (l!=[] && pc(rhead(l))==CRPC) ->  sICR_PC_CBout(cb(rhead(l))).
                                                  CallBackQueue(rtail(l)) +
   (l!=[] && pc(rhead(l))==GeoPC) ->  sIGeoPC_CBout(cb(rhead(l))).
                                                   CallBackQueue(rtail(l))+
   (l==[]) -> sum c:Command. sIPDU(c) . CallBackQueue (l);


proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) . sIPC(2,c) . sIPC(3,c) .
                                                 sIPC(4,c) . srelease|sIPC(5,c) .  Handler +
               sum m:Bmsg . rMsghandler(m) | sIPC_Broadcast(1,m) . sIPC_Broadcast(2,m) .
           sIPC_Broadcast(3,m) .sIPC_Broadcast(4,m) . srelease|sIPC_Broadcast(5,m) . Handler;

```

```
325
326 proc System=
327       hide({           ICR_PC,ICR_PC_Broadcast,ICR_PC_CBout,ICR_PC_CBin,
328                        IGeoPC,IGeoPC_Broadcast,IGeoPC_CBout,IGeoPC_CBin,
329                        IPC,IPC_Broadcast,IPC_CBout,IPC_CBin,
330                release,
331           Msghandler,commandhandler
332              },
333           allow({IPDU,
334                  ICR_PC,ICR_PC_Broadcast,ICR_PC_CBin, ICR_PC_CBout,
335                  IGeoPC,IGeoPC_Broadcast,IGeoPC_CBin,IGeoPC_CBout,
336                  IPC   ,IPC_Broadcast   ,IPC_CBin , IPC_CBout,
337                  IndicationCB,
338                  commandhandler|IPC,Msghandler|IPC_Broadcast, release|IPC,release|IPC_Broadcast,
339                  release,
340                  Illegal
341                },
342            comm({sIPDU|rIPDU-> IPDU,
343                   sICR_PC|rICR_PC->ICR_PC,
344                  sICR_PC_Broadcast|rICR_PC_Broadcast->ICR_PC_Broadcast,
345                  sIGeoPC|rIGeoPC->IGeoPC,
346                  sIGeoPC_Broadcast|rIGeoPC_Broadcast->IGeoPC_Broadcast,
347                  sICR_PC_CBout|rICR_PC_CB->ICR_PC_CBout,
348                  sICR_PC_CB|rICR_PC_CBin->ICR_PC_CBin,
349                  sIGeoPC_CBout|rIGeoPC_CB->IGeoPC_CBout,
350                  sIGeoPC_CB|rIGeoPC_CBin->IGeoPC_CBin,
351                  sIPC_CBout  | rIPC_CB -> IPC_CBout,
352                      sIPC_CB | rIPC_CBin -> IPC_CBin,
353                  srelease | rrelease ->release,
354                      rIPC|sIPC->IPC,
355                  rIPC_Broadcast|sIPC_Broadcast->IPC_Broadcast,
356
357                  sMsghandler | rMsghandler -> Msghandler,
358                  scommandhandler | rcommandhandler -> commandhandler
359
360               },
361                          CallBackQueue([])  ||
362                           ControlPC(PC_Off)  ||
363                            GeoPC(PC_Off)      ||
364                            NormalPC(1,PC_Off) ||
365                            NormalPC(2,PC_Off) ||
366                            NormalPC(3,PC_Off) ||
367                            NormalPC(4,PC_Off) ||
368                            NormalPC(5,PC_Off) ||
369                           Handler                   ||
370                           PDU_State_Machine(PDU_Off,false,false,false,0))));
371
372 init System;
373
```

C Asynchronous push model with global synchronous communication

```
1
2  sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff |
3                                       emergencyOff | onPressed;
4  Bmsg = struct restart | shutdown ;
5  PCsCallbacks = struct controlPowerOff | stop | started ;
6  PCState = struct PC_Off | Operational | WaitingShutdown | StartingUp | OS_Shutdown |
7                                                                    StopPressed;
8  PDUState = struct PDU_Off | StartingUp_CR_PC | StartingUpAllPcs | SystemStandby |
9                                  System_On | Emergency_Off | System_Off | Geo_Stop ;
10 IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;
11 PCs = struct CRPC | GeoPC | NormalPC ;
12 sort  CBMsg = struct msg(pc:PCs,id:Pos,cb:PCsCallbacks);
```

```
13
14  act Illegal,
15  srelease,rrelease,release;
16  sIPDU, rIPDU, IPDU,
17  sICR_PC,rICR_PC,ICR_PC,
18  sIGeoPC,rIGeoPC,IGeoPC,
19   scommandhandler, rcommandhandler, commandhandler : Command;
20  sIPC,rIPC,IPC : Pos # Command;
21  sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
22  sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
23  sMsghandler, rMsghandler, Msghandler: Bmsg;
24  sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;
25  rICR_PC_CB, sICR_PC_CBout, ICR_PC_CBout, sICR_PC_CB,   rICR_PC_CBin, ICR_PC_CBin,
26  rIGeoPC_CB, sIGeoPC_CBout,  IGeoPC_CBout , sIGeoPC_CB, rIGeoPC_CBin ,
27                                                          IGeoPC_CBin: PCsCallbacks;
28
29  rIPC_CB,   sIPC_CBout,    IPC_CBout,   sIPC_CB,  rIPC_CBin,
30                                                    IPC_CBin : Pos # PCsCallbacks;
31
32  IndicationCB : IndicationMsg;
33
34  proc ControlPC(s:PCState) = (
35  (s==PC_Off) ->
36  ( rICR_PC(powerOn) . ControlPC(StartingUp) +
37   rICR_PC(powerOff) . Illegal . delta +
38   rICR_PC_Broadcast(restart) . Illegal . delta +
39   rICR_PC_Broadcast(shutdown) . Illegal . delta
40  ) +
41  (s==Operational) ->
42  ( rICR_PC(powerOn) . Illegal . delta +
43   rICR_PC(powerOff) . ControlPC(PC_Off) +
44   sICR_PC_CB(controlPowerOff) . ControlPC(WaitingShutdown) +
45   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
46   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
47  )+
48  (s==WaitingShutdown) ->
49  ( rICR_PC(powerOn) . Illegal . delta +
50   rICR_PC(powerOff) . ControlPC(PC_Off) +
51   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
52   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
53  )+
54  (s==OS_Shutdown) -> (
55   rICR_PC(powerOn) . Illegal . delta +
56   rICR_PC(powerOff) . ControlPC(PC_Off) +
57   rICR_PC_Broadcast(restart) . Illegal . delta +
58   rICR_PC_Broadcast(shutdown) . Illegal . delta
59  )+
60  (s==StartingUp) ->
61  ( rICR_PC(powerOn) . Illegal . delta +
62   rICR_PC(powerOff) . ControlPC(PC_Off) +
63   rICR_PC_Broadcast(restart) . Illegal . delta +
64   rICR_PC_Broadcast(shutdown) . Illegal . delta +
65   sICR_PC_CB(started) . ControlPC(Operational)
66  )
67  );
68
69
70  proc GeoPC(s:PCState) = (
71  (s==PC_Off) ->
72  ( rIGeoPC_Broadcast(shutdown) . Illegal . delta +
73   rIGeoPC(powerOn) . GeoPC(StartingUp) +
74   rIGeoPC(onPressed) . Illegal . delta +
75   rIGeoPC(powerOff) . Illegal . delta +
76   rIGeoPC_Broadcast(restart) . Illegal . delta
77  )+
78  (s==Operational) ->
79  ( rIGeoPC(powerOn) . Illegal . delta +
```

```
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . GeoPC(Operational) + % it can happen if the restart is issued before
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 sIGeoPC_CB(stop) . GeoPC(StopPressed)
)+
(s==StopPressed) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . GeoPC(Operational)
)+
(s==OS_Shutdown) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(StartingUp) +
 rIGeoPC_Broadcast(restart) . Illegal . delta +
 sIGeoPC_CB(started) . GeoPC(Operational)
)
);

proc NormalPC(id:Pos,s:PCState) = (
(s==PC_Off) ->
( rIPC(id,powerOn) . NormalPC(id,StartingUp) +
 rIPC(id,powerOff) . Illegal . delta +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==Operational) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
 rIPC_Broadcast(id,restart) . NormalPC(id,StartingUp)
)+
(s==OS_Shutdown) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,StartingUp) +
 rIPC_Broadcast(id,restart) . Illegal . delta +
 sIPC_CB(id,started) . NormalPC(id,Operational)
) );

% the PDU design

proc PDU_State_Machine(s:PDUState,geopcOn,crpcOn,geoPressed:Bool,startedPc:Nat) = (
(s==PDU_Off) ->
( rIPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
                     PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc)
```

```
147  )+
148  (s==StartingUp_CR_PC) ->
149  (
150   rIPDU(PDUswitchOff) . sICR_PC(powerOff) .
151                                PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
152   rIPDU(powerOn) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
153   rIPDU(powerOff) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
154   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
155                              PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
156   rIPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
157                              PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
158   rICR_PC_CB(controlPowerOff) . Illegal . delta +
159   rICR_PC_CB(started) . IndicationCB(systemStandby) .
160                            PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
161   rIGeoPC_CB(stop) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
162   rIGeoPC_CB(started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
163   sum id:Pos . rIPC_CB(id,started) .
164                           PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc)
165  )+
166  (s==SystemStandby) ->
167  (
168   rIPDU(PDUswitchOff) . sICR_PC(powerOff) .
169                                PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
170   rIPDU(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) . IndicationCB(startingUp) .
171                                       PDU_State_Machine(StartingUpAllPcs,false,true,false,0) +
172   rIPDU(powerOff) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
173   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
174                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
175   rIPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
176                            PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
177   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sICR_PC(powerOff) . IndicationCB(off) .
178                              PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
179   rIGeoPC_CB(stop) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
180   rICR_PC_CB(started) . Illegal . delta +
181   rIGeoPC_CB(started) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
182   sum id:Pos . rIPC_CB(id,started) .
183                            PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc)
184  )+
185  (s==System_On) ->
186  (
187   rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
188                              PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
189   rIPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) . sMsghandler(restart) .
190     rrelease . IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
191   rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
192       sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(systemStandby) .
193         PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
194   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
195            IndicationCB(off) .  PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
196   rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
197         IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
198   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
199                       sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
200                          sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(off) .
201                            PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
202   rIGeoPC_CB(stop) . IndicationCB(geoStop) .
203                              PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
204   rICR_PC_CB(started) . Illegal . delta +
205   rIGeoPC_CB(started) . Illegal . delta +
206   sum id:Pos . rIPC_CB(id,started) . Illegal . delta
207  )+
208  (s==Emergency_Off) ->
209  (
210   rIPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
211   rIPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) .
212        IndicationCB(startingUp) .PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
213   rIPDU(powerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
```

```
214   rIPDU(forcedPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
215   rIPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
216   rICR_PC_CB(controlPowerOff) .
217                         PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
218   rIGeoPC_CB(stop) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
219   rICR_PC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
220   rIGeoPC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
221   sum id:Pos . rIPC_CB(id,started) .
222                           PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc)
223  )+
224  (s==System_Off) ->
225  (
226   rIPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
227   rIPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) .
228        IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
229   rIPDU(powerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
230   rIPDU(forcedPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
231   rIPDU(emergencyOff) .
232                         PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
233   rICR_PC_CB(controlPowerOff) .
234                         PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
235   rIGeoPC_CB(stop) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
236   rICR_PC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
237   rIGeoPC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
238   sum id:Pos . rIPC_CB(id,started) .
239                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc)
240  )+
241  (s==Geo_Stop) ->
242  (
243   rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
244                    PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
245   rIPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
246                     PDU_State_Machine(System_On,geopcOn,crpcOn,false,startedPc) +
247   rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
248                  sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(systemStandby) .
249                  PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
250   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
251        IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
252   rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
253     IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
254   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
255           sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
256                sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(off) .
257                 PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
258   rIGeoPC_CB(stop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
259   rICR_PC_CB(started) . Illegal . delta +
260   rIGeoPC_CB(started) . Illegal . delta +
261   sum id:Pos . rIPC_CB(id,started) . Illegal . delta
262  )+
263  (s==StartingUpAllPcs) ->
264  (
265   rIPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
266                           PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
267   rIPDU(powerOn) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
268  (crpcOn ) -> rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
269          rrelease . sIGeoPC(powerOff) | scommandhandler(powerOff) .
270         IndicationCB(systemStandby) . PDU_State_Machine(SystemStandby,false,crpcOn,false,0) +
271  (! crpcOn ) -> rIPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
272                       rrelease . sIGeoPC(powerOff) | scommandhandler(powerOff) .
273                          PDU_State_Machine(StartingUp_CR_PC,false,crpcOn,false,0) +
274   rIPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
275                    IndicationCB(off) .
276                           PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
277   rIPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
278     IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
279   (crpcOn ) -> rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) .
280        sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
```

```
281                sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(off) .
282                PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
283   (! crpcOn ) -> rICR_PC_CB(controlPowerOff) .
284                   PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
285   ((geopcOn && startedPc==5 && geoPressed)) -> rICR_PC_CB(started) . IndicationCB(geoStop) .
286                           PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
287   ((geopcOn && startedPc==5 && ! geoPressed)) -> rICR_PC_CB(started) . IndicationCB(systemOn) .
288                          PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
289   ((! geopcOn || startedPc<5)) -> rICR_PC_CB(started) .
290                     PDU_State_Machine(StartingUpAllPcs,geopcOn,true,geoPressed,startedPc) +
291   rIGeoPC_CB(stop) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,true,startedPc)+
292   (! geopcOn && (startedPc<5 || ! crpcOn)) -> rIGeoPC_CB(started) .
293                     PDU_State_Machine(StartingUpAllPcs,true,crpcOn,geoPressed,startedPc) +
294   (crpcOn && startedPc==5 && ! geoPressed) -> rIGeoPC_CB(started) . IndicationCB(systemOn) .
295                          PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
296   (crpcOn && startedPc==5 && geoPressed) -> rIGeoPC_CB(started) . IndicationCB(geoStop) .
297                           PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
298   (startedPc<4) -> sum id:Pos . rIPC_CB(id,started) .
299                 PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1) +
300   (startedPc==4 && geopcOn && crpcOn && ! geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
301      IndicationCB(systemOn) . PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
302   (startedPc==4 && geopcOn && crpcOn && geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
303        IndicationCB(geoStop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
304   ((startedPc==4) && (geopcOn==false || ! crpcOn)) -> sum id:Pos . rIPC_CB(id,started) .
305                     PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1)
306   )
307   );
308
309    proc CallBackQueue(l:List(CBMsg))=
310      sum id:Pos,cbm:PCsCallbacks . rIPC_CBin(id,cbm) . CallBackQueue(msg(NormalPC,id,cbm) |> l) +
311      sum cbm:PCsCallbacks . rICR_PC_CBin(cbm) . CallBackQueue(msg(CRPC,1,cbm) |> l) +
312      sum cbm:PCsCallbacks . rIGeoPC_CBin(cbm) . CallBackQueue(msg(GeoPC,1,cbm) |> l) +
313      (l!=[] && pc(rhead(l))==NormalPC) ->  sIPC_CBout(id(rhead(l)),cb(rhead(l))). CallBackQueue(rtail(l)) +
314      (l!=[] && pc(rhead(l))==CRPC) ->  sICR_PC_CBout(cb(rhead(l))). CallBackQueue(rtail(l)) +
315      (l!=[] && pc(rhead(l))==GeoPC) ->  sIGeoPC_CBout(cb(rhead(l))). CallBackQueue(rtail(l))+
316      (l==[]) -> sum c:Command. sIPDU(c) . CallBackQueue (l);
317
318
319   proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) | sIPC(2,c) | sIPC(3,c) |
320                                                          sIPC(4,c) | sIPC(5,c) .   Handler +
321                  sum m:Bmsg . rMsghandler(m) | sIPC_Broadcast(1,m) . sIPC_Broadcast(2,m) .
322                          sIPC_Broadcast(3,m) .sIPC_Broadcast(4,m) . srelease|sIPC_Broadcast(5,m) . Handler;
323
324
325   proc System=
326         hide({ ICR_PC,ICR_PC_Broadcast,ICR_PC_CBout,ICR_PC_CBin,
327                IGeoPC,IGeoPC_Broadcast,IGeoPC_CBout,IGeoPC_CBin,
328                IPC,IPC_Broadcast,IPC_CBout,IPC_CBin,
329                release,
330                 Msghandler,commandhandler
331              },
332           allow({IPDU,
333                  ICR_PC,ICR_PC_Broadcast,ICR_PC_CBin, ICR_PC_CBout,
334                  IGeoPC,IGeoPC_Broadcast,IGeoPC_CBin,IGeoPC_CBout,
335                  IPC   ,IPC_Broadcast   ,IPC_CBin , IPC_CBout,
336                  IndicationCB,
337                  IGeoPC|commandhandler|IPC|IPC|IPC|IPC|IPC,
338                  Msghandler|IPC_Broadcast,
339                  release|IPC_Broadcast,
340                   release,
341                  Illegal
342                 },
343            comm({sIPDU|rIPDU-> IPDU,
344                  sICR_PC|rICR_PC->ICR_PC,
345                  sICR_PC_Broadcast|rICR_PC_Broadcast->ICR_PC_Broadcast,
346                  sIGeoPC|rIGeoPC->IGeoPC,
347                  sIGeoPC_Broadcast|rIGeoPC_Broadcast->IGeoPC_Broadcast,
```

```
348                 sICR_PC_CBout|rICR_PC_CB->ICR_PC_CBout,
349                 sICR_PC_CB|rICR_PC_CBin->ICR_PC_CBin,
350                 sIGeoPC_CBout|rIGeoPC_CB->IGeoPC_CBout,
351                 sIGeoPC_CB|rIGeoPC_CBin->IGeoPC_CBin,
352                 sIPC_CBout  | rIPC_CB -> IPC_CBout,
353                     sIPC_CB | rIPC_CBin -> IPC_CBin,
354                 srelease | rrelease ->release,
355                     rIPC|sIPC->IPC,
356                 rIPC_Broadcast|sIPC_Broadcast->IPC_Broadcast,
357
358                 sMsghandler | rMsghandler -> Msghandler,
359                 scommandhandler | rcommandhandler -> commandhandler
360
361               },
362                           CallBackQueue([])  ||
363                            ControlPC(PC_Off)  ||
364                             GeoPC(PC_Off)      ||
365                             NormalPC(1,PC_Off) ||
366                             NormalPC(2,PC_Off) ||
367                             NormalPC(3,PC_Off) ||
368                             NormalPC(4,PC_Off) ||
369                             NormalPC(5,PC_Off) ||
370                            Handler                   ||
371                            PDU_State_Machine(PDU_Off,false,false,false,0))));
372
373   init System;
374
375
```

D Synchronous push model

```
1
2   sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff |
3                                                                  emergencyOff | onPressed;
4   Bmsg = struct restart | shutdown ;
5   PCsCallbacks = struct controlPowerOff | stop | started ;
6   PCState = struct PC_Off | Operational | WaitingShutdown | StartingUp | OS_Shutdown | StopPressed;
7   PDUState = struct PDU_Off | StartingUp_CR_PC | StartingUpAllPcs | SystemStandby | System_On |
8                                                        Emergency_Off | System_Off | Geo_Stop ;
9   IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;
10
11  act Illegal,
12  srelease,rrelease,release;
13  IPDU,
14  sICR_PC,rICR_PC,ICR_PC,
15  sIGeoPC,rIGeoPC,IGeoPC,
16   scommandhandler, rcommandhandler, commandhandler : Command;
17  sIPC,rIPC,IPC : Pos # Command;
18  sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
19  sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
20  sMsghandler, rMsghandler, Msghandler: Bmsg;
21  sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;
22  sICR_PC_CB,rICR_PC_CB,ICR_PC_CB,
23  sIGeoPC_CB,rIGeoPC_CB,IGeoPC_CB: PCsCallbacks;
24  sIPC_CB,rIPC_CB,IPC_CB : Pos # PCsCallbacks;
25  IndicationCB : IndicationMsg;
26
27  proc ControlPC(s:PCState) = (
28  (s==PC_Off) ->
29  ( rICR_PC(powerOn) . ControlPC(StartingUp) +
30   rICR_PC(powerOff) . Illegal . delta +
31   rICR_PC_Broadcast(restart) . Illegal . delta +
32   rICR_PC_Broadcast(shutdown) . Illegal . delta
33  ) +
34  (s==Operational) ->
35  ( rICR_PC(powerOn) . Illegal . delta +
```

```
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 sICR_PC_CB(controlPowerOff) . ControlPC(WaitingShutdown) +
 rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
)+
(s==WaitingShutdown) ->
( rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
)+
(s==OS_Shutdown) -> (
 rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . Illegal . delta +
 rICR_PC_Broadcast(shutdown) . Illegal . delta
)+
(s==StartingUp) ->
( rICR_PC(powerOn) . Illegal . delta +
 rICR_PC(powerOff) . ControlPC(PC_Off) +
 rICR_PC_Broadcast(restart) . Illegal . delta +
 rICR_PC_Broadcast(shutdown) . Illegal . delta +
 sICR_PC_CB(started) . ControlPC(Operational)
)
);


proc GeoPC(s:PCState) = (
(s==PC_Off) ->
( rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC(powerOn) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC(powerOff) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==Operational) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . GeoPC(Operational) + % it can happen if the restart is issued before
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 sIGeoPC_CB(stop) . GeoPC(StopPressed)
)+
(s==StopPressed) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . GeoPC(Operational)
)+
(s==OS_Shutdown) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(StartingUp) +
 rIGeoPC_Broadcast(restart) . Illegal . delta +
 sIGeoPC_CB(started) . GeoPC(Operational)
)
);

proc NormalPC(id:Pos,s:PCState) = (
```

```
(s==PC_Off) ->
( rIPC(id,powerOn) . NormalPC(id,StartingUp) +
 rIPC(id,powerOff) . Illegal . delta +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==Operational) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
 rIPC_Broadcast(id,restart) . NormalPC(id,StartingUp)
)+
(s==OS_Shutdown) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,StartingUp) +
 rIPC_Broadcast(id,restart) . Illegal . delta +
 sIPC_CB(id,started) . NormalPC(id,Operational)
) );

% the PDU design

proc PDU_State_Machine(s:PDUState,geopcOn,crpcOn,geoPressed:Bool,startedPc:Nat) = (
(s==PDU_Off) ->
( IPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
                     PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                          PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==StartingUp_CR_PC) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) .
                       PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOff) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                       PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
                      PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . Illegal . delta +
 rICR_PC_CB(started) . IndicationCB(systemStandby) .
                       PDU_State_Machine(SystemStandby,geopcOn,true,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                       PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==SystemStandby) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) .
                            PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease . IndicationCB(startingUp) .
                                    PDU_State_Machine(StartingUpAllPcs,false,true,false,0) +
 IPDU(powerOff) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                         PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
```

```
                          PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sICR_PC(powerOff) .
         IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . Illegal . delta +
 rIGeoPC_CB(started) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==System_On) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
                      rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) . sMsghandler(restart) .
    rrelease . IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
 IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
      sIGeoPC(powerOff) . scommandhandler(powerOff) .  rrelease . IndicationCB(systemStandby) .
                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
 rrelease . IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) .
                scommandhandler(powerOff) .  rrelease . IndicationCB(off) .
                           PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
                 sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) . sIGeoPC(powerOff) .
        scommandhandler(powerOff) . rrelease . IndicationCB(off) .
                                PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . IndicationCB(geoStop) .
                                  PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . Illegal . delta +
 rIGeoPC_CB(started) . Illegal . delta +
 sum id:Pos . rIPC_CB(id,started) . Illegal . delta
)+
(s==Emergency_Off) ->
(
 IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
                IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
 IPDU(powerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                             PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==System_Off) ->
(
 IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
       IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
 IPDU(powerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                            PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==Geo_Stop) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
                       rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
```

```
237   IPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
238                                      PDU_State_Machine(System_On,geopcOn,crpcOn,false,startedPc) +
239   IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
240                       sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
241                     IndicationCB(systemStandby) .
242                                PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
243   IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
244                                rrelease . IndicationCB(off) .
245                                PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
246   IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
247             rrelease . IndicationCB(off) .
248                         PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
249   rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
250               sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) . sIGeoPC(powerOff) .
251                 scommandhandler(powerOff) . rrelease . IndicationCB(off) .
252                               PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
253   rIGeoPC_CB(stop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
254   rICR_PC_CB(started) . Illegal . delta +
255   rIGeoPC_CB(started) . Illegal . delta +
256   sum id:Pos . rIPC_CB(id,started) . Illegal . delta
257  )+
258  (s==StartingUpAllPcs) ->
259  (
260   IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
261                           rrelease . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
262   IPDU(powerOn) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
263   (crpcOn ) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
264               rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) .
265                                  rrelease . IndicationCB(systemStandby) .
266                   PDU_State_Machine(SystemStandby,false,crpcOn,false,0) +
267   (! crpcOn ) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
268               rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) .  rrelease .
269                   PDU_State_Machine(StartingUp_CR_PC,false,crpcOn,false,0) +
270   IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
271      rrelease . IndicationCB(off) .
272                      PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
273   IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
274                    rrelease . IndicationCB(off) .
275                       PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
276   (crpcOn ) -> rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) .
277                  sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
278              sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
279                           IndicationCB(off) .
280                           PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
281   (! crpcOn ) -> rICR_PC_CB(controlPowerOff) .
282                            PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
283   ((geopcOn && startedPc==5 && geoPressed)) -> rICR_PC_CB(started) . IndicationCB(geoStop) .
284                                    PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
285   ((geopcOn && startedPc==5 && ! geoPressed)) -> rICR_PC_CB(started) . IndicationCB(systemOn) .
286                                       PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
287   ((! geopcOn || startedPc<5)) -> rICR_PC_CB(started) .
288                                  PDU_State_Machine(StartingUpAllPcs,geopcOn,true,geoPressed,startedPc) +
289   rIGeoPC_CB(stop) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,true,startedPc)+
290   (! geopcOn && (startedPc<5 || ! crpcOn)) -> rIGeoPC_CB(started) .
291                                    PDU_State_Machine(StartingUpAllPcs,true,crpcOn,geoPressed,startedPc) +
292   (crpcOn && startedPc==5 && ! geoPressed) -> rIGeoPC_CB(started) . IndicationCB(systemOn) .
293                                      PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
294   (crpcOn && startedPc==5 && geoPressed) -> rIGeoPC_CB(started) . IndicationCB(geoStop) .
295                                       PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
296   (startedPc<4) -> sum id:Pos . rIPC_CB(id,started) .
297                                  PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1) +
298   (startedPc==4 && geopcOn && crpcOn && ! geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
299            IndicationCB(systemOn) . PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
300   (startedPc==4 && geopcOn && crpcOn && geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
301               IndicationCB(geoStop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
302   ((startedPc==4) && (geopcOn==false || ! crpcOn)) -> sum id:Pos . rIPC_CB(id,started) .
303                              PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1)
```

```
)
);

proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) . sIPC(2,c) . sIPC(3,c) .
                                  sIPC(4,c) .srelease|sIPC(5,c) . Handler +
               sum m:Bmsg    . rMsghandler(m) | sIPC_Broadcast(1,m) . sIPC_Broadcast(2,m) .
                sIPC_Broadcast(3,m) . sIPC_Broadcast(4,m) . srelease|sIPC_Broadcast(5,m) . Handler;

proc System=
      hide({ ICR_PC,ICR_PC_Broadcast,ICR_PC_CB,
             IGeoPC,IGeoPC_Broadcast,IGeoPC_CB,
             IPC,IPC_Broadcast,IPC_CB,
                 release,Msghandler,commandhandler
           },
        allow({IPDU,
                   ICR_PC,ICR_PC_Broadcast,ICR_PC_CB,
               IGeoPC,IGeoPC_Broadcast,IGeoPC_CB,
               IPC,IPC_Broadcast, IPC_CB,
               IndicationCB,
               commandhandler|IPC,
               Msghandler|IPC_Broadcast,
               release|IPC,release|IPC_Broadcast,
               Illegal
              },
         comm({sICR_PC|rICR_PC->ICR_PC,
               sICR_PC_Broadcast|rICR_PC_Broadcast->ICR_PC_Broadcast,
               sIGeoPC|rIGeoPC->IGeoPC,
               sIGeoPC_Broadcast|rIGeoPC_Broadcast->IGeoPC_Broadcast,
               sIGeoPC_CB|rIGeoPC_CB->IGeoPC_CB,
               sICR_PC_CB|rICR_PC_CB->ICR_PC_CB,
               sIPC_CB | rIPC_CB -> IPC_CB,
               srelease | rrelease ->release,
               rIPC|sIPC->IPC,
               rIPC_Broadcast|sIPC_Broadcast->IPC_Broadcast,

               sMsghandler | rMsghandler -> Msghandler,
               scommandhandler | rcommandhandler -> commandhandler

             },
                        ControlPC(PC_Off)  ||
                          GeoPC(PC_Off)      ||
                          NormalPC(1,PC_Off) ||
                          NormalPC(2,PC_Off) ||
                          NormalPC(3,PC_Off) ||
                          NormalPC(4,PC_Off) ||
                          NormalPC(5,PC_Off) ||
                         Handler                   ||
                         PDU_State_Machine(PDU_Off,false,false,false,0))));

init System;
```

E Synchronous push model with global synchronous communication

```

sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff |
                                                   emergencyOff | onPressed;
Bmsg = struct restart | shutdown ;
PCsCallbacks = struct controlPowerOff | stop | started ;
PCState = struct PC_Off | Operational | WaitingShutdown | StartingUp | OS_Shutdown | StopPressed;
PDUState = struct PDU_Off | StartingUp_CR_PC | StartingUpAllPcs | SystemStandby | System_On |
                                              Emergency_Off | System_Off | Geo_Stop ;
IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;

act Illegal,
```

```
12  srelease,rrelease,release;
13  IPDU,
14  sICR_PC,rICR_PC,ICR_PC,
15  sIGeoPC,rIGeoPC,IGeoPC,
16   scommandhandler, rcommandhandler, commandhandler : Command;
17  sIPC,rIPC,IPC : Pos # Command;
18  sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
19  sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
20  sMsghandler, rMsghandler, Msghandler: Bmsg;
21  sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;
22  sICR_PC_CB,rICR_PC_CB,ICR_PC_CB,
23  sIGeoPC_CB,rIGeoPC_CB,IGeoPC_CB: PCsCallbacks;
24  sIPC_CB,rIPC_CB,IPC_CB : Pos # PCsCallbacks;
25  IndicationCB : IndicationMsg;
26
27  proc ControlPC(s:PCState) = (
28  (s==PC_Off) ->
29  ( rICR_PC(powerOn) . ControlPC(StartingUp) +
30   rICR_PC(powerOff) . Illegal . delta +
31   rICR_PC_Broadcast(restart) . Illegal . delta +
32   rICR_PC_Broadcast(shutdown) . Illegal . delta
33  ) +
34  (s==Operational) ->
35  ( rICR_PC(powerOn) . Illegal . delta +
36   rICR_PC(powerOff) . ControlPC(PC_Off) +
37   sICR_PC_CB(controlPowerOff) . ControlPC(WaitingShutdown) +
38   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
39   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
40  )+
41  (s==WaitingShutdown) ->
42  ( rICR_PC(powerOn) . Illegal . delta +
43   rICR_PC(powerOff) . ControlPC(PC_Off) +
44   rICR_PC_Broadcast(restart) . ControlPC(StartingUp) +
45   rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
46  )+
47  (s==OS_Shutdown) -> (
48   rICR_PC(powerOn) . Illegal . delta +
49   rICR_PC(powerOff) . ControlPC(PC_Off) +
50   rICR_PC_Broadcast(restart) . Illegal . delta +
51   rICR_PC_Broadcast(shutdown) . Illegal . delta
52  )+
53  (s==StartingUp) ->
54  ( rICR_PC(powerOn) . Illegal . delta +
55   rICR_PC(powerOff) . ControlPC(PC_Off) +
56   rICR_PC_Broadcast(restart) . Illegal . delta +
57   rICR_PC_Broadcast(shutdown) . Illegal . delta +
58   sICR_PC_CB(started) . ControlPC(Operational)
59  )
60  );
61
62
63  proc GeoPC(s:PCState) = (
64  (s==PC_Off) ->
65  ( rIGeoPC_Broadcast(shutdown) . Illegal . delta +
66   rIGeoPC(powerOn) . GeoPC(StartingUp) +
67   rIGeoPC(onPressed) . Illegal . delta +
68   rIGeoPC(powerOff) . Illegal . delta +
69   rIGeoPC_Broadcast(restart) . Illegal . delta
70  )+
71  (s==Operational) ->
72  ( rIGeoPC(powerOn) . Illegal . delta +
73   rIGeoPC(powerOff) . GeoPC(PC_Off) +
74   rIGeoPC(onPressed) . GeoPC(Operational) + % it can happen if the restart is issued before
75   rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
76   rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
77   sIGeoPC_CB(stop) . GeoPC(StopPressed)
78  )+
```

```
(s==StopPressed) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
 rIGeoPC_Broadcast(restart) . GeoPC(StartingUp) +
 rIGeoPC(onPressed) . GeoPC(Operational)
)+
(s==OS_Shutdown) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC(onPressed) . Illegal . delta +
 rIGeoPC_Broadcast(shutdown) . Illegal . delta +
 rIGeoPC_Broadcast(restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIGeoPC(powerOn) . Illegal . delta +
 rIGeoPC(powerOff) . GeoPC(PC_Off) +
 rIGeoPC_Broadcast(shutdown) . GeoPC(StartingUp) +
 rIGeoPC_Broadcast(restart) . Illegal . delta +
 sIGeoPC_CB(started) . GeoPC(Operational)
)
);

proc NormalPC(id:Pos,s:PCState) = (
(s==PC_Off) ->
( rIPC(id,powerOn) . NormalPC(id,StartingUp) +
 rIPC(id,powerOff) . Illegal . delta +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==Operational) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
 rIPC_Broadcast(id,restart) . NormalPC(id,StartingUp)
)+
(s==OS_Shutdown) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . Illegal . delta +
 rIPC_Broadcast(id,restart) . Illegal . delta
)+
(s==StartingUp) ->
( rIPC(id,powerOn) . Illegal . delta +
 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
 rIPC_Broadcast(id,shutdown) . NormalPC(id,StartingUp) +
 rIPC_Broadcast(id,restart) . Illegal . delta +
 sIPC_CB(id,started) . NormalPC(id,Operational)
) );

% the PDU design

proc PDU_State_Machine(s:PDUState,geopcOn,crpcOn,geoPressed:Bool,startedPc:Nat) = (
(s==PDU_Off) ->
( IPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
           PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) .
                    PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
            PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==StartingUp_CR_PC) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) .
```

```
                          PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(powerOn) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(powerOff) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                         PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
                       PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(controlPowerOff) . Illegal . delta +
  rICR_PC_CB(started) . IndicationCB(systemStandby) .
                        PDU_State_Machine(SystemStandby,geopcOn,true,geoPressed,startedPc) +
  rIGeoPC_CB(stop) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
  rIGeoPC_CB(started) . PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc) +
  sum id:Pos . rIPC_CB(id,started) .
                       PDU_State_Machine(StartingUp_CR_PC,geopcOn,crpcOn,geoPressed,startedPc)
 )+
 (s==SystemStandby) ->
 (
  IPDU(PDUswitchOff) . sICR_PC(powerOff) .
                                  PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn)  . IndicationCB(startingUp) .
                                       PDU_State_Machine(StartingUpAllPcs,false,true,false,0) +
  IPDU(powerOff) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
                           PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sICR_PC(powerOff) .
          IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rIGeoPC_CB(stop) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(started) . Illegal . delta +
  rIGeoPC_CB(started) . PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
  sum id:Pos . rIPC_CB(id,started) .
                         PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc)
 )+
 (s==System_On) ->
 (
  IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
                               PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) .
                    sMsghandler(restart) . rrelease . IndicationCB(startingUp) .
                                         PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
  IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
               sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(systemStandby) .
                           PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
            IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
          IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
          sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
           sIGeoPC(powerOff) | scommandhandler(powerOff)  . IndicationCB(off) .
                               PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rIGeoPC_CB(stop) . IndicationCB(geoStop) .
                         PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(started) . Illegal . delta +
  rIGeoPC_CB(started) . Illegal . delta +
  sum id:Pos . rIPC_CB(id,started) . Illegal . delta
 )+
 (s==Emergency_Off) ->
 (
  IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn)  .
             IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
  IPDU(powerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(forcedPowerOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
  rICR_PC_CB(controlPowerOff) .
```

```
                          PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                          PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==System_Off) ->
(
 IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn)  .
            IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPcs,false,false,false,0) +
 IPDU(powerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(started) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 sum id:Pos . rIPC_CB(id,started) .
                           PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc)
)+
(s==Geo_Stop) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
                                  PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
                                    PDU_State_Machine(System_On,geopcOn,crpcOn,false,startedPc) +
 IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
              sIGeoPC(powerOff) | scommandhandler(powerOff)  . IndicationCB(systemStandby) .
                             PDU_State_Machine(SystemStandby,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
            IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
          IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
          sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
           sIGeoPC(powerOff) | scommandhandler(powerOff)  . IndicationCB(off) .
                             PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 rIGeoPC_CB(stop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
 rICR_PC_CB(started) . Illegal . delta +
 rIGeoPC_CB(started) . Illegal . delta +
 sum id:Pos . rIPC_CB(id,started) . Illegal . delta
)+
(s==StartingUpAllPcs) ->
(
 IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
                                    PDU_State_Machine(PDU_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(powerOn) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
 (crpcOn ) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
                  rrelease . sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(systemStandby) .
                PDU_State_Machine(SystemStandby,false,crpcOn,false,0) +
 (! crpcOn ) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
                rrelease . sIGeoPC(powerOff) | scommandhandler(powerOff) .
                 PDU_State_Machine(StartingUp_CR_PC,false,crpcOn,false,0) +
 IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
              IndicationCB(off) . PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff)  .
            IndicationCB(off) . PDU_State_Machine(Emergency_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 (crpcOn ) -> rICR_PC_CB(controlPowerOff) . sICR_PC_Broadcast(shutdown) . sIGeoPC_Broadcast(shutdown) .
            sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
            sIGeoPC(powerOff) | scommandhandler(powerOff)  . IndicationCB(off) .
                                  PDU_State_Machine(System_Off,geopcOn,crpcOn,geoPressed,startedPc) +
 (! crpcOn ) -> rICR_PC_CB(controlPowerOff) .
                           PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc) +
 ((geopcOn && startedPc==5 && geoPressed)) -> rICR_PC_CB(started) . IndicationCB(geoStop) .
                                    PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
```

```
280   ((geopcOn && startedPc==5 && ! geoPressed)) -> rICR_PC_CB(started) .
281                 IndicationCB(systemOn) . PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
282   ((! geopcOn || startedPc<5)) -> rICR_PC_CB(started) .
283                               PDU_State_Machine(StartingUpAllPcs,geopcOn,true,geoPressed,startedPc) +
284   rIGeoPC_CB(stop) . PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,true,startedPc)+
285   (! geopcOn && (startedPc<5 || ! crpcOn)) -> rIGeoPC_CB(started) .
286                               PDU_State_Machine(StartingUpAllPcs,true,crpcOn,geoPressed,startedPc) +
287   (crpcOn && startedPc==5 && ! geoPressed) -> rIGeoPC_CB(started) . IndicationCB(systemOn) .
288                                         PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
289   (crpcOn && startedPc==5 && geoPressed) -> rIGeoPC_CB(started) . IndicationCB(geoStop) .
290                                        PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
291   (startedPc<4) -> sum id:Pos . rIPC_CB(id,started) .
292                                  PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1) +
293   (startedPc==4 && geopcOn && crpcOn && ! geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
294                     IndicationCB(systemOn) . PDU_State_Machine(System_On,geopcOn,crpcOn,geoPressed,startedPc) +
295   (startedPc==4 && geopcOn && crpcOn && geoPressed) -> sum id:Pos . rIPC_CB(id,started) .
296                      IndicationCB(geoStop) . PDU_State_Machine(Geo_Stop,geopcOn,crpcOn,geoPressed,startedPc) +
297   ((startedPc==4) && (geopcOn==false || ! crpcOn)) -> sum id:Pos . rIPC_CB(id,started) .
298                                  PDU_State_Machine(StartingUpAllPcs,geopcOn,crpcOn,geoPressed,startedPc+1)
299 )
300 );
301
302 proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) | sIPC(2,c) | sIPC(3,c) | sIPC(4,c) |
303                           sIPC(5,c) . Handler +
304                 sum m:Bmsg    . rMsghandler(m) | sIPC_Broadcast(1,m) . sIPC_Broadcast(2,m) .
305                 sIPC_Broadcast(3,m) . sIPC_Broadcast(4,m) . srelease|sIPC_Broadcast(5,m) . Handler;
306
307 proc System=
308       hide({ ICR_PC,ICR_PC_Broadcast,ICR_PC_CB,
309              IGeoPC,IGeoPC_Broadcast,IGeoPC_CB,
310              IPC,IPC_Broadcast,IPC_CB,
311                  release,Msghandler,commandhandler
312             },
313         allow({IPDU,
314                     ICR_PC,ICR_PC_Broadcast,ICR_PC_CB,
315                IGeoPC,IGeoPC_Broadcast,IGeoPC_CB,
316                IPC,IPC_Broadcast, IPC_CB,
317                IndicationCB,
318                IGeoPC|commandhandler|IPC|IPC|IPC|IPC|IPC ,
319                Msghandler|IPC_Broadcast,
320                release|IPC_Broadcast,
321                Illegal
322               },
323          comm({sICR_PC|rICR_PC->ICR_PC,
324                sICR_PC_Broadcast|rICR_PC_Broadcast->ICR_PC_Broadcast,
325                sIGeoPC|rIGeoPC->IGeoPC,
326                sIGeoPC_Broadcast|rIGeoPC_Broadcast->IGeoPC_Broadcast,
327                sIGeoPC_CB|rIGeoPC_CB->IGeoPC_CB,
328                sICR_PC_CB|rICR_PC_CB->ICR_PC_CB,
329                sIPC_CB | rIPC_CB -> IPC_CB,
330                srelease | rrelease ->release,
331                rIPC|sIPC->IPC,
332                rIPC_Broadcast|sIPC_Broadcast->IPC_Broadcast,
333
334                sMsghandler | rMsghandler -> Msghandler,
335                scommandhandler | rcommandhandler -> commandhandler
336
337              },
338                         ControlPC(PC_Off)  ||
339                           GeoPC(PC_Off)      ||
340                           NormalPC(1,PC_Off) ||
341                           NormalPC(2,PC_Off) ||
342                           NormalPC(3,PC_Off) ||
343                           NormalPC(4,PC_Off) ||
344                           NormalPC(5,PC_Off) ||
345                          Handler                   ||
346                          PDU_State_Machine(PDU_Off,false,false,false,0))));
```

```
347
348 init System;
```

F Poll model

```
1  sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff |
2            emergencyOff | onPressed | queryCRPCStatus | queryCRPCPowerOffFlag |
3                 queryGeoStopFlag | queryGeoPCstatus | queryPCstatus  ;
4
5  sort TimerData = struct pollGeoPC | pollCRPC |pollPC;
6
7  Bmsg = struct restart | shutdown ;
8
9  PCState = struct PC_Off | PC_On | WaitingShutdown | OS_Shutdown | StopPressed;
10
11 RetVal = struct IsOperational | IsNotOperational | IsOn | IsOff | StopIsPressed | StopIsNotPressed;
12
13 PDUState = struct PDU_Off | StartingUpCrPC | StartingUpAllPCs | System_Standby | System_On |
14                Emergency_Off | System_Off | Geo_Stop | none | WaitingCRPCReply |
15                WaitingCRPCStatusReply | CheckingGeoStopStatus | CheckingCRPCFlag |
16                WaitingGeoPCStatusReply | WaitingPC1statusReply | WaitingPC2statusReply |
17                WaitingPC3statusReply | WaitingPC4statusReply | WaitingPC5statusReply;
18
19 IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;
20
21 PCs = struct CRPC | GeoPC | NormalPC ;
22
23 act Illegal,
24 srelease,rrelease,release;
25 IPDU,
26 sICR_PC,rICR_PC,ICR_PC,
27 sIGeoPC,rIGeoPC,IGeoPC,
28 scommandhandler, rcommandhandler, commandhandler : Command;
29
30 IPDUTimer: TimerData;
31
32 sICR_PCrVal,rICR_PCrVal,ICR_PCrVal,
33 sIGeoPCrVal,rIGeoPCrVal,IGeoPCrVal : RetVal;
34 sIPCrVal,rIPCrVal,IPCrVal : Pos # RetVal;
35
36 sIPC,rIPC,IPC : Pos # Command;
37
38 sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
39 sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
40 sMsghandler, rMsghandler, Msghandler: Bmsg;
41
42 sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;
43
44 IndicationCB : IndicationMsg;
45
46 proc ControlPC(s:PCState) = (
47         (s==PC_Off) -> (
48                 rICR_PC(powerOn) . ControlPC(PC_On) +
49                  rICR_PC(powerOff) . Illegal . delta +
50                  rICR_PC(queryCRPCStatus) . Illegal . delta +
51                  rICR_PC(queryCRPCPowerOffFlag) . Illegal . delta +
52                  rICR_PC_Broadcast(restart) . Illegal . delta +
53                  rICR_PC_Broadcast(shutdown) . Illegal . delta
54         )
55         +
56         (s==PC_On) -> (
57                 rICR_PC(powerOn) . Illegal . delta +
58                  rICR_PC(powerOff) . ControlPC(PC_Off) +
59                  rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsOperational) . ControlPC(PC_On) +
60                  rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsNotOperational) . ControlPC(PC_On) +
61                  rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOn) . ControlPC(PC_On) +
```

```
                 rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(PC_On)+
                 rICR_PC_Broadcast(restart) . ControlPC(PC_On) +
                 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
        )
        +
        (s==OS_Shutdown) -> (
                rICR_PC(powerOn) . Illegal . delta +
                 rICR_PC(powerOff) . ControlPC(PC_Off) +
                 rICR_PC(queryCRPCStatus) . Illegal . delta +
                 rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(OS_Shutdown) +
                 rICR_PC_Broadcast(restart) . ControlPC(OS_Shutdown) +
                 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
        )
);

proc GeoPC(s:PCState) = (
        (s==PC_Off) -> (
                 rIGeoPC(powerOn) . GeoPC(PC_On) +
                 rIGeoPC(powerOff) . Illegal . delta +
                 rIGeoPC(queryGeoStopFlag) . Illegal . delta +
                 rIGeoPC(queryGeoPCstatus) . Illegal . delta +
                 rIGeoPC(onPressed) . Illegal . delta +
                rIGeoPC_Broadcast(shutdown) . Illegal . delta +
                 rIGeoPC_Broadcast(restart) . Illegal . delta
        )
        +
        (s==PC_On) -> (
                 rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsNotPressed) . GeoPC(PC_On) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsPressed) . GeoPC(StopPressed) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsOperational) . GeoPC(PC_On) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsNotOperational) . GeoPC(PC_On) +
                 rIGeoPC(onPressed) . Illegal . delta +
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(PC_On)
        )
        +
        (s==OS_Shutdown) -> (
                 rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . Illegal . delta +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsNotOperational) . GeoPC(OS_Shutdown) +
                 rIGeoPC(onPressed) . Illegal . delta +
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(OS_Shutdown)
        )
        +
        (s==StopPressed) -> (
          rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsPressed) . GeoPC(StopPressed) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsOperational) . GeoPC(StopPressed) +
                 rIGeoPC(onPressed) . GeoPC(PC_On) +
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(StopPressed)
        )
);

proc NormalPC(id:Pos,s:PCState) = (
        (s==PC_Off) -> (
                rIPC(id,powerOn) . NormalPC(id,PC_On) +
                 rIPC(id,powerOff) . Illegal . delta +
                 rIPC(id,queryPCstatus) . Illegal . delta +
                 rIPC_Broadcast(id,shutdown) . Illegal . delta +
                 rIPC_Broadcast(id,restart) . Illegal . delta
        )
```

```
129         +
130         (s==PC_On) -> (
131                 rIPC(id,powerOn) . Illegal . delta +
132                  rIPC(id,powerOff) . NormalPC(id,PC_Off) +
133                  rIPC(id,queryPCstatus) . sIPCrVal(id,IsOperational) . NormalPC(id,PC_On) +
134                  rIPC(id,queryPCstatus) . sIPCrVal(id,IsNotOperational) . NormalPC(id,PC_On) +
135                  rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
136                  rIPC_Broadcast(id,restart) . NormalPC(id,PC_On)
137         )
138         +
139         (s==OS_Shutdown) -> (
140                 rIPC(id,powerOn) . Illegal . delta +
141                  rIPC(id,powerOff) . NormalPC(id,PC_Off) +
142                  rIPC(id,queryPCstatus) . sIPCrVal(id,IsNotOperational) . NormalPC(id,OS_Shutdown) +
143                  rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
144                  rIPC_Broadcast(id,restart) . NormalPC(id,OS_Shutdown)
145         )
146 );
147
148 % the PDU design
149 proc PDU_State_Machine(s:PDUState,cRPCstarted,geoPCstarted,geoPressed:Bool,state:PDUState) = (
150 (s==PDU_Off) -> (
151         IPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
152                         PDU_State_Machine(StartingUpCrPC,false,false,false,none)
153         )
154         +
155 (s==StartingUpCrPC) -> (
156        IPDU(PDUswitchOff) . sICR_PC(powerOff) .
157                         PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
158        IPDU(powerOn) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
159        IPDU(powerOff) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
160        IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
161                 PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
162        IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
163                 PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
164        IPDUTimer(pollPC) . sICR_PC(queryCRPCStatus) .
165                  PDU_State_Machine(WaitingCRPCReply,cRPCstarted,geoPCstarted,geoPressed,state)
166         )
167         +
168 (s==System_Off) -> (
169        IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
170        IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
171             IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
172        IPDU(powerOff) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
173         IPDU(forcedPowerOff) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
174         IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state)
175         )
176         +
177 (s==Emergency_Off) -> (
178         IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
179          IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
180              IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
181          IPDU(powerOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
182          IPDU(forcedPowerOff) .
183                     PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
184          IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state)
185         )
186         +
187 (s==WaitingCRPCReply) -> (
188         rICR_PCrVal(IsOperational) . IndicationCB(systemStandby) .
189                 PDU_State_Machine(System_Standby,true,geoPCstarted,geoPressed,state) +
190         rICR_PCrVal(IsNotOperational) .
191                      PDU_State_Machine(StartingUpCrPC,false,geoPCstarted,geoPressed,state)
192         )
193         +
194 (s==StartingUpAllPCs) -> (
195          IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
```

```
196             rrelease . PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
197         IPDU(powerOn) .
198                     PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state) +
199         (cRPCstarted) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
200           rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
201            IndicationCB(systemStandby) .
202                             PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,false,none) +
203         (!cRPCstarted) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
204            rrelease . sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease .
205                 PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,false,none) +
206          IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
207             rrelease . IndicationCB(off) .
208                         PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
209          IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
210             rrelease . IndicationCB(off) .
211                          PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
212          IPDUTimer(pollPC) . sICR_PC(queryCRPCStatus) .
213                 PDU_State_Machine(WaitingCRPCStatusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
214          (geoPCstarted) -> IPDUTimer(pollGeoPC) . sIGeoPC(queryGeoStopFlag) .
215         PDU_State_Machine(CheckingGeoStopStatus,cRPCstarted,geoPCstarted,geoPressed,StartingUpAllPCs) +
216          (!geoPCstarted) -> IPDUTimer(pollGeoPC) .
217                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state) +
218          (cRPCstarted) -> IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
219              PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,StartingUpAllPCs) +
220          (!cRPCstarted) -> IPDUTimer(pollCRPC) .
221                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
222         )
223         +
224   (s==System_Standby) -> (
225         IPDU(PDUswitchOff) . sICR_PC(powerOff) .
226                 PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
227          IPDU(powerOn) . sIGeoPC(powerOn) . scommandhandler(powerOn) . rrelease .
228                 IndicationCB(startingUp) .
229                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)  +
230          IPDU(powerOff) . PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,geoPressed,state) +
231          IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
232                 PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
233          IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
234                 PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
235         IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
236                 PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,System_Standby)
237         )
238         +
239   (s==WaitingCRPCStatusReply) -> (
240         rICR_PCrVal(IsOperational) . sIGeoPC(queryGeoPCstatus) .
241                 PDU_State_Machine(WaitingGeoPCStatusReply,true,geoPCstarted,geoPressed,state) +
242         rICR_PCrVal(IsNotOperational) .
243                         PDU_State_Machine(StartingUpAllPCs,false,geoPCstarted,geoPressed,state)
244         )
245         +
246   (s==CheckingGeoStopStatus) -> (
247         (state==StartingUpAllPCs) -> rIGeoPCrVal(StopIsPressed) .
248                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,true,state) +
249         (state==System_On) -> rIGeoPCrVal(StopIsPressed) . IndicationCB(geoStop) .
250                 PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,true,state) +
251         (state==StartingUpAllPCs) -> rIGeoPCrVal(StopIsNotPressed) .
252                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,false,state) +
253         (state==System_On) -> rIGeoPCrVal(StopIsNotPressed) .
254                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,false,state)
255         )
256         +
257   (s==CheckingCRPCFlag) -> (
258         (state!=System_Standby) -> rICR_PCrVal(IsOn) . sICR_PC_Broadcast(shutdown) .
259                 sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
260                 sIGeoPC(powerOff) . scommandhandler(powerOff) . rrelease . IndicationCB(off) .
261                  PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
262         (state==System_Standby) -> rICR_PCrVal(IsOn) . sICR_PC_Broadcast(shutdown) .
```

```
263                 sICR_PC(powerOff) . IndicationCB(off) .
264                 PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
265         (state==Geo_Stop) -> rICR_PCrVal(IsOff) .
266                 PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,geoPressed,none) +
267         (state==System_On) -> rICR_PCrVal(IsOff) .
268                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,none) +
269         (state==StartingUpAllPCs) -> rICR_PCrVal(IsOff) .
270                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,none) +
271         (state==System_Standby) -> rICR_PCrVal(IsOff) .
272                 PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,geoPressed,none)
273         )
274         +
275   (s==WaitingGeoPCStatusReply) -> (
276         rIGeoPCrVal(IsOperational) . sIPC(1,queryPCstatus) .
277                 PDU_State_Machine(WaitingPC1statusReply,cRPCstarted,true,geoPressed,state) +
278         rIGeoPCrVal(IsNotOperational) .
279                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,false,geoPressed,state)
280         )
281         +
282   (s==Geo_Stop) -> (
283          IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
284                 rrelease .
285                 PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
286          IPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
287                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,state) +
288          IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
289                 sIGeoPC(powerOff) .
290                  scommandhandler(powerOff) . rrelease . IndicationCB(systemStandby) .
291                 PDU_State_Machine(System_Standby,cRPCstarted,false,false,state) +
292          IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
293                 rrelease .
294                 IndicationCB(off) .
295                 PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
296          IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
297                 rrelease .
298                  IndicationCB(off) .
299                 PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
300         IPDUTimer(pollCRPC) .
301                 sICR_PC(queryCRPCPowerOffFlag) .
302                 PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,Geo_Stop)
303         )
304         +
305   (s==System_On) -> (
306         IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
307                 rrelease .
308                 PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
309          IPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) . sMsghandler(restart) .
310                 rrelease . IndicationCB(startingUp) .
311                                     PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
312          IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) .
313                 rrelease . sIGeoPC(powerOff) .
314                 scommandhandler(powerOff) .  rrelease . IndicationCB(systemStandby) .
315                 PDU_State_Machine(System_Standby,cRPCstarted,false,false,state) +
316         IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
317                  rrelease .
318                 IndicationCB(off) .
319                        PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
320         IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) . scommandhandler(powerOff) .
321                 rrelease .  IndicationCB(off) .
322                    PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
323         IPDUTimer(pollGeoPC) . sIGeoPC(queryGeoStopFlag) .
324                 PDU_State_Machine(CheckingGeoStopStatus,cRPCstarted,geoPCstarted,geoPressed,System_On) +
325          IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
326                 PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,System_On)
327         )
328         +
329   (s==WaitingPC1statusReply) -> (
```

```
330         rIPCrVal(1,IsOperational) . sIPC(2,queryPCstatus) .
331                 PDU_State_Machine(WaitingPC2statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
332         rIPCrVal(1,IsNotOperational) .
333                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
334         )
335         +
336   (s==WaitingPC2statusReply) -> (
337         rIPCrVal(2,IsOperational) . sIPC(3,queryPCstatus) .
338                 PDU_State_Machine(WaitingPC3statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
339         rIPCrVal(2,IsNotOperational) .
340                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
341         )
342         +
343   (s==WaitingPC3statusReply) -> (
344         rIPCrVal(3,IsOperational) . sIPC(4,queryPCstatus) .
345                 PDU_State_Machine(WaitingPC4statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
346         rIPCrVal(3,IsNotOperational) .
347                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
348         )
349         +
350   (s==WaitingPC4statusReply) -> (
351         rIPCrVal(4,IsOperational) . sIPC(5,queryPCstatus) .
352                 PDU_State_Machine(WaitingPC5statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
353         rIPCrVal(4,IsNotOperational) .
354                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
355         )
356         +
357   (s==WaitingPC5statusReply) -> (
358         (!geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(systemOn) .
359                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,state) +
360         (geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(geoStop) .
361                 PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,geoPressed,state) +
362         rIPCrVal(5,IsNotOperational) .
363                          PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
364         )
365   );
366
367   proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) . sIPC(2,c) . sIPC(3,c) .sIPC(4,c) .
368         srelease | sIPC(5,c) .  Handler +
369                sum m:Bmsg . rMsghandler(m) | sIPC_Broadcast(1,m) . sIPC_Broadcast(2,m) .
370         sIPC_Broadcast(3,m) . sIPC_Broadcast(4,m) . srelease|sIPC_Broadcast(5,m) . Handler;
371
372   proc System=
373        hide({  IPDUTimer,ICR_PC,ICR_PC_Broadcast,
374                IGeoPC,IGeoPC_Broadcast,
375                IPC,IPC_Broadcast,
376                ICR_PCrVal,
377                IGeoPCrVal,
378                IPCrVal ,
379                release,Msghandler,commandhandler
380
381             },
382         allow({IPDU,IPDUTimer,
383                ICR_PC,ICR_PC_Broadcast,
384                IGeoPC,IGeoPC_Broadcast,
385                IPC   ,IPC_Broadcast   ,
386                 ICR_PCrVal,
387                 IGeoPCrVal,
388                 IPCrVal ,
389                        IndicationCB,
390                        commandhandler|IPC,Msghandler|IPC_Broadcast,
391                 release|IPC,
392                 release|IPC_Broadcast,
393                release,
394                        Illegal
395               },
396           comm({sICR_PC | rICR_PC -> ICR_PC,
```

```
                sICR_PC_Broadcast | rICR_PC_Broadcast -> ICR_PC_Broadcast,
                sIGeoPC | rIGeoPC->IGeoPC,
                sIGeoPC_Broadcast | rIGeoPC_Broadcast -> IGeoPC_Broadcast,
                srelease | rrelease -> release,
                            rIPC | sIPC -> IPC,
                rIPC_Broadcast | sIPC_Broadcast -> IPC_Broadcast,
                sMsghandler | rMsghandler -> Msghandler,
                        scommandhandler | rcommandhandler -> commandhandler,
                 sICR_PCrVal | rICR_PCrVal -> ICR_PCrVal,
                 sIGeoPCrVal | rIGeoPCrVal -> IGeoPCrVal,
                 sIPCrVal | rIPCrVal -> IPCrVal
              },
                           ControlPC(PC_Off)  ||
                            GeoPC(PC_Off)      ||
                            NormalPC(1,PC_Off) ||
                            NormalPC(2,PC_Off) ||
                            NormalPC(3,PC_Off) ||
                            NormalPC(4,PC_Off) ||
                            NormalPC(5,PC_Off) ||
                           Handler                  ||
                           PDU_State_Machine(PDU_Off,false,false,false,none))));

init System;
```

G Poll model with global synchronous communication

```

sort Command = struct PDUswitchOn | PDUswitchOff | powerOn | powerOff | forcedPowerOff |
            emergencyOff | onPressed | queryCRPCStatus | queryCRPCPowerOffFlag |
            queryGeoStopFlag | queryGeoPCstatus | queryPCstatus  ;

sort TimerData = struct pollGeoPC | pollCRPC |pollPC;

Bmsg = struct restart | shutdown ;

PCState = struct PC_Off | PC_On | WaitingShutdown | OS_Shutdown | StopPressed;

RetVal = struct IsOperational | IsNotOperational | IsOn | IsOff | StopIsPressed | StopIsNotPressed;

PDUState = struct PDU_Off | StartingUpCrPC | StartingUpAllPCs | System_Standby | System_On |
            Emergency_Off | System_Off | Geo_Stop | none | WaitingCRPCReply |
            WaitingCRPCStatusReply | CheckingGeoStopStatus | CheckingCRPCFlag |
            WaitingGeoPCStatusReply | WaitingPC1statusReply | WaitingPC2statusReply |
            WaitingPC3statusReply | WaitingPC4statusReply | WaitingPC5statusReply;

IndicationMsg = struct startingUp | off | systemOn | systemStandby | geoStop ;

PCs = struct CRPC | GeoPC | NormalPC ;

act Illegal,
srelease,rrelease,release;
IPDU,
sICR_PC,rICR_PC,ICR_PC,
sIGeoPC,rIGeoPC,IGeoPC,
scommandhandler, rcommandhandler, commandhandler : Command;

IPDUTimer: TimerData;

sICR_PCrVal,rICR_PCrVal,ICR_PCrVal,
sIGeoPCrVal,rIGeoPCrVal,IGeoPCrVal : RetVal;
sIPCrVal,rIPCrVal,IPCrVal : Pos # RetVal;

sIPC,rIPC,IPC : Pos # Command;

sICR_PC_Broadcast,rICR_PC_Broadcast,ICR_PC_Broadcast,
```

```
sIGeoPC_Broadcast,rIGeoPC_Broadcast,IGeoPC_Broadcast,
sMsghandler, rMsghandler, Msghandler: Bmsg;

sIPC_Broadcast,rIPC_Broadcast,IPC_Broadcast : Pos # Bmsg;

IndicationCB : IndicationMsg;

proc ControlPC(s:PCState) = (
        (s==PC_Off) -> (
                rICR_PC(powerOn) . ControlPC(PC_On) +
                 rICR_PC(powerOff) . Illegal . delta +
                 rICR_PC(queryCRPCStatus) . Illegal . delta +
                 rICR_PC(queryCRPCPowerOffFlag) . Illegal . delta +
                 rICR_PC_Broadcast(restart) . Illegal . delta +
                 rICR_PC_Broadcast(shutdown) . Illegal . delta
        )
        +
        (s==PC_On) -> (
                rICR_PC(powerOn) . Illegal . delta +
                 rICR_PC(powerOff) . ControlPC(PC_Off) +
                 rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsOperational) . ControlPC(PC_On) +
                 rICR_PC(queryCRPCStatus) . sICR_PCrVal(IsNotOperational) . ControlPC(PC_On) +
                 rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOn) . ControlPC(PC_On) +
                 rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(PC_On)+
                 rICR_PC_Broadcast(restart) . ControlPC(PC_On) +
                 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
        )
        +
        (s==OS_Shutdown) -> (
                rICR_PC(powerOn) . Illegal . delta +
                 rICR_PC(powerOff) . ControlPC(PC_Off) +
                 rICR_PC(queryCRPCStatus) . Illegal . delta +
                 rICR_PC(queryCRPCPowerOffFlag) . sICR_PCrVal(IsOff) . ControlPC(OS_Shutdown) +
                 rICR_PC_Broadcast(restart) . ControlPC(OS_Shutdown) +
                 rICR_PC_Broadcast(shutdown) . ControlPC(OS_Shutdown)
        )
);

proc GeoPC(s:PCState) = (
        (s==PC_Off) -> (
                 rIGeoPC(powerOn) . GeoPC(PC_On) +
                 rIGeoPC(powerOff) . Illegal . delta +
                 rIGeoPC(queryGeoStopFlag) . Illegal . delta +
                 rIGeoPC(queryGeoPCstatus) . Illegal . delta +
                 rIGeoPC(onPressed) . Illegal . delta +
                rIGeoPC_Broadcast(shutdown) . Illegal . delta +
                 rIGeoPC_Broadcast(restart) . Illegal . delta
        )
        +
        (s==PC_On) -> (
                 rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsNotPressed) . GeoPC(PC_On) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsPressed) . GeoPC(StopPressed) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsOperational) . GeoPC(PC_On) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsNotOperational) . GeoPC(PC_On) +
                 rIGeoPC(onPressed) . Illegal . delta +
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(PC_On)
        )
        +
        (s==OS_Shutdown) -> (
                 rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . Illegal . delta +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsNotOperational) . GeoPC(OS_Shutdown) +
                 rIGeoPC(onPressed) . Illegal . delta +
```

```
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(OS_Shutdown)
        )
        +
        (s==StopPressed) -> (
          rIGeoPC(powerOn) . Illegal . delta +
                 rIGeoPC(powerOff) . GeoPC(PC_Off) +
                 rIGeoPC(queryGeoStopFlag) . sIGeoPCrVal(StopIsPressed) . GeoPC(StopPressed) +
                 rIGeoPC(queryGeoPCstatus) . sIGeoPCrVal(IsOperational) . GeoPC(StopPressed) +
                 rIGeoPC(onPressed) . GeoPC(PC_On) +
                rIGeoPC_Broadcast(shutdown) . GeoPC(OS_Shutdown) +
                 rIGeoPC_Broadcast(restart) . GeoPC(StopPressed)
        )
);

proc NormalPC(id:Pos,s:PCState) = (
        (s==PC_Off) -> (
                rIPC(id,powerOn) . NormalPC(id,PC_On) +
                 rIPC(id,powerOff) . Illegal . delta +
                 rIPC(id,queryPCstatus) . Illegal . delta +
                 rIPC_Broadcast(id,shutdown) . Illegal . delta +
                 rIPC_Broadcast(id,restart) . Illegal . delta
        )
        +
        (s==PC_On) -> (
                rIPC(id,powerOn) . Illegal . delta +
                 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
                 rIPC(id,queryPCstatus) . sIPCrVal(id,IsOperational) . NormalPC(id,PC_On) +
                 rIPC(id,queryPCstatus) . sIPCrVal(id,IsNotOperational) . NormalPC(id,PC_On) +
                 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
                 rIPC_Broadcast(id,restart) . NormalPC(id,PC_On)
        )
        +
        (s==OS_Shutdown) -> (
                rIPC(id,powerOn) . Illegal . delta +
                 rIPC(id,powerOff) . NormalPC(id,PC_Off) +
                 rIPC(id,queryPCstatus) . sIPCrVal(id,IsNotOperational) . NormalPC(id,OS_Shutdown) +
                 rIPC_Broadcast(id,shutdown) . NormalPC(id,OS_Shutdown) +
                 rIPC_Broadcast(id,restart) . NormalPC(id,OS_Shutdown)
        )
);

% the PDU design

proc PDU_State_Machine(s:PDUState,cRPCstarted,geoPCstarted,geoPressed:Bool,state:PDUState) = (
(s==PDU_Off) -> (
        IPDU(PDUswitchOn) . sICR_PC(powerOn) . IndicationCB(startingUp) .
                PDU_State_Machine(StartingUpCrPC,false,false,false,none)
)
+
(s==StartingUpCrPC) -> (
         IPDU(PDUswitchOff) . sICR_PC(powerOff) .
                PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDU(powerOn) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDU(powerOff) . PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
                PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
                PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDUTimer(pollPC) . sICR_PC(queryCRPCStatus) .
                PDU_State_Machine(WaitingCRPCReply,cRPCstarted,geoPCstarted,geoPressed,state)
)
+
(s==System_Off) -> (
        IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
        IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) .
                IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
```

```
174          IPDU(powerOff) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
175          IPDU(forcedPowerOff) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
176          IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state)
177  )
178  +
179  (s==Emergency_Off) -> (
180          IPDU(PDUswitchOff) . PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
181           IPDU(powerOn) . sICR_PC(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) .
182                  IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
183           IPDU(powerOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
184           IPDU(forcedPowerOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
185           IPDU(emergencyOff) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state)
186  )
187  +
188  (s==WaitingCRPCReply) -> (
189          rICR_PCrVal(IsOperational) . IndicationCB(systemStandby) .
190                  PDU_State_Machine(System_Standby,true,geoPCstarted,geoPressed,state) +
191          rICR_PCrVal(IsNotOperational) . PDU_State_Machine(StartingUpCrPC,false,geoPCstarted,geoPressed,state)
192  )
193  +
194  (s==StartingUpAllPCs) -> (
195          IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
196                  PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
197          IPDU(powerOn) . PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state) +
198          (cRPCstarted) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
199                  sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(systemStandby) .
200                   PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,false,none) +
201          (!cRPCstarted) -> IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease .
202                   sIGeoPC(powerOff) | scommandhandler(powerOff) .
203                  PDU_State_Machine(StartingUpCrPC,cRPCstarted,geoPCstarted,false,none) +
204          IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
205                    IndicationCB(off) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
206          IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
207                  IndicationCB(off) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
208                  IPDUTimer(pollPC) . sICR_PC(queryCRPCStatus) .
209                  PDU_State_Machine(WaitingCRPCStatusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
210          (geoPCstarted) -> IPDUTimer(pollGeoPC) . sIGeoPC(queryGeoStopFlag) .
211                  PDU_State_Machine(CheckingGeoStopStatus,cRPCstarted,geoPCstarted,geoPressed,StartingUpAllPCs) +
212          (!geoPCstarted) -> IPDUTimer(pollGeoPC) .
213                  PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state) +
214          (cRPCstarted) -> IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
215                  PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,StartingUpAllPCs) +
216          (!cRPCstarted) -> IPDUTimer(pollCRPC) .
217                  PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
218  )
219  +
220  (s==System_Standby) -> (
221          IPDU(PDUswitchOff) . sICR_PC(powerOff) .
222                  PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
223          IPDU(powerOn) . sIGeoPC(powerOn) | scommandhandler(powerOn) . IndicationCB(startingUp) .
224                   PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)  +
225          IPDU(powerOff) . PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,geoPressed,state) +
226          IPDU(forcedPowerOff) . sICR_PC(powerOff) . IndicationCB(off) .
227                  PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
228           IPDU(emergencyOff) . sICR_PC(powerOff) . IndicationCB(off) .
229                  PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
230          IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
231                  PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,System_Standby)
232          )
233          +
234  (s==WaitingCRPCStatusReply) -> (
235          rICR_PCrVal(IsOperational) . sIGeoPC(queryGeoPCstatus) .
236                  PDU_State_Machine(WaitingGeoPCStatusReply,true,geoPCstarted,geoPressed,state) +
237          rICR_PCrVal(IsNotOperational) . PDU_State_Machine(StartingUpAllPCs,false,geoPCstarted,geoPressed,state)
238  )
239  +
240  (s==CheckingGeoStopStatus) -> (
```

```
241         (state==StartingUpAllPCs) -> rIGeoPCrVal(StopIsPressed) .
242                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,true,state) +
243         (state==System_On) -> rIGeoPCrVal(StopIsPressed) . IndicationCB(geoStop) .
244                 PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,true,state) +
245         (state==StartingUpAllPCs) -> rIGeoPCrVal(StopIsNotPressed) .
246                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,false,state) +
247         (state==System_On) -> rIGeoPCrVal(StopIsNotPressed) .
248                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,false,state)
249 )
250 +
251 (s==CheckingCRPCFlag) -> (
252         (state!=System_Standby) -> rICR_PCrVal(IsOn) . sICR_PC_Broadcast(shutdown) .
253                 sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease . sICR_PC(powerOff) .
254                 sIGeoPC(powerOff) | scommandhandler(powerOff) . IndicationCB(off) .
255                  PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
256         (state==System_Standby) -> rICR_PCrVal(IsOn) . sICR_PC_Broadcast(shutdown) .
257                 sICR_PC(powerOff) . IndicationCB(off) .
258                  PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
259         (state==Geo_Stop) -> rICR_PCrVal(IsOff) .
260                 PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,geoPressed,none) +
261         (state==System_On) -> rICR_PCrVal(IsOff) .
262                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,none) +
263         (state==StartingUpAllPCs) -> rICR_PCrVal(IsOff) .
264                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,none) +
265         (state==System_Standby) -> rICR_PCrVal(IsOff) .
266                 PDU_State_Machine(System_Standby,cRPCstarted,geoPCstarted,geoPressed,none)
267 )
268 +
269 (s==WaitingGeoPCStatusReply) -> (
270         rIGeoPCrVal(IsOperational) . sIPC(1,queryPCstatus) .
271                 PDU_State_Machine(WaitingPC1statusReply,cRPCstarted,true,geoPressed,state) +
272         rIGeoPCrVal(IsNotOperational) .
273                 PDU_State_Machine(StartingUpAllPCs,cRPCstarted,false,geoPressed,state)
274 )
275 +
276 (s==Geo_Stop) -> (
277         IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
278                 PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
279         IPDU(powerOn) . sIGeoPC(onPressed) . IndicationCB(systemOn) .
280                 PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,state) +
281         IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease . sIGeoPC(powerOff) |
282                 scommandhandler(powerOff) . IndicationCB(systemStandby) .
283                 PDU_State_Machine(System_Standby,cRPCstarted,false,false,state) +
284         IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
285                 IndicationCB(off) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
286         IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
287                 IndicationCB(off) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
288         IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
289                 PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,Geo_Stop)
290 )
291 +
292 (s==System_On) -> (
293         IPDU(PDUswitchOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
294                 PDU_State_Machine(PDU_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
295         IPDU(powerOn) . sICR_PC_Broadcast(restart) . sIGeoPC_Broadcast(restart ) . sMsghandler(restart) .
296                  rrelease . IndicationCB(startingUp) . PDU_State_Machine(StartingUpAllPCs,false,false,false,none) +
297          IPDU(powerOff) . sIGeoPC_Broadcast(shutdown) . sMsghandler(shutdown) . rrelease . sIGeoPC(powerOff) |
298                 scommandhandler(powerOff) . IndicationCB(systemStandby) .
299                 PDU_State_Machine(System_Standby,cRPCstarted,false,false,state) +
300         IPDU(forcedPowerOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
301                 IndicationCB(off) . PDU_State_Machine(System_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
302         IPDU(emergencyOff) . sICR_PC(powerOff) . sIGeoPC(powerOff) | scommandhandler(powerOff) .
303                 IndicationCB(off) . PDU_State_Machine(Emergency_Off,cRPCstarted,geoPCstarted,geoPressed,state) +
304         IPDUTimer(pollGeoPC) . sIGeoPC(queryGeoStopFlag) .
305                 PDU_State_Machine(CheckingGeoStopStatus,cRPCstarted,geoPCstarted,geoPressed,System_On) +
306         IPDUTimer(pollCRPC) . sICR_PC(queryCRPCPowerOffFlag) .
307                 PDU_State_Machine(CheckingCRPCFlag,cRPCstarted,geoPCstarted,geoPressed,System_On)
```

```
)
+
(s==WaitingPC1statusReply) -> (
        rIPCrVal(1,IsOperational) . sIPC(2,queryPCstatus) .
                PDU_State_Machine(WaitingPC2statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
        rIPCrVal(1,IsNotOperational) .
                PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
+
(s==WaitingPC2statusReply) -> (
        rIPCrVal(2,IsOperational) . sIPC(3,queryPCstatus) .
                PDU_State_Machine(WaitingPC3statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
        rIPCrVal(2,IsNotOperational) .
                PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
+
(s==WaitingPC3statusReply) -> (
        rIPCrVal(3,IsOperational) . sIPC(4,queryPCstatus) .
                PDU_State_Machine(WaitingPC4statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
        rIPCrVal(3,IsNotOperational) .
                PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
+
(s==WaitingPC4statusReply) -> (
        rIPCrVal(4,IsOperational) . sIPC(5,queryPCstatus) .
                PDU_State_Machine(WaitingPC5statusReply,cRPCstarted,geoPCstarted,geoPressed,state) +
        rIPCrVal(4,IsNotOperational) .
                PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
+
(s==WaitingPC5statusReply) -> (
        (!geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(systemOn) .
                PDU_State_Machine(System_On,cRPCstarted,geoPCstarted,geoPressed,state) +
        (geoPressed) -> rIPCrVal(5,IsOperational) . IndicationCB(geoStop) .
                PDU_State_Machine(Geo_Stop,cRPCstarted,geoPCstarted,geoPressed,state) +
        rIPCrVal(5,IsNotOperational) .
                PDU_State_Machine(StartingUpAllPCs,cRPCstarted,geoPCstarted,geoPressed,state)
)
);

proc Handler = sum c:Command . rcommandhandler(c) | sIPC(1,c) | sIPC(2,c) | sIPC(3,c) | sIPC(4,c) |
                sIPC(5,c) .   Handler + sum m:Bmsg . rMsghandler(m) | sIPC_Broadcast(1,m) .
                sIPC_Broadcast(2,m) . sIPC_Broadcast(3,m) .sIPC_Broadcast(4,m) .
                srelease|sIPC_Broadcast(5,m) . Handler;

proc System=
      hide({  IPDUTimer,ICR_PC,ICR_PC_Broadcast,
              IGeoPC,IGeoPC_Broadcast,
              IPC,IPC_Broadcast,
              ICR_PCrVal,
              IGeoPCrVal,
              IPCrVal ,
              release,Msghandler,commandhandler

           },
        allow({IPDU,IPDUTimer,
               ICR_PC,ICR_PC_Broadcast,
               IGeoPC,IGeoPC_Broadcast,
               IPC   ,IPC_Broadcast   ,
                ICR_PCrVal,
               IGeoPCrVal,
               IPCrVal ,
               IndicationCB,
               IGeoPC|commandhandler|IPC|IPC|IPC|IPC|IPC ,
               Msghandler|IPC_Broadcast,release|IPC_Broadcast,
               Illegal
              },
```

```
375           comm({sICR_PC | rICR_PC -> ICR_PC,
376                 sICR_PC_Broadcast | rICR_PC_Broadcast -> ICR_PC_Broadcast,
377                 sIGeoPC | rIGeoPC->IGeoPC,
378                 sIGeoPC_Broadcast | rIGeoPC_Broadcast -> IGeoPC_Broadcast,
379                 srelease | rrelease -> release,
380                 rIPC | sIPC -> IPC,
381                 rIPC_Broadcast | sIPC_Broadcast -> IPC_Broadcast,
382                 sMsghandler | rMsghandler -> Msghandler,
383                 scommandhandler | rcommandhandler -> commandhandler,
384                 sICR_PCrVal | rICR_PCrVal -> ICR_PCrVal,
385                 sIGeoPCrVal | rIGeoPCrVal -> IGeoPCrVal,
386                 sIPCrVal | rIPCrVal -> IPCrVal
387               },
388                           ControlPC(PC_Off)  ||
389                            GeoPC(PC_Off)      ||
390                            NormalPC(1,PC_Off) ||
391                            NormalPC(2,PC_Off) ||
392                            NormalPC(3,PC_Off) ||
393                            NormalPC(4,PC_Off) ||
394                            NormalPC(5,PC_Off) ||
395                           Handler                   ||
396                           PDU_State_Machine(PDU_Off,false,false,false,none))));
397
398   init System;
399
```