Power

V4.9.3

User Manual

September 2023

Document issue: 4

Page Left Intentionally Blank

Use of the software and of the present software tutorial is submitted to a license agreement to be accepted before the software installation on a computer.

All suggestion or error concerning the software or this software manual can be sent to systema.business@airbus.com

Page Left Intentionally Blank

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

Power Systema is a system level tool to define the power architecture of a satellite. It allows to compute the electrical and thermal behavior of a schematically defined architecture. A schematic editor interface allows to build the power architecture with various elements such as solar array generators, batteries, regulators … Then a power solver allows to compute the in orbit electrical performances of the architecture, taking into account the constraints due to space environment (radiation, thermal, seasonal effects...).

Power is a plug-in application of the Systema environment. This Manual describes the Power application and its integration into the Systema v4 framework. For the general usage of Systema, please refer to the Systema User Manual.

The Power application globally transforms the electrical architecture model into a mathematical model. A complete power analysis of a space system (orbiting around a planet or along an interplanetary trajectory) can then be performed thanks to the power solver included in Thermisol. The results of thermal computation (external fluxes, thermal couplings, temperatures) are accessible by the Thermica application. Thus, in order to take into account the space environment (external fluxes, thermal effects …) in the power analysis, the Power and Thermica applications should be used simultaneously. This manual is only dedicated to the Power application. The definition of the model, trajectory, kinematic and mission and the computation of the external fluxes and temperatures are not detailed in this manual. For more information, please refer to the Systema, Thermica and Thermisol User Manuals.

This document is the user's manual of the Power application. It is split into two main sections:

- First a tutorial which allows to enter straight into the use of Power
- And then the description of the various aspects of this tool.

1 Getting started with Power Systema

1.1 Principles

Power is a plug-in application of the Systema environment dedicated to the electrical power computation. A schematic editor interface ("Schematic" tab in the graphical user interface) allows to build the power architecture with various elements such as solar array, batteries and regulators, taking into account electrical and thermal aspects.

Some input parameters (thermal node of a solar array, fluxes that impact the solar array …) can be directly provided by the Thermica application.

Once the power system is defined, the Power solver allows to compute the electrical and thermal behavior of the system.

To perform a complete power analysis, the user needs to define the following input data:

Under Systema-Thermica

- The geometric model (modeler and meshing)
- The orbit and pointing parameters (trajectory, kinematic and mission)
- Specific parameters for the thermal computation such as capacitance, conductive couplings, internal dissipations …

Under Power

- The electrical architecture (schematic)
- The computation parameters

This chapter describes the procedure to follow and gives the guidelines for a classic use of the application. It allows the user to get directly familiar with the tool.

The way to define the geometric model, the orbit, the pointing and the thermal parameters is not covered in this chapter. For more information, please refer to the Systema and Thermica user manuals. Only the electrical architecture definition and the use of the power solver are explained below through a basic example.

1.2 Electrical architecture building

1.2.1 Open a new schematic file

Launch the Systema-Power software and go in the schematic tab. You have to use the schematic editor to build the electrical architecture you want to simulate. Create a new file in the schematic editor (Click on

File/New or on the button).

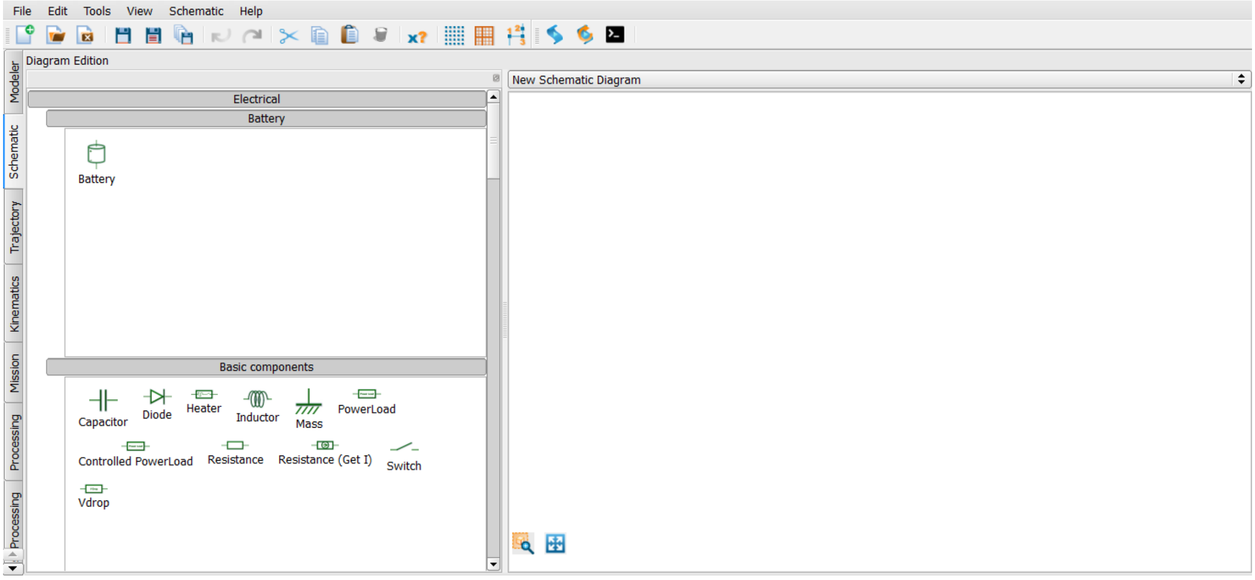


Figure 1.2-1 New schematic file

All the available electrical elements (also named components) appear on the left and the diagram window is empty. You are now ready to work.

1.2.2 Architecture building

The architecture to simulate is for example the following:

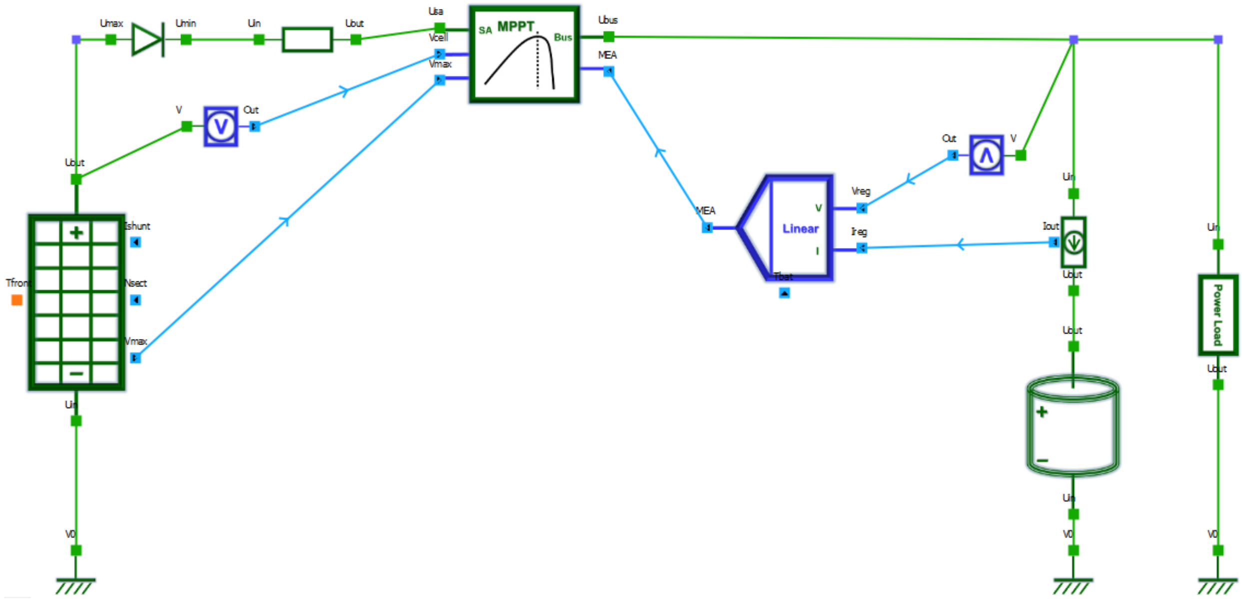


Figure 1.2-2 Unregulated bus architecture

To build your architecture, click on the diode on the left and drag and drop it in the diagram window. Then drag and drop the resistance in the diagram window. To link the two elements, you have to click on one connector, hold down the left mouse button while moving the mouse toward the other connector to be joined.

You can double click on the diode or the resistance in the diagram window to edit it and change the parameters value of the element.

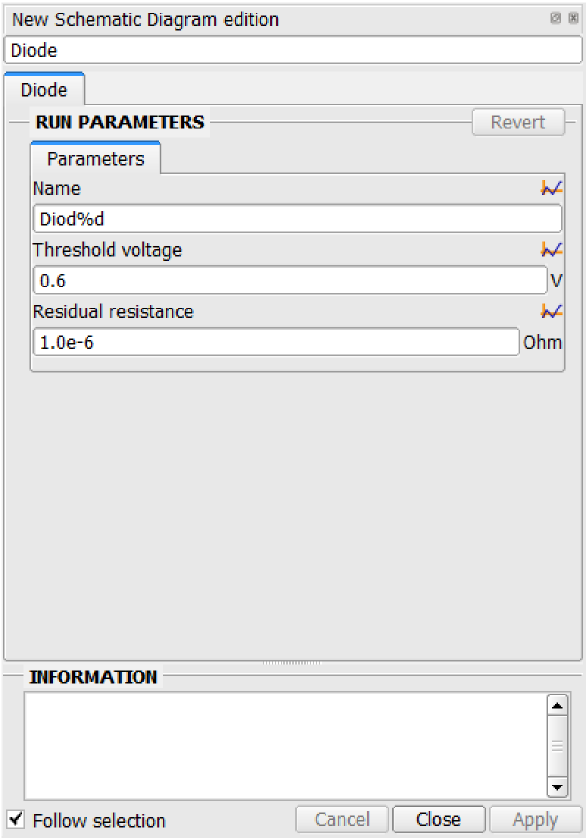


Figure 1.2-3 Edition area

Note that you can change the name of your element, but all elements of the architecture should have a unique name. The uniqueness of the name is ensured if you use the default name.

If you want to rotate an element, click on it and use the blue circle just above.

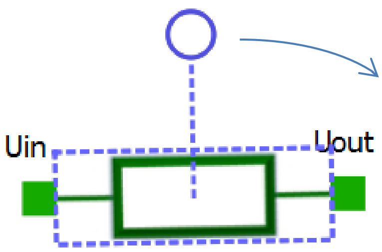


Figure 1.2-4 How to rotate an element

Use the same process to add the other elements and complete the architecture. Please refer to the Systema user manual for more details on how to create a schematic diagram.

The following array lists the parameter values to change in the architecture for our example:

Electrical element	parameter	value
Solar Array	Name	SA
	Number of cells in series	20
	Number of cells in parallel	100
	Area correction factor	1
	Panel area	25m2
	Total number of SA sections	1
	Solar Flux	1367 W/m2
	Threshold for cell's activation	0.001
	Loss factor	1.0
	Cell size (in cell characteristics tab)	30.18cm2
	Other cell characteristics	Use the default values
Diode	Name	Diode
	Threshold voltage	0.6V
	Residual resistance	1.0e-6 Ohm
Resistance	Name	R_SA
	Resistance	0.01 Ohm
MPPT	Name	MPPT
	Vcell initialisation	0.0V
	MPPT efficiency	1.0
	Maximum output power	10000 W
	Minimum input voltage of MPPT	0.0 V
	Maximum input voltage of MPPT	100 V
	Internal regulation constant	100
	Coefficient applied to the integral	1.0
Resistance (Get I)	Name	R_Bat

	Resistance	0.015 Ohm
Linear Regulation	Name	LinReg
	Current limitation	80 A
	Absolute voltage limitation	48 V
	Voltage limitation at 0°C	48 V
	Variation coefficient of limitation voltage according to temperature	0.0 V/K
	Epsilon for the taper voltage threshold	0.05 V
	deltaV for the taper voltage computation	0.5 V
	Regulation constant	100
Battery	Name	Bat
	Number of cells in series	10
	Number of cells in parallel	200
	Cell nominal capacity	3 Ah
	Total resistance of the battery internal wiring	0.001 Ohm
	Initial SOC value	50.0%
	Internal voltage polynomial coefficients	Use the default values
	Internal resistor polynomial coefficient	Use the default values
Power Load	Name	PLoad
	Load power consumption value	3000 W
	Power offset	0.0 W

You can define the thermal node of the Solar Array by double clicking on the “Tfront” thermal connector. The edition window allows you to specify the status of the node (in tab Parameters) and to define its node number (in tab Numbering).

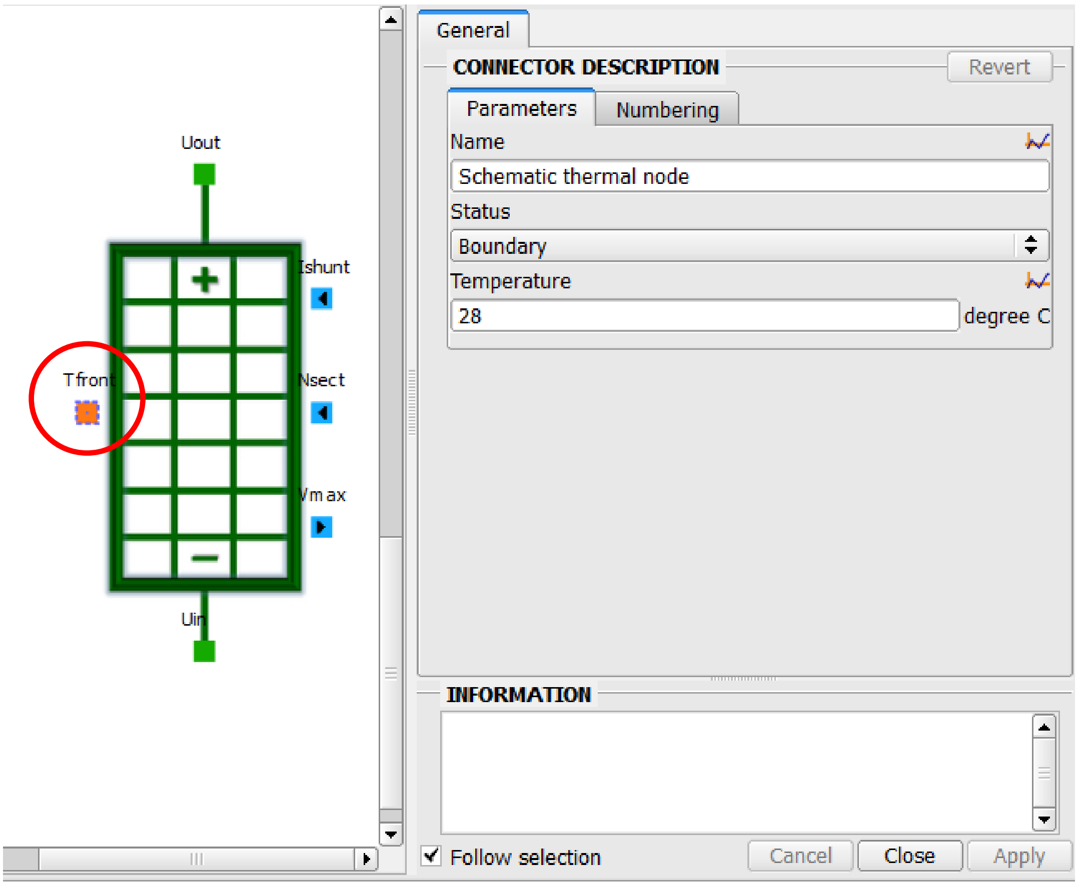


Figure 1.2-5 Thermal node of the Solar Array

Note that you can link the electrical architecture to the geometric model (if you perform a complete power analysis including the Thermica application) by using the same node number in the geometric model and in the architecture (the status of Tfront should then be “virtual”). In that case, the temperature of the solar panel will be computed by Thermica and will be influenced by the space environment.

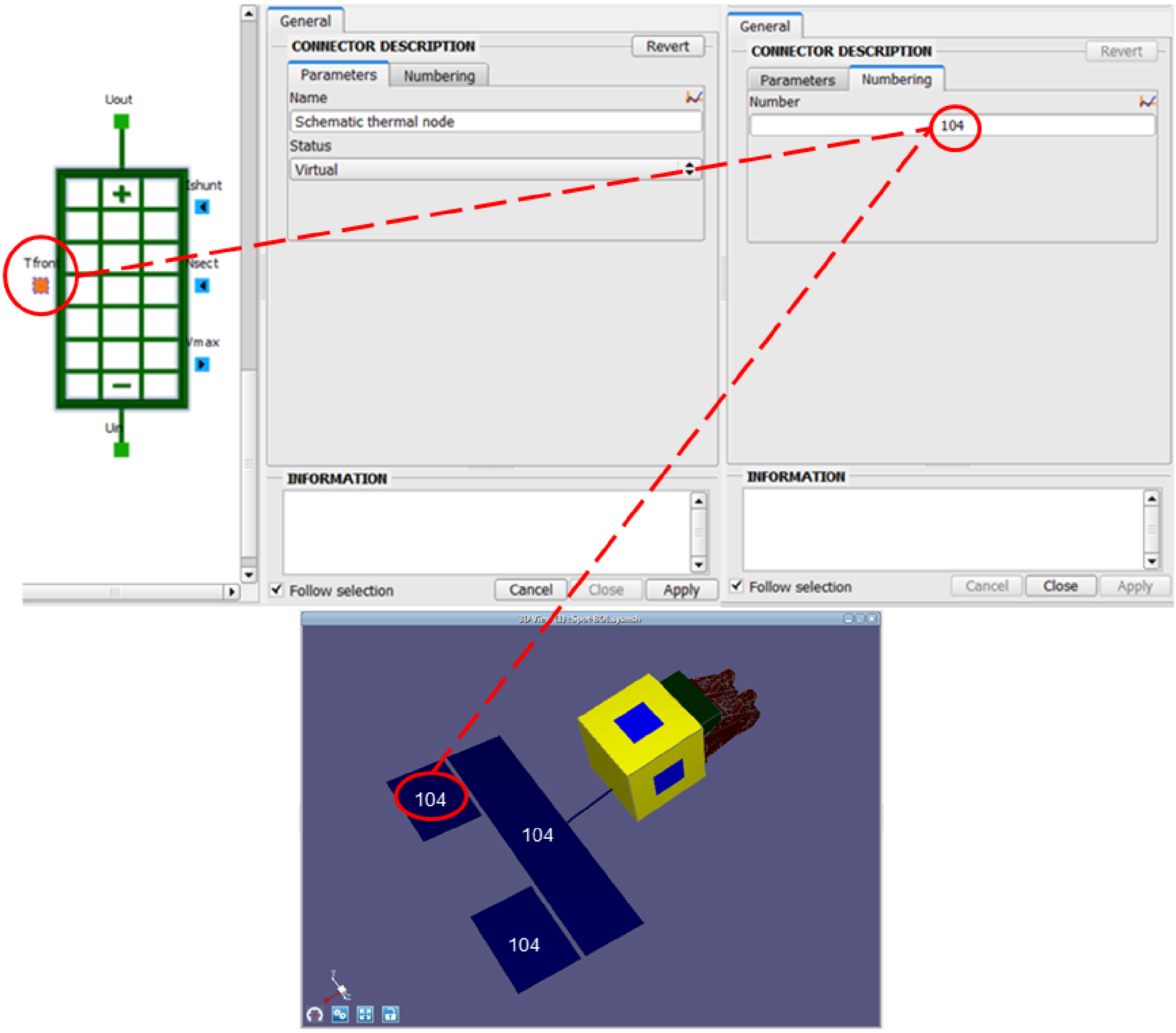


Figure 1.2-6 Define an existing geometrical node as thermal node for the solar array

If you use an existing geometrical node as thermal node for the solar array, you can leave the default formula (QS:Tfront + QA:Tfront) / A:Tfront / ALP:Tfront in the “Solar Flux” parameter of the Solar Array to use directly the absorbed solar fluxes computed by Thermica.

In our example, we have no associated geometric model and we have to define a new thermal node. Choose a boundary node with a fixed temperature of 28°C.

Save your schematic using File/Save option in the menu bar or the button.

1.3 Process creation and computation

Go in the processing tab to create your process.

Create a new processing using File/New option in the menu bar or the button.

Double click on the Power4 module in the module list (on the left) to add the box to the process diagram. You should obtain the following diagram.

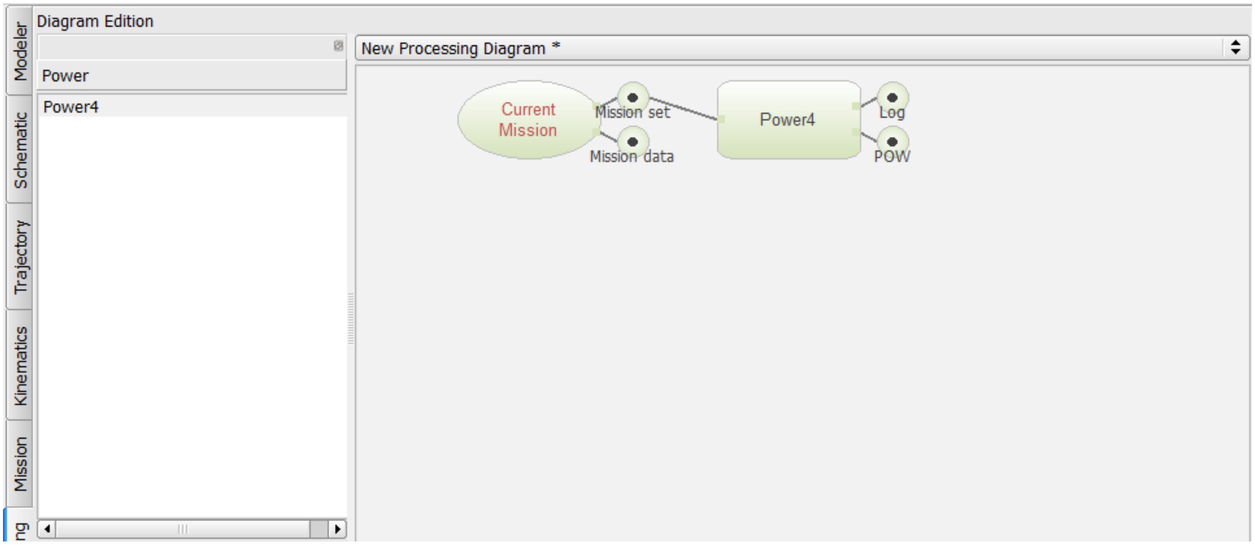


Figure 1.3-1 First part of the process diagram

Double click on the current mission box to edit it. Select “schematic” as the type of exported structure and select the schematic file you just saved in the previous section.

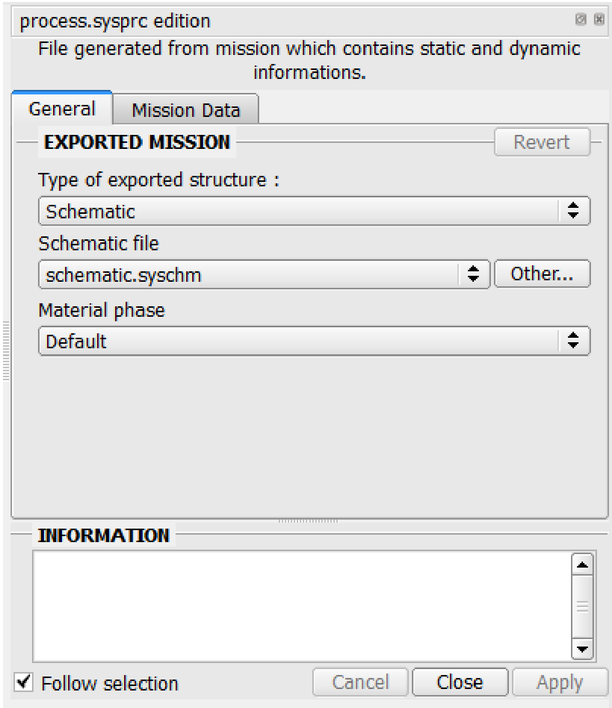


Figure 1.3-2 Edition of the current mission module

The power module transforms the electrical architecture model (.syschm file) into a mathematical model (.pow file). To compute the electrical and thermal behavior of our system, we have to use the solver module. We also need the Skeleton module to create an input file for the solver (.dck file) from the mathematical model (.pow file) and some simulation parameters (specified by editing the Skeleton module). So to complete the process diagram, you have to:

- click on Thermisol in the module list (on the left) to expand the Thermisol module list
- add the skeleton and solver modules to the diagram
- link the POW output of the Power module to the POW input of the Skeleton module
- deactivate all the other inputs of the Skeleton module
- link the DCK output of the Skeleton module to the DCK input of the Solver module

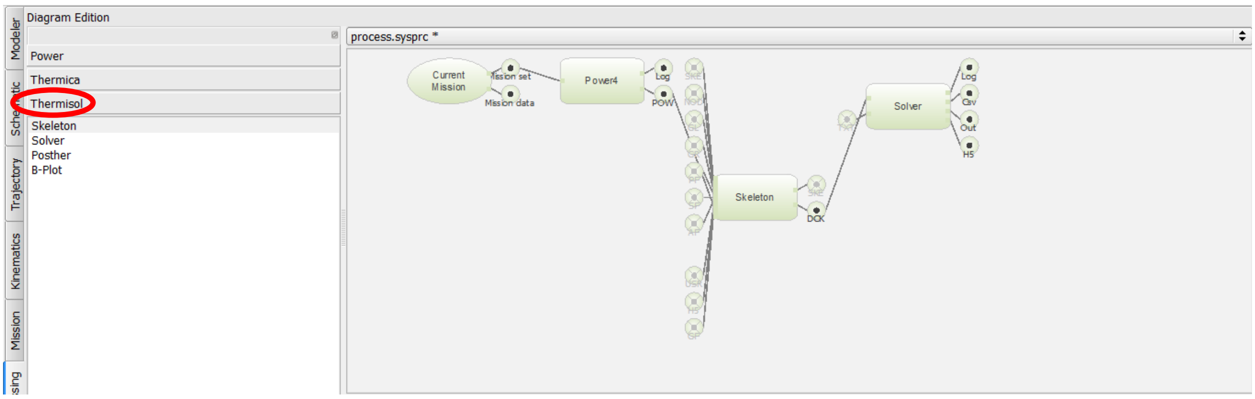


Figure 1.3-3 Complete process diagram

Double click on the Skeleton module to edit it:

- in the “Steady-State” tab, deactivate the computation of the steady state module
- in the “Transient” tab, choose the Thermo-Electrical Transient (TRANS_UT) resolution routine
- in the “Time options” tab, select manual selection for the time specification and leave the default values for the simulation parameters

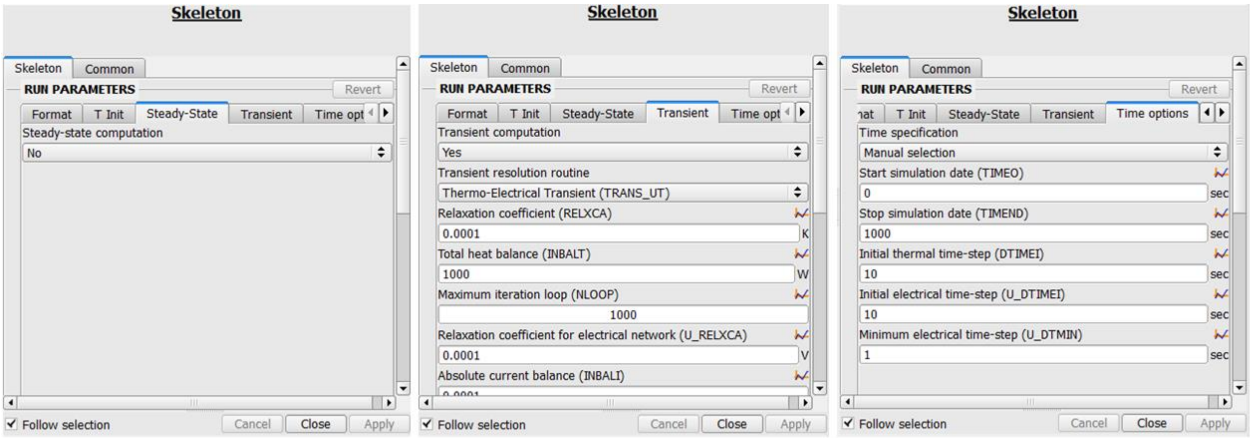


Figure 1.3-4 Edition of the Skeleton module

Save the process using File/Save option in the menu bar or the button.

Click on the button to run the simulation. During computation, the different menus and windows of screen are not accessible. Nevertheless, it is still possible to monitor the computation progress on the process run window that appears. The run is finished when the two progress bars indicate 100%.

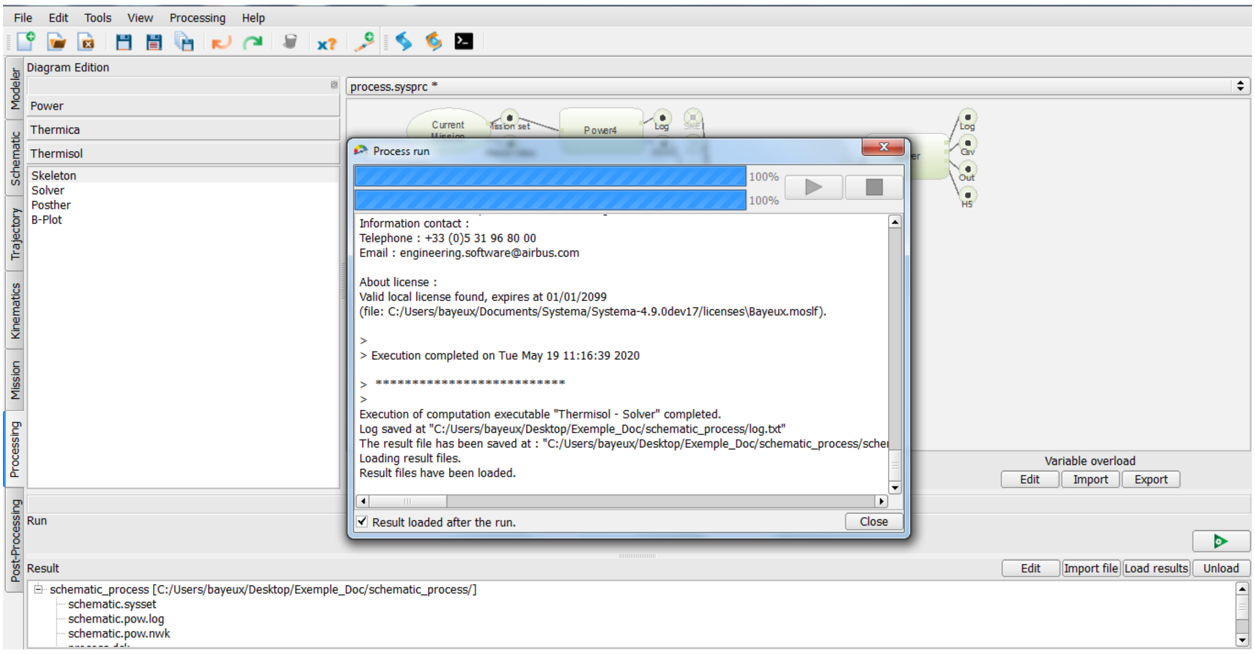


Figure 1.3-5 Process run window

Click on Close button when the run is finished.

It is strongly advised not to stop the calculation before it is finished. If you really need to stop the calculation, use the task manager rather than the Stop or Close button.

Several files have been generated:

- .sysset contains a mission synthesis for the Power and Thermica applications.
- .pow.log is the log file of the power module. It contains information about the process but also a description of the system as it is analysed by the module.
- .pow.nwk is a txt file containing the mathematical description of the electrical architecture.
- .dck is the input file for the solver. It contains the mathematical model of the electrical architecture and the run parameters.
- .temp.log is the log file of the solver module. It relates the main events occurring during translation task.
- .temp.csv is a spreadsheet run report. It contains information to check the convergence of the simulation.
- .temp.out contains the standard output text generated during the calculation.
- .temp.h5 is a result file in the HDF5 format.

1.4 Viewing results

It is possible to visualize the simulation results thanks to the Graph View accessible from the Modeler tab:

- In the Modeler tab, use the View/Add/Graph View option of the menu bar or click on the button to open a viewport of graph.
- Open the configuration window (in the right click menu, select configuration window or use the button)
- Go in the data tab of the configuration window
- In the “Ordinate” part, choose the .temp.h5 file generated by the solver
- Then you can select the results you want to display in the graph viewport by choosing an entity.

Select for example power/Electric Potentials to visualize the potential of all the electrical nodes.

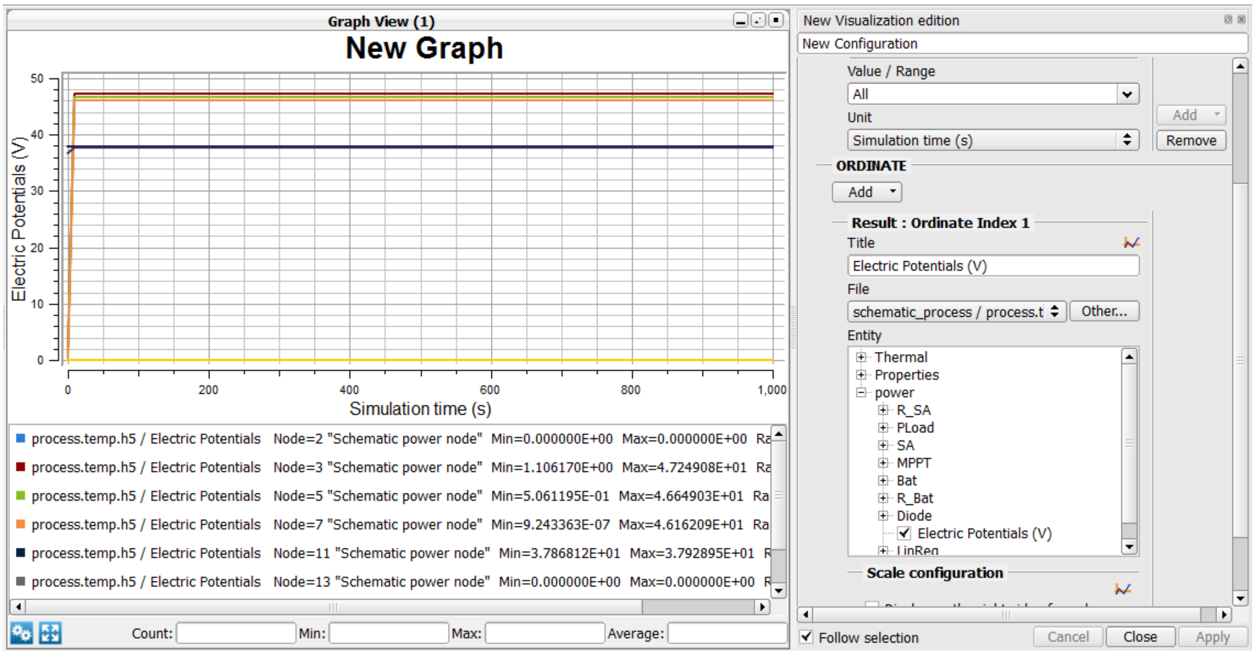


Figure 1.4-1 visualization of electric potentials

Note that you can display the node numbers on the schematic diagram. To do that, go back in the schematic tab and use the Schematic/Display Number menu option or click on the button.

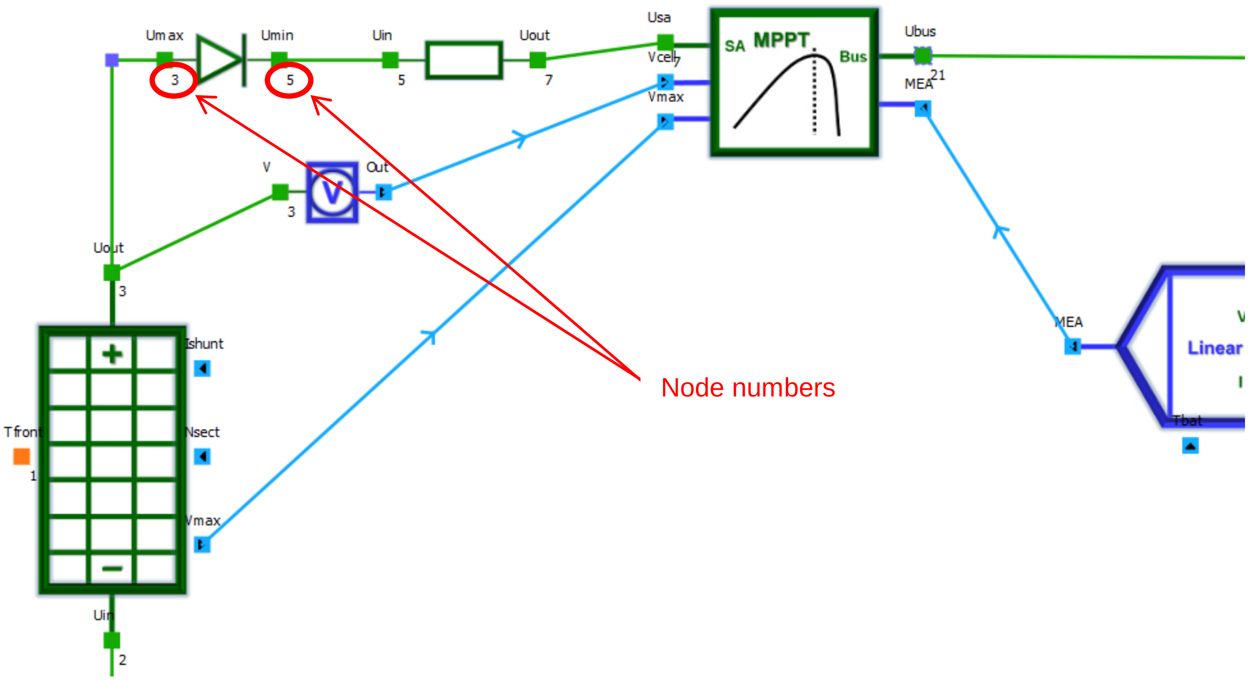


Figure 1.4-2 Display node numbers on the schematic diagram

To display the results for only specific node(s), go in the General tab and modify the node value/range. The picture below shows the electric potentials of nodes corresponding to the input and output of the diode (3 and 5 on this example).

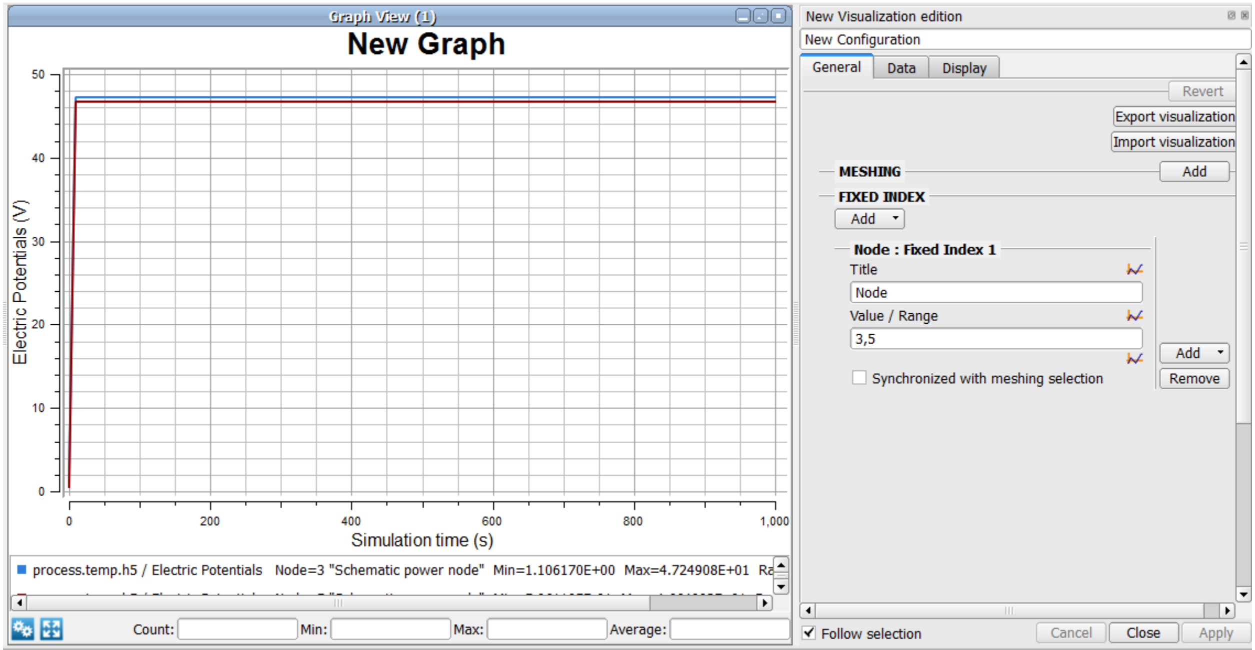


Figure 1.4-3 Visualization of electric potentials for specific nodes

If you now want to display the input and output current of the solar array, write "All" in the Node value/Range of the general tab. Then, go back to the data tab, select power/SA/Current Values for the entity and apply.

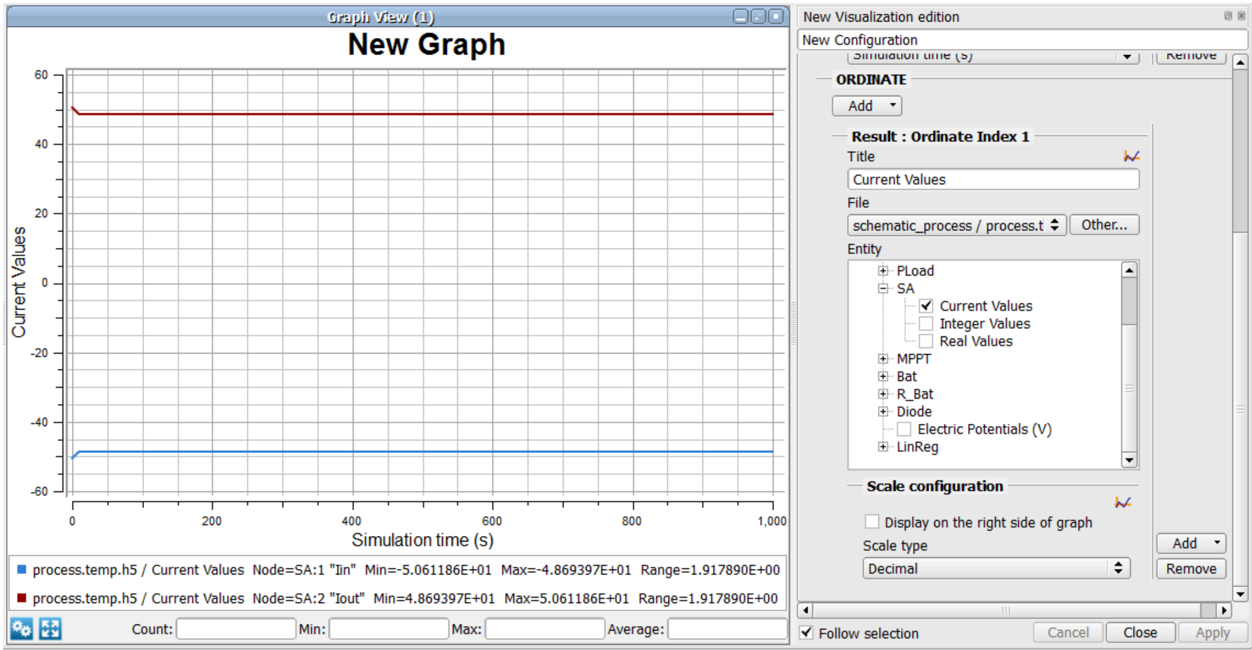


Figure 1.4-4 Visualization of the input and output current of the Solar Array

Finally, you can also visualize all the input and output parameters of the Solar Array by selecting power/SA/Integer Values (for all the integer parameters) or power/SA/Real Values (for all the real parameters).

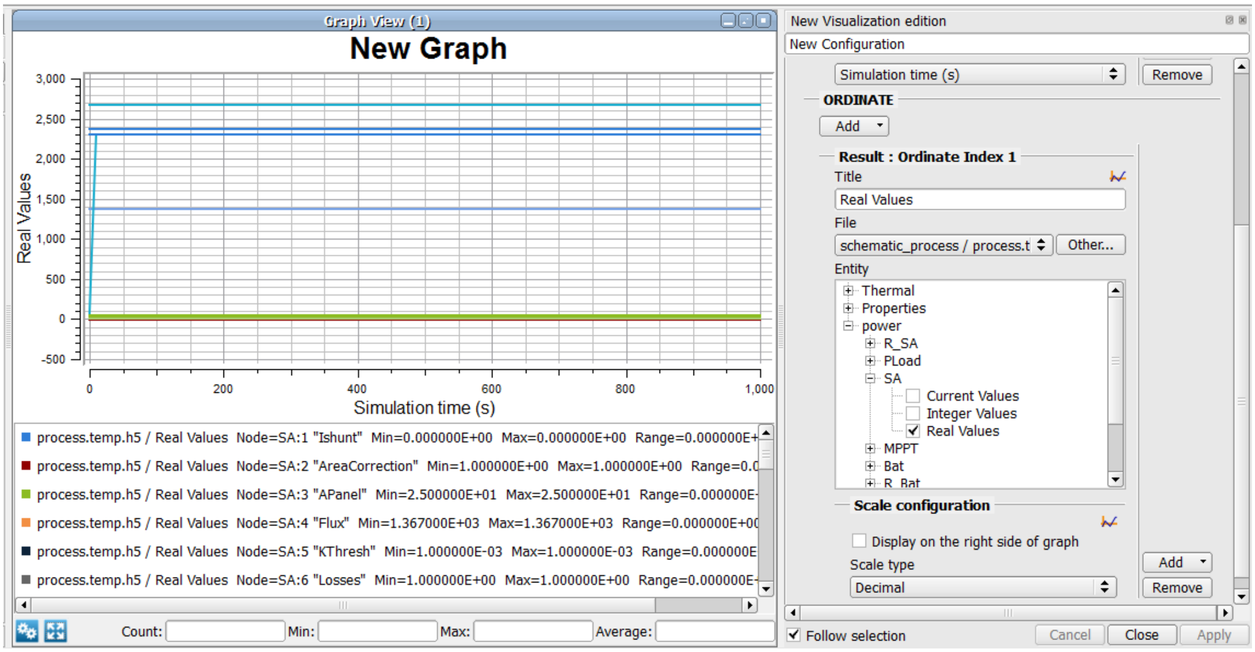


Figure 1.4-5 Visualization of the real input and output parameters of the Solar Array

It is also possible to display only the desired parameters. To do that, specify the index of the desired parameter(s) in the Node value/range field of the general tab. Please refer to section 7.3 “List of parameter indexes” to see the list of the parameter indexes for each component. You can also find the parameters indexes in the legend of the graph when you plot all the parameters of a component as previously.

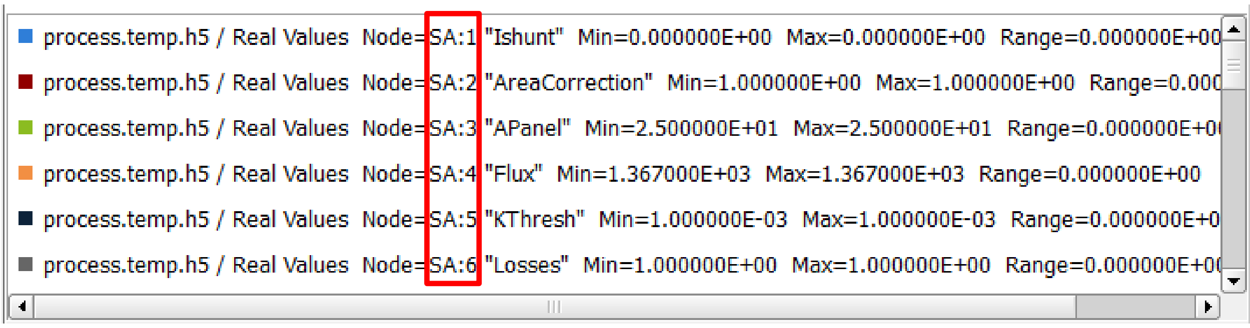


Figure 1.4-6 How to find the parameter indexes in the legend

The picture below shows how to display the maximum power “Pmax” and the delivered power “Pout” of the Solar Array (here, the respective indexes are SA:22 and SA:25).

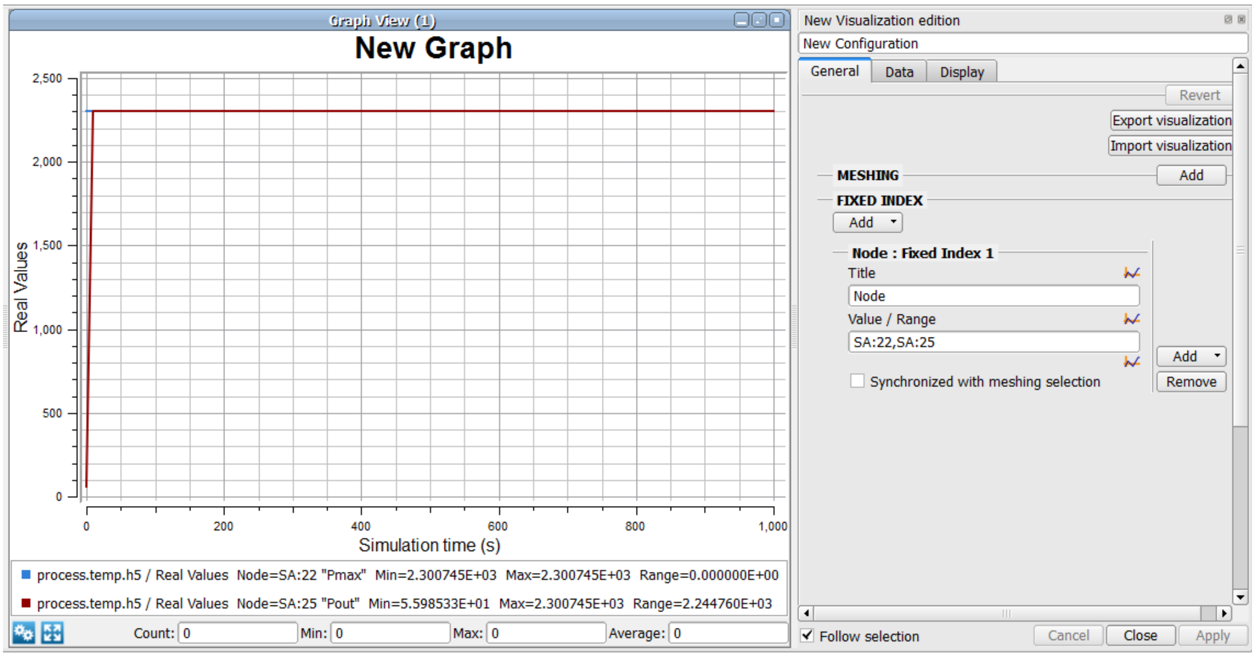


Figure 1.4-7 Visualization of some specific real parameters of the Solar Array

The same process is applicable for plotting the currents or parameters of the other components.

For the general usage of the Graph View, please refer to the Systema User Manual.

2 Schematic editor

2.1 General use

For the general use of the schematic editor, please refer to the Systema User Manual (chapter “Schematic management”).

2.2 Component library management

The following picture shows the schematic editor interface.

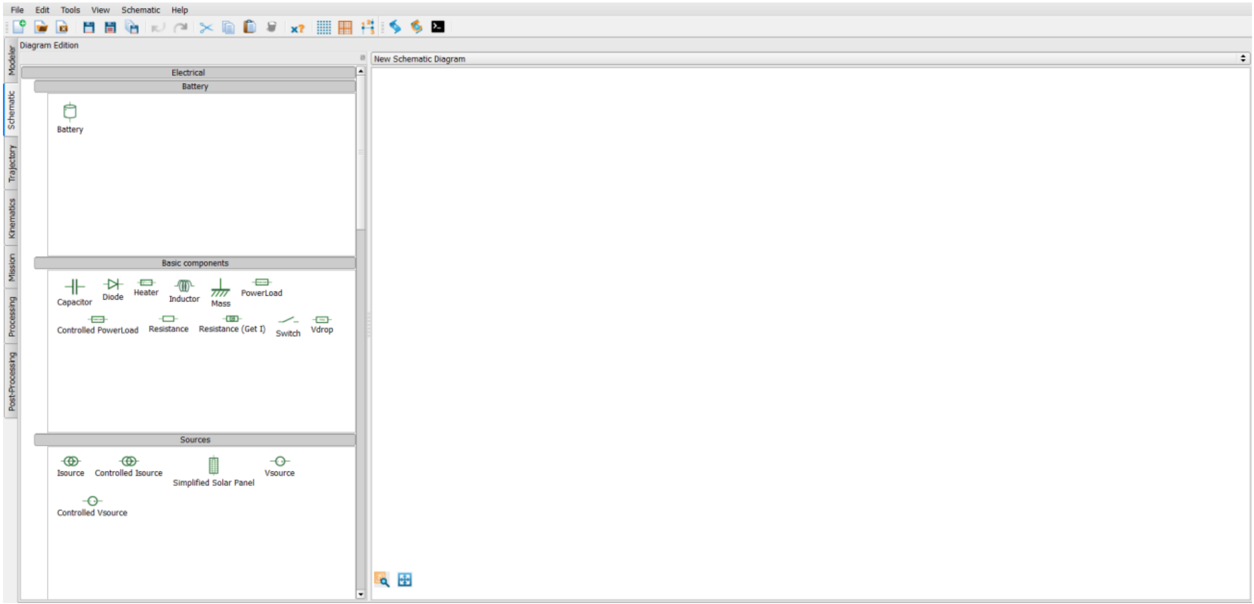


Figure 2.2-1 Schematic editor interface

The left part of the window is dedicated to the component browser. This browser contains all the electrical elements available to build an electrical architecture.

By default, the components that are displayed in the browser are loaded from the folder “components” in the installation directory (SYSTEMA_INSTALL_DIR/applications/Power-X.X.X/components)

It is possible to load another component library by using the File/Settings option of the menu bar:

- Click File/Settings in the menu bar
- Click on “Schematic” in the settings list
- Unselect “Use default location”
- Choose your own components path
- Apply

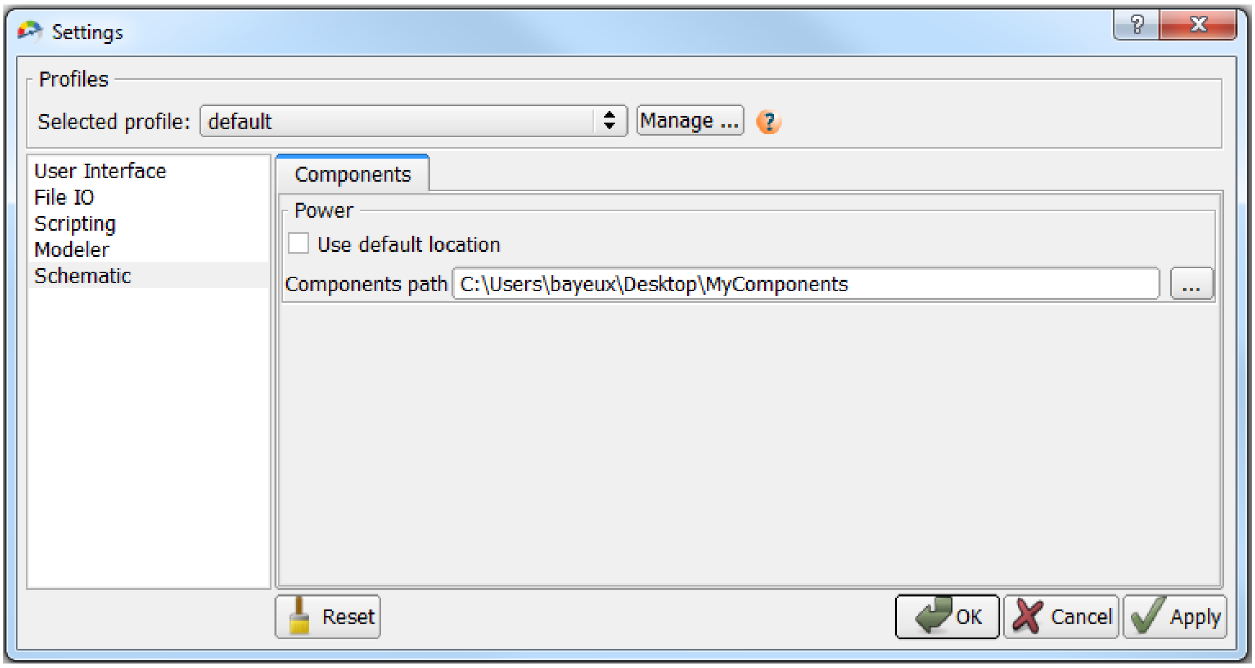


Figure 2.2-2 Change the components library loaded in the GUI by using the settings

Warning: it is then necessary to restart Systema in order to load the components corresponding to the modified path.

It is also possible to load another component library by setting an environment variable:
SYSTEMA_COMPONENTS_PATH = path allows to define the path where the components are stored (this also requires a restart of Systema)

Warning: if the environment variable SYSTEMA_COMPONENTS_PATH is set, the component path specified in the settings is no longer taken into account. Please unset the environment variable to use the components path defined in the settings.

To load a component in the schematic editor, Systema needs a .jpeg file containing the drawing of the component and a .sysapp file containing all the description of the component (name, type, parameters …)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SYSML CONTENTS="SCHEMBOX_DEFINITION" SYSML_VERSION="1.1">
    <SCHEMBOX_DEF NAME="Resistance" EXE="Resistance" TYPE="Electrical" SUBTYPE="Basic components" SYMBOL="./Resistance.jpg">
        <DESCRIPTION></DESCRIPTION>
        <CONNECTOR_DEF TYPE="ELEC" SIDE="LEFT" ROLE_TITLE="Uin" ROLE_DESC="Uin"/>
        <CONNECTOR_DEF TYPE="ELEC" SIDE="RIGHT" ROLE_TITLE="Uout" ROLE_DESC="Uout"/>
        <RUN_PARAMETERS_DEF>
            <PARAM_CATEGORY_DEF NAME="Parameters">

                <PARAMETER_DEF ID="idName" NAME="Name" TYPE="STRING" UNIT="">
                    <ELEM ID="Default_Value" TYPE="STRING" STR="Res%d"/>
                </PARAMETER_DEF>

                <PARAMETER_DEF ID="R" NAME="Resistance" TYPE="VALUE" UNIT="Ohm">
                    <ELEM ID="Default_Value" TYPE="VALUE" VALUE="1.0"/>
                </PARAMETER_DEF>
            </PARAM_CATEGORY_DEF>
        </RUN_PARAMETERS_DEF>
    </SCHEMBOX_DEF>
</SYSML>
```

Figure 2.2-3 Example of a .sysapp file

3 Power module

The power module transforms the electrical architecture model (.syschm file) into a mathematical model (.pow file). It is launched from the processing tab of Systema.

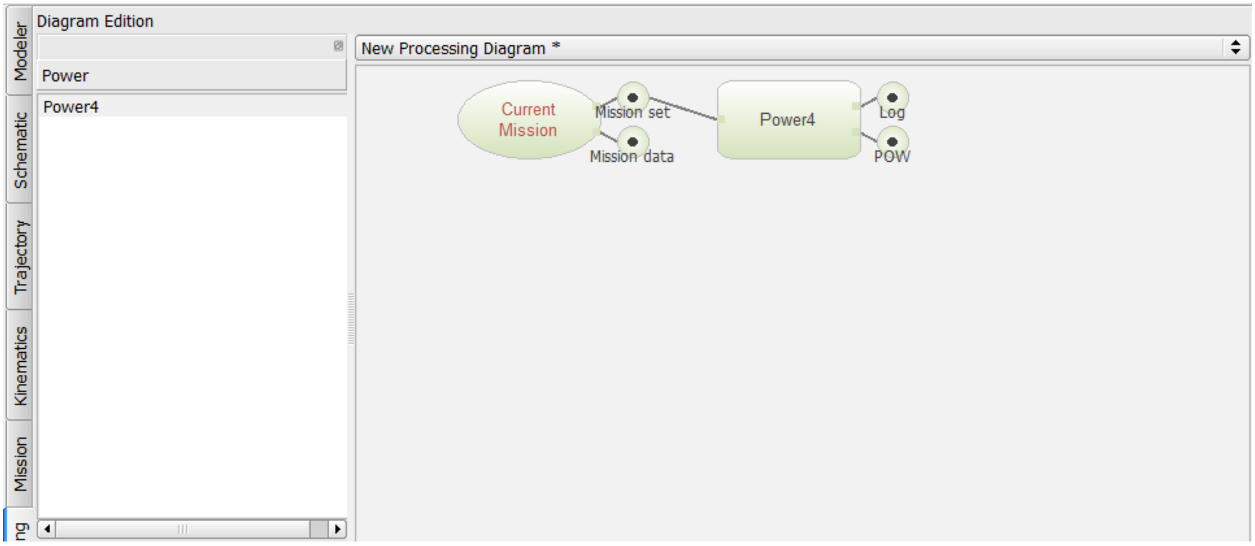


Figure 2.2-1 The power module

3.1 Input and output files

3.1.1 Inputs

The power module has only one input:

- .sysset file contains the mission synthesis for the Thermica and Power application. It includes in particular the synthesis of the schematic file.

3.1.2 Outputs

The power module generates two output files:

- .pow.log is the log file of the power module. It contains information about the process but also a description of the system as it is analysed by the module.
- .pow.nwk is a txt file containing the mathematical description of the electrical architecture.

3.1.3 Parameters

Since Power-4.9.2, the power module has the following parameters:

- "Parametric schematic". This option allows the advanced variables defined in a schematic diagram to be exported to the pow.nwk file. Only variables used in components or connectors are exported except for :
    - The names of the components or connectors.
    - The numbers of the connectors
    - If the variable's formula contains the keyword Tfront

  If a parameter of a schematic element is defined by a variable that depends on another (for example var2 = var1 * 2.), the option must be activated. If the variables used in the formula (var1 in the example) are not used in a schematic element, they will not be exported to the pow.nwk file. In this case, they will have to be defined in another file (for example, the USR input file of the Skeleton).
  The variables are exported in the $LOCALS paragraph. They cannot be modified by the solver during a calculation.
  By default, this option is turned off.

3.2 Setting of the Current Mission

To run the power module, you need to choose a "current mission". To do that, double click on the Current Mission box to edit it.

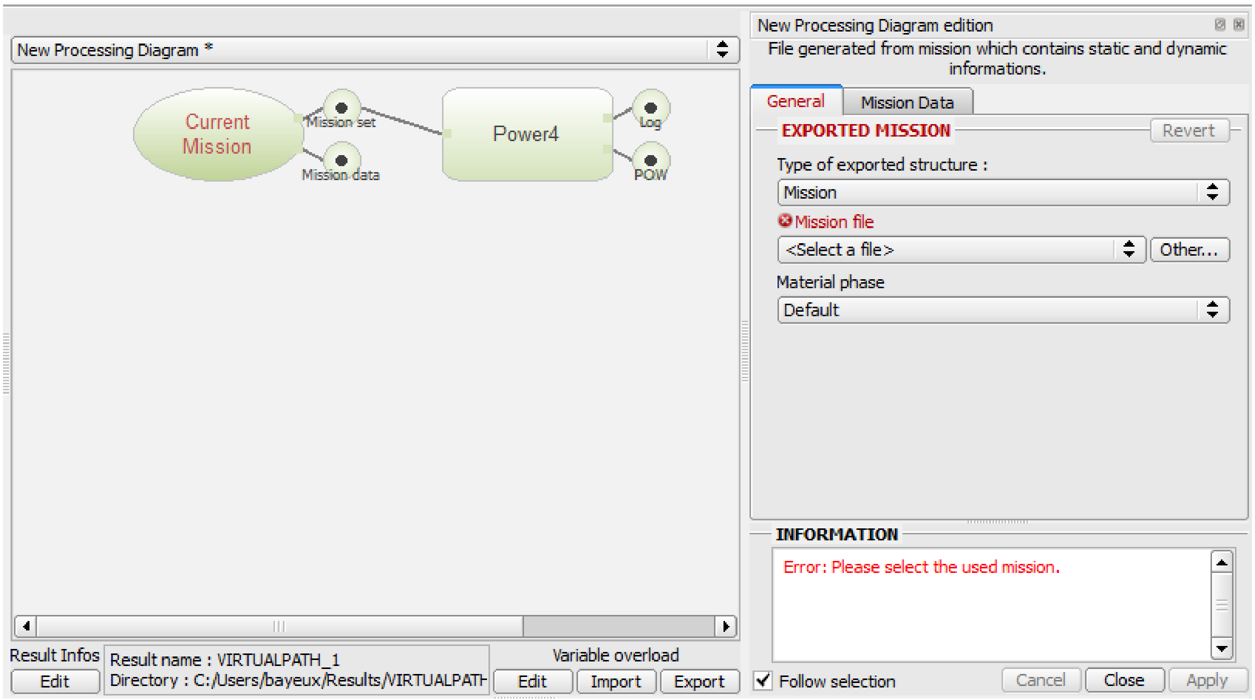


Figure 3.2-1 Setting of the Current Mission

- If your electrical architecture is associated to a mission (model, trajectory, kinematic, mission), select "mission" as type of exported structure, select your mission file (.sysmis in which a schematic file is included) and apply.
- If your electrical architecture is not associated to a mission, select "Schematic" as type of exported

structure, select your schematic file and apply.

3.3 Structure of the output file

The pow.nwk file is the output txt file containing the mathematical description of the electrical architecture. The problem is described by a set of nodes, being either thermal nodes or electrical nodes. Electrical nodes are set with electrical properties (voltage) and links with other nodes (components).

The picture below give an example of output file generates by the Power module.

```
#***********************************************#
#                                               #
#         Network file created by Power4        #
#                                               #
#***********************************************#

$INFOS

#  Power4 Schematic
#  ----------------
#
#  File : schematic.pow.nwk
#  Version : V.4.9.0 released on September 2020
#  Executed on : Tue Sep 08 10:58:36 2020
#  Input files :
#    - Sysset : schematic.sysset


$GLOBAL
$ENDGLOBAL


$NODES

   # Export of Schematic Thermal Nodes

   # Export of Schematic Electrical Nodes
   V         0 = 'Schematic power node'
   V         1 = 'Schematic power node'
   U         2 = 'Schematic power node'


$CONDUCTORS
   # Export of Schematic Components
   Vs = Vsource(Uin=0,Uout=1) = [ Vs=2.0 ]
   Res = Resistance(Uout=0,Uin=2) = [ R=1.0 ]
   Diode = Diode(Umin=2,Umax=1) = [ Vd=0.6,
                                    Ron=1.0e-6 ]
```

Figure 3.3-1 Example of output file pow.nwk

The corresponding schematic is the following:

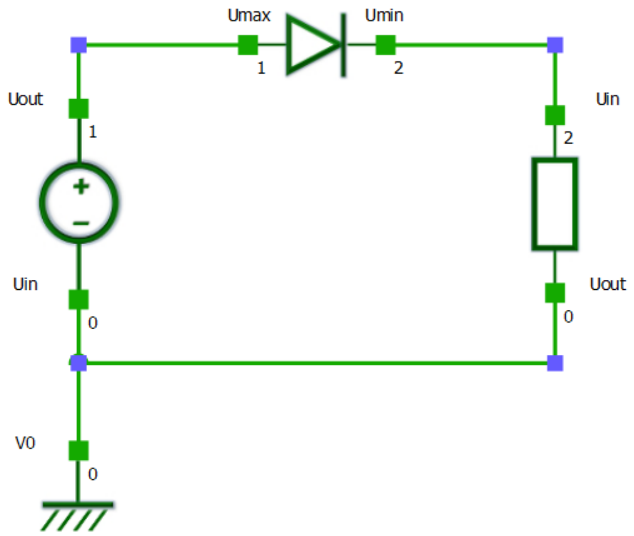


Figure 3.3-2 Schematic corresponding to the output file example

The output file of the power module is generally composed of 2 main paragraphs.

The $NODES paragraph of the file contains the declaration of electrical nodes defined by a type (U= free potential or V=fixed potential), a node number, a name, and eventually initial properties.

In the example, 3 electrical nodes are declared. Node 0 and 1 have a fixed potential and the potential of node 2 will be calculated during the simulation.

The $CONDUCTORS paragraph contains the declaration of electrical components between electrical and thermal nodes and the value of their parameters.

In the example, a voltage source of 2V is defined between node 0 and 1. A diode is declared between node 1 and 2 with a threshold voltage of 0.6V. There is also a resistance of 1 Ohm between node 0 and 2.

4 Power solver

The solver used to calculate the solution is Thermisol-Power. It is an extension of Thermisol for electrical or thermo-electrical simulations. In this User Manual, only the extra features added to Thermisol for the purpose of electrical model designs and simulations are presented. Reading the Thermisol User Manual is a prerequisite for understanding this chapter. In particular, Thermisol-Power is based on the Thermisol language (ESATAN-like) and it is supposed that this language is already known.

4.1 General principle

The Thermisol-Power solver is based on a nodal method, also called “lumped parameter method”. The problem is described by a set of nodes, being either thermal nodes (see Thermisol User Manual) or electrical nodes. Electrical nodes are set with:

- Electrical properties (voltage)
- Links with other nodes (components)

The network properties are as follows:

Network properties	***Symbol***	***Type***	***Entity type***
Node status *‘U’ for a free electrical potential (voltage)* *‘V’ for a fixed (boundary) potential*	NS	Character	Nodal
Electrical potential (voltage)	V	Real	Nodal
Battery	Battery	Component	Coupling
Capacitor	Capacitor	Component	Coupling
Comparator	Comparator	Component	Coupling
Diode	Diode	Component	Coupling
Inductor (self)	Inductor	Component	Coupling
Integrator	Integrator	Component	Coupling
Intensity source	Isource	Component	Coupling
Linear Regulation	LinearRegulation	Component	Coupling
Power Load	PowerLoad	Component	Coupling
Resistance	Resistance	Component	Coupling

Shunt Regulator	ShuntRegulator	Component	Coupling
Solar Array	SolarArray	Component	Coupling
Switch	Switch	Component	Coupling
Voltage drop	Vdrop	Component	Coupling
Voltage source	Vsource	Component	Coupling

Each datum can be time-dependent or can depend on any criterion defined by the user. The declaration and usage of electrical nodes and components are later explained in the section 4.3, “Structure and content of the input file”. As for thermal models (in which temperatures are computed from thermal flux budgets at thermal potential), the electrical network is solved using the equilibrium of electrical flux (currents) at each electrical potential:

Thermal equations For each thermal node *i*	**Electrical equation** For each electrical node *i*
Steady-state: $0 = Q_{ji} + Q_i$ The sum of all thermal powers exchanged with other thermal nodes (through thermal couplings) plus thermal powers directly applied on node *i* is equal to zero (equilibrate thermal budget).	Steady-state: $0 = I_{ji}$ The sum of all electrical currents exchanged with other electrical nodes (through electrical components) is equal to zero.
Transient: $C_i \frac{dT_i}{dt} = Q_{ji} + Q_i$ The thermal budget is equal to the variation of the temperature according to its thermal capacity.	For electrical simulations, the transient equation on the electrical nodes remains the same as the steady-state equation. Transient effects are to be defined at components level so to compute the value of the current at each component’s pin according to time dependencies.

The input data of the Thermisol-Power solver are described in text file(s), with a Fortran-like syntax to define the nodal network and a Fortran-like language to program any arbitrary behavior (time-dependent phenomena, temperature-dependent data, etc). The input data are translated into a Fortran code, compiled and linked with a computation library; this produces an executable program which generates the solution.

For electrical simulations, the input data also include the usage of components. The code of each component (description and equations) is contained in a text file (.powcmp file). All the powcmp files are also compiled to be integrated into the executable program. The powcmp files of the standard components are provided in the installation directory with the Power application (SYSTEMA_INSTALL_DIR/applications/Power-X.X.X/powcmp). By default, Thermisol-Power will parse the powcmp folder of the installation directory to find the powcmp files.

It is possible to choose another powcmp folder by using the Systema settings:

- Click on File/Settings in the menu bar
- Click on “Schematic” in the settings list
- Select the “Powcmp” tab
- Unselect “Use default location”
- Choose your own powcmp path
- Apply

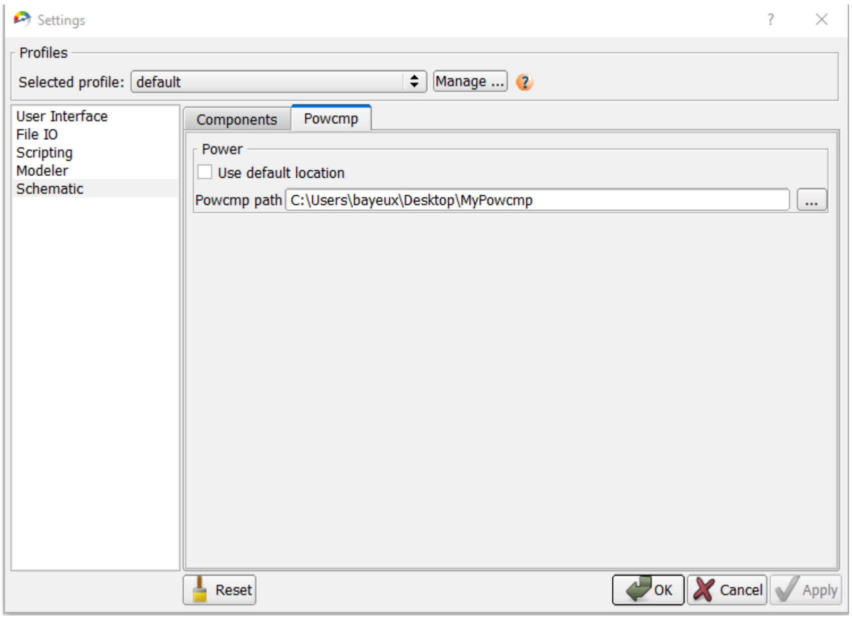


Figure 4.1-1 Change the powcmp folder by using the settings

It is also possible to choose another powcmp folder by using the environment variable POWCMP_DIR:

POWCMP_DIR = path allows to define the path where the powcmp files are stored.

Warning: if the environment variable POWCMP_DIR is set, the powcmp path specified in the settings is no longer taken into account. Please unset the environment variable to use the powcmp path defined in the settings.

After parsing the powcmp folder (using the default or modified path), the solver will also parse the working directory (directory containing the .dck file) to compile the powcmp files found.

In any case, the working directory has priority over the powcmp folder: if a file with the same name exists in both folders, the file selected will be that of the working directory.

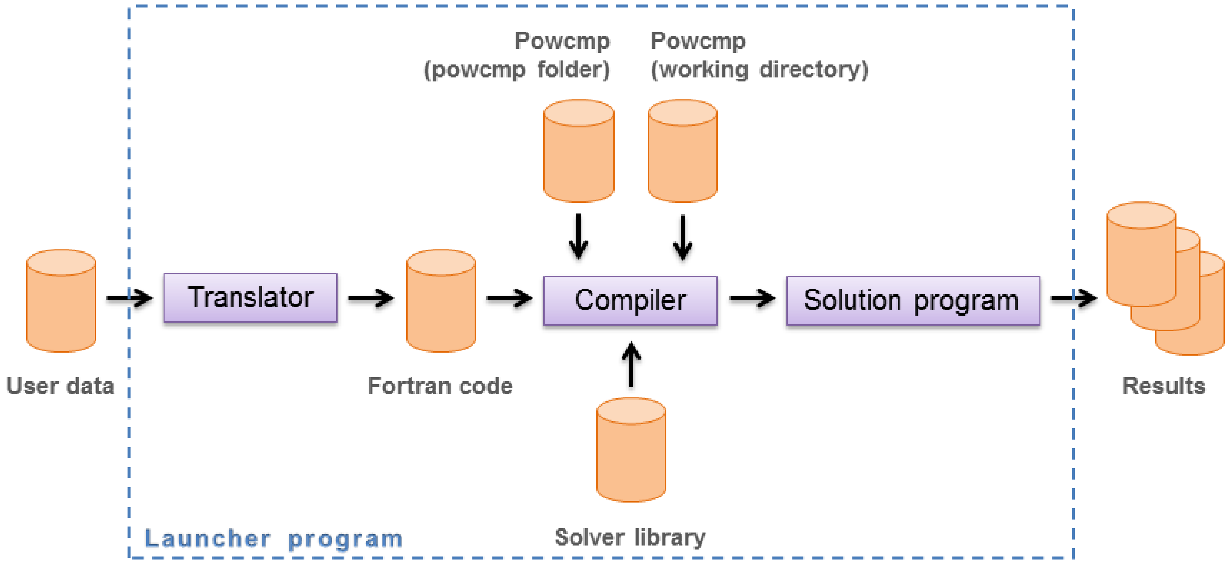


Figure 4.1-2 Diagram of the solver architecture

4.2 Input and output files

Usually, the whole process is launched with only one parameter: the name of the main input file. Then, several output files are produced. In most cases, their name is based on the name of the input file, with a "zz_" prefix and a suffix which depends on the file type. The "zz_prefix" is a way to gather all these files in a file explorer. The following tables present the different INPUT and OUTPUT files:

Input files

Name	***File function***	***Description***
Filename	Main input file	This is the only input the user has to specify to the solver
(any name)	Other input files	Any arbitrary number of files, used by means of $INCLUDE instructions, or by sub-modeling techniques

Output files

Name	File function	Description
zz_filename_for.f zz_filename_com.fi	Fortran code file	These files contain the output of the Fortran-like language translation

zz_filename_for.o	Compiled code	Compiled code
zz_filename_prg_SYS	Solution program	Solution program (SYS refers to the operating system)
filename.log	Log file	This file relates the main events occurring during the translation task
filename.out	Text file	This file contains the standard output text generated during the calculation
filename.csv	Spreadsheet run report	This file contains useful information (convergence criteria) to check the convergence of the simulation
modelname.temp.h5	result file	Result file in the HDF5 format
(any name)	Other specific output files	Other specific output files, depending on the solver subroutines called during the calculation
(any name)	User specific output files	Output files which may be created by user subroutines

4.3 Structure and content of the input file

A model is a combination of different blocks of instructions. Some are used to define the electrical and/or thermal network, others to declare constants, variables or arrays, and finally some are executive instructions called at different moments of the solver execution.

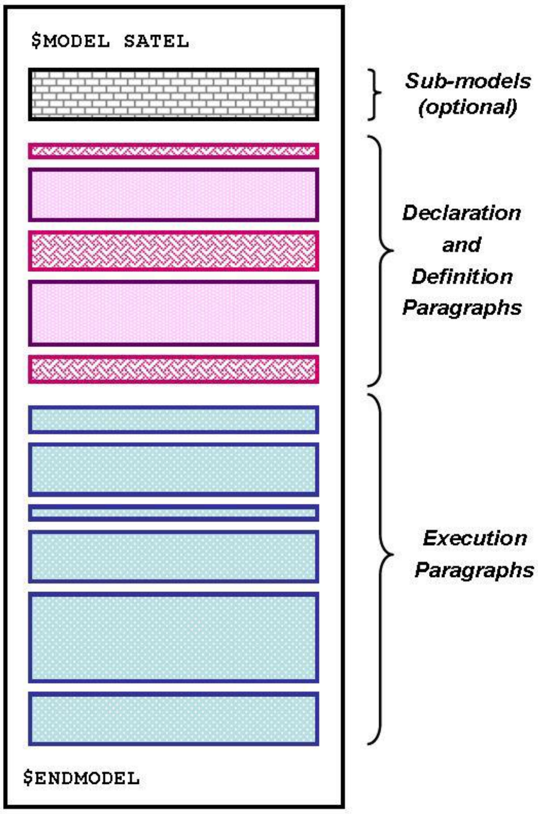


Figure 4.3-1 Schematic view of a Thermisol-Power input file

In this section, only the information specific to perform electrical simulations are presented. For a more complete description of the structure of the input file, please refer to the Thermisol User Manual.

4.3.1 Definition paragraphs

Those paragraphs describe the network configuration, i.e. the nodes and the components. The symbols related to the defined network are called "Mortran symbols" (node numbers, voltage, components, component parameters …).

It is possible to use Mortran expressions to affect a nodal property or a component parameter. The Mortran syntax is described in "The Mortran language" section of the Thermisol User Manual.

Note that any network data (nodal entity or component) defined with formulas or Mortran expressions will be automatically added to the UGENMOR subroutine (see the Mortran language section of the Thermisol User Manual).

$NODES

This paragraph contains the declaration of electrical (and thermal) nodes, defined by a node number and initial properties. The syntax is as follows:

Type number = 'name', properties_list;
where:
- **type** = U (free potential) or V (fixed potential)
- **number** = a positive integer
- **'name'** = a string with single quotes
- **properties_list** = initial values of nodes properties (voltage), separated by commas.

A fixed potential node is used to represent a mathematical boundary to the problem, the voltage being prescribed by the user as a fixed value, or one varying with respect to time, or possibly some other quantity. Boundary node voltages are not changed by the solution.
A free potential node is one whose voltage will be calculated during solution.

$CONDUCTORS

This paragraph contains the declaration of electrical components between electrical and thermal nodes. For any component kind, the structure of its declaration follows the same rules. A list of standard components provided with the Power application is given in the next chapter. A dedicated chapter will then introduce user's defined components.
The declaration of a component is written as follow:
name [idx, jdx] = Component type (connector list) = [parameter list];
where:
- **name** is a custom name given by the user for this component instance. Each component shall have a distinct name (unless indexes are used so to identify each component).
- **[idx, jdx]** is an optional set of index that may be used to complete the component name so to get an easy declaration of several components within a loop. One or two indexes may be used.
- **Component type** is the generic name of the component declared. It may be a component type from the standard library or a user's component.
- The **connector list** is used to connect all the connectors defined for the component type to electrical or thermal node.
- The **parameter list** is used to set the component parameter values. Expressions can be used.

Example of a $CONDUCTORS paragraph (1/3)

```
$CONDUCTORS
Res = Resistance (Uin=1, Uout=2) = [ R=50.0 ];
Diod = Diode (Umin=2, Umax=3) = [ Ron=0.001, Vd=0.2 ];
```

Example of a $CONDUCTORS paragraph using a loop declaration (2/3)

```
DO C = 1,10
Res[C] = Resistance (Uin=1, Uout=2) = [ R=50.0 ];
END DO
```

Example of a $CONDUCTORS paragraph using the Mortran syntax (3/3)

```
MyComp = MyComponent (Uin=1, Uout=2) = [ Param= expression ];
```

The components are defined with a set of connectors and parameters.
The electrical connector names start by the letter U and are also associated to currents with the same name but a letter I instead of the prefix U.
The thermal connector names start by the letter T. A thermal connector gives access to both the temperature and the internal dissipation (Q) of the connected thermal node.

Note that the dissipation of a thermal connector Q instantiate the thermal nodal entity QI which shall be reserved, i.e. not instantiate elsewhere.
It is possible to collect several dissipations from different components as long as the component equations set the dissipation to zero at initialization and increment it (i.e. uses a += operation) at finalization rather than using an instantiation (i.e. with the = operator).

The component parameters are defined in the powcmp file. They are of two types:

- External **variables** which may be inputs or outputs from the components. Those are accessible from the model at definition or in execution instructions.
- **Local** variables which are only visible to the component internal code.

Any component is also described by its equations in the powcmp file. There are three types of equations:

- Initialization (**Init**) used to initialize variables that do not required to be updated during the resolution.
- Resolution (**Solve**) which is the intrinsic behavior of the component. It is expected to value the current on each electrical connector (and also the jacobian matrix dIdV for numerical convergence purpose).
- Finalization (**End**) used to post-process the converged result, typically to value output parameters and thermal dissipations.

Sign convention

The convention for the intensity sign on a component is as follow:

- A current that gets in a component through a connector is negative
- A current that gets out of a component through a connector is positive

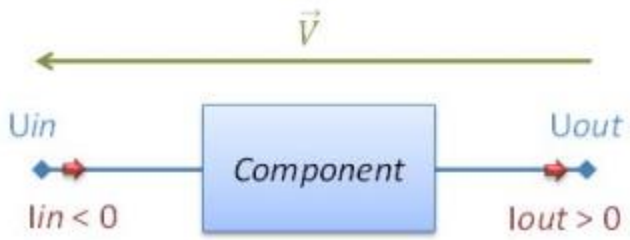


Figure 4.3-2 Sign convention for the intensity

GENMOR Code for components

As for thermal declaration, an automatic GENerated MORtran code is created when parameters are given by variables, formulas or functions.

However, for the Power add-on, it exists in fact one GENMOR code by component, in addition to general one. The general power GENMOR code is used to update the data at electric node declarations but for components dedicated codes are automatically created in order to update the inputs of the components each time their equations shall be called.
This mechanism ensures that all data are always consistent.

Accessing a component parameter

To use a component parameter, the following syntax can be used:
componentName [idx, jdx]->parameterName
Note that the connector properties (U, I or T, Q) are also accessible.

Example of a component parameter call

```
VSource = Vsource(Uin=1, Uout=3) = [Vs=INTRP1(U_TIMEN, Vtab, 1)];
Regul = LinearRegulation = [ Ireg = Vsource->Iout,
                             Vreg = V3,
                             Ilim = 10.,
                             VlimABS = 35.,
                             Vlim0 = 35. ];
SAR = ShuntRegulator(Uin=2, Uout=3) = [ MEA = Regul->MEA,
                                        Vd = 1.0,
                                        Rs = 0.01,
                                        Rshunt = 0.01 ];
```

4.3.2 Declaration paragraphs

$CONTROL

This paragraph contains the definition of variables used by the solution routines during computation. The control variables dedicated to electrical simulations are the following (please refer to the Thermisol User Manual for the other control variables):

Control variables specified by the user

Name	*Type*	*Typical value*	*Description*
U_RELXCA	Real	1e-4 V	Maximum permissible voltage change on one node between two iterations.
INBALI	Real	1e-5 A	Maximum permissible current imbalance (maximum allowable « epsilon » obtained on the electrical current budget on one node) **Warning:** INBALI should be much smaller (ratio 10 or 100 at least) than the values of current expected in the model. If not, the electrical current budget and thus the results could be wrong.
U_NLOOP	Integer	10000	Maximum number of iterations allowed for system convergence at

			each time step.
CMP_EPS	Real	1e-4	Component convergence tolerance
CMP_NLOOP	Integer	10	Maximum number of iterations allowed for component convergence.
U_DTIMEI *(transient only)*	Real	1 s	Input time step (it is advisable to have $U\_DTIMEI \leq DTIMEI$ where DTIMEI is the thermal time step)
U_DTMIN *(transient only)*	Real	1e-4	Minimum time step (should be less than or equal to U_DTIMEI)

Control variables computed during the solution

Name	*Type*	*Description*
U_RELXCC	Real	Maximum voltage change obtained between two iterations
U_NRLXCC	Integer	Node on which the maximum voltage change has been obtained
ENBALI	Real	Absolute current exchange on all electrical current budget
U_LOOPCT	Integer	Number of iterations performed
U_DTIMEU *(transient only)*	Real	Time step used
U_TIMEO *(transient only)*	Real	Initial time of current time step
U_TIMEN *(transient only)*	Real	End time of current time step

General control variables

Name	*Type*	*Description*
UCSV_FREQ	Integer	The power report will be written at this frequency in the csv control file. - Transient solver: if UCSV_FREQ=-1 the power report will be written at the same time as the thermal report - Steady solver: if UCSV_FREQ=-1 the power report will not be written at all

4.3.3 Execution paragraphs

$VPower

The VPower executive block is dedicated to electrical dependencies. Similarly to VTEMPERATURE (see Thermisol User Manual), this block is used to code general instructions that modify component parameters. This paragraph is called at each iteration of the electrical convergence loop.

Since electrical components are coded with possible dependencies through the GENMOR code (automatic generated Mortran), the use of this block might be very limited but still offers to implement instructions outside of the components and their parameters.

4.4 Library functions and subroutines

4.4.1 Solutions routines

The solutions routines allow the resolution of the electric network for steady-state or transient problems coupled or not with thermal convergence. These routines are usually called in the $EXECUTION paragraph.

SUBROUTINE STEAD_U

This subroutine computes the solution of an electrical steady-state problem.

Driving parameters

- **U_RELXCA:** Maximum potential change for one node over a convergence loop. The criterion is: U_RELXCC < U_RELXCA.
- **U_NLOOP:** Maximum number of iterations allowed. The criterion is: U_LOOPCT < U_NLOOP.
- **INBALI:** Maximum current sum imbalance allowed on electrical nodes. The criterion is: ENBALI < INBALI.

Execution schema

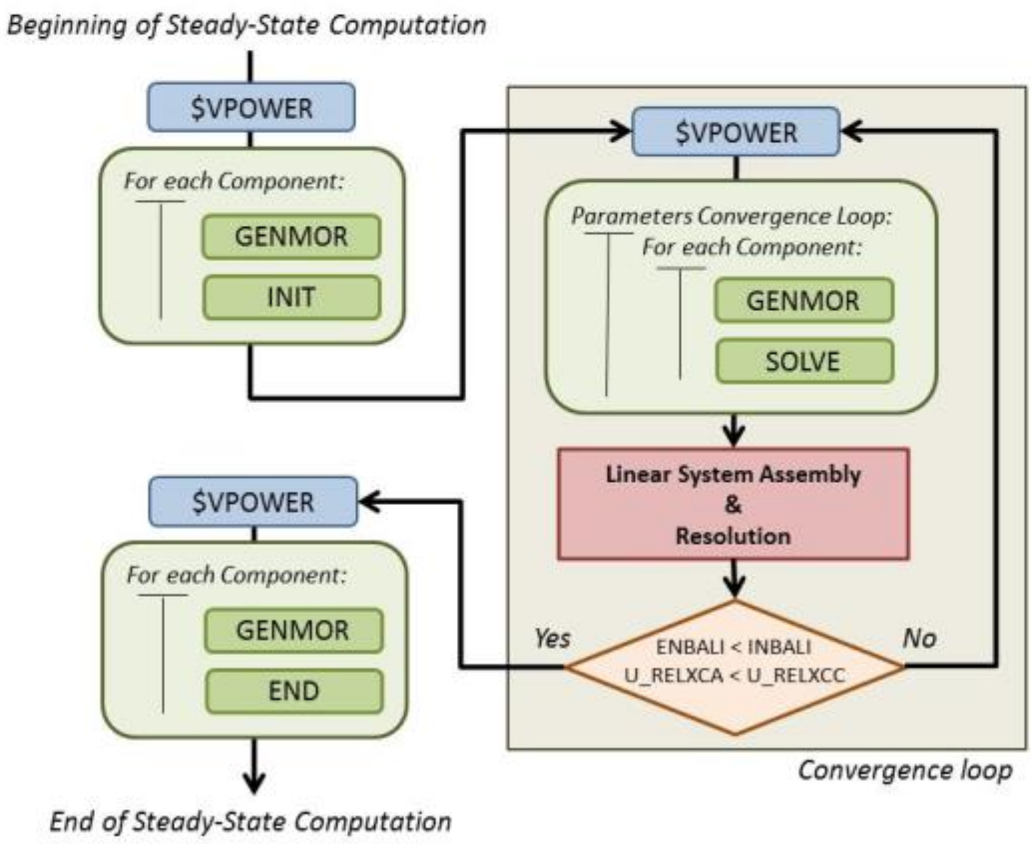


Figure 4.4-1 Execution schema for the STEAD_U solution routine

Linear System

The resolved linear system is the following:

$$\left(\frac{dI}{dU}\right)_{ij} \cdot (\Delta U)_i = \left(\sum_j I_{ij}\right)_i$$

The matrix of the system is the assembly of the Jacobian matrix of the component's I(V) equations. The second member is the sum of all currents at each electrical node. At convergence, this vector has to be equal to zero. The unknown vector resolved is the variation to be applied on the node potentials to get balanced current sums everywhere. For linear problems, the convergence is obtained into a single resolution of the linear system. Non-linear problems are solved by iterating on successive linearized systems. This is called a Newton resolution.

Resolution of the Linear System

As for the thermal solver SOLVFM, the linear system is solved by a Krylov algorithm based on a conjugate gradient method. In order to avoid over-resolution of the system (i.e. accurate resolution of an approximate linearized system), the convergence criteria of the Krylov iterations are automatically adapted according to the distance between the state of the solution and its convergence.

Damping method

One convergence loop is in fact a little more complex than presented on the previous diagram. From the initial state of the solution we compute the error (ENBALI), the resolution of the linear system leads to a new solution for which we can also compute the new error. According to those, the final solution of the current convergence loop will be adjusted on the descent direction by applying a damping factor.

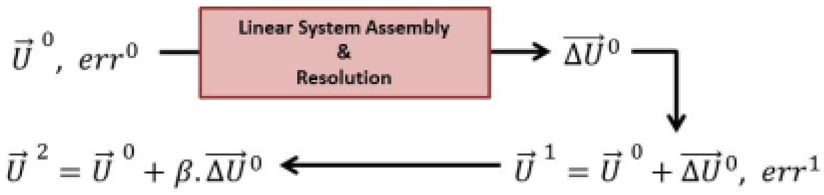


Figure 4.4-2 Convergence loop schema for the STEAD_U solution routine

Example of linear problem resolution

The following problem is completely linear. The initial state corresponds to the voltage of each node at their declarations (by default zero).

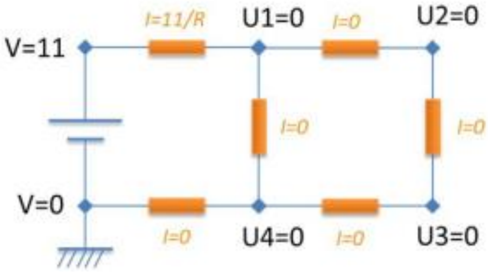


Figure 4.4-3 Example of linear problem resolution

To solve that problem, the STEAD_U algorithm will build the following matrix system:

$$\begin{pmatrix} 3/R & -1/R & 0 & -1/R \\ -1/R & 2/R & -1/R & 0 \\ 0 & -1/R & 2/R & -1/R \\ -1/R & 0 & -1/R & 3/R \end{pmatrix} . \begin{pmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \end{pmatrix} = \begin{pmatrix} 11/R \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The matrix is composed of the jacobian terms of each resistance summed at all nodes. For example, the first line of the matrix means that the current at node 1 will:

- Decrease of 3/R if U1 is increased by 1
- Increase of 1/R if U2 is increased by 1
- Increase of 1/R if U4 is increased by 1

On the secondary member, we have node current imbalances. In that case, the initial state implies that the node 1 is imbalanced by 11/R.

The direct resolution of that system is:

$$\begin{pmatrix} \Delta U_1 \\ \Delta U_2 \\ \Delta U_3 \\ \Delta U_4 \end{pmatrix} = \begin{pmatrix} 7/11 & 6/11 & 5/11 & 4/11 \\ 6/11 & 13/11 & 9/11 & 5/11 \\ 5/11 & 9/11 & 13/11 & 6/11 \\ 4/11 & 5/11 & 6/11 & 7/11 \end{pmatrix} . \begin{pmatrix} 11 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 7 \\ 6 \\ 5 \\ 4 \end{pmatrix}$$

The solution of the problem is:

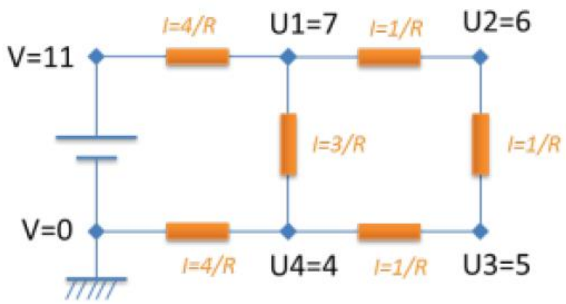


Figure 4.4-4 Solution of the linear problem

SUBROUTINE STEAD_UT

This subroutine computes the solution of coupled electrical / thermal steady-state problem.

Driving parameters

- ***Same as STEAD_U**: For the electric network resolution*
- ***Same as SOLVFM/SOLVIT:** For the thermal network resolution*

Execution schema

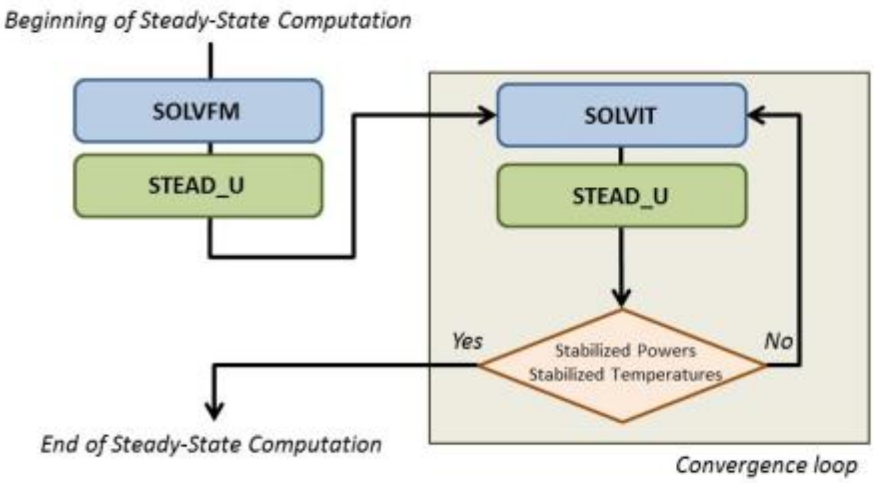


Figure 4.4-5 Execution schema for the STEAD_UT solution routine

Domain Decomposition Principle

To get the convergence of a coupled system of thermal and electrical networks we apply the principle of domain decomposition, meaning that the thermal and electrical domains are solved individually but share boundary conditions. In our case, the relationships between the two domains are the following:

- The electrical network depends on some thermal nodes temperatures
- The thermal network depends on dissipations resolved by the electrical network

The domain decomposition loop can be represented as follow:

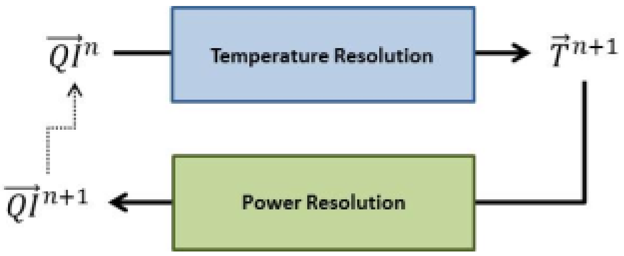


Figure 4.4-6 Domain decomposition loop for the STEAD_UT solution routine

Since the two domains are solved within a loop, the order for solving each domain is not very important. However, it is preferred to start by a temperature resolution because we may suppose that the dissipations from the electrical network represent a small amount compared to other thermal powers exchanged with thermal boundary conditions (radiative exchanges with space, external fluxes…).
The first thermal computation is using the routine SOLVFM, i.e. a Newton-Krylov approach, in order to solve globally the thermal system with a matrix system. During the convergence loop, we can assume that the temperature computation is a correction of the previous one according to the correction applied on the dissipated (or absorbed in the case of solar cells for example) powers solved by the electrical network. For these convergence loops, using the SOLVIT method, i.e. a Newton-Raphson approach, shall provide a faster convergence.
As for the STEAD_U algorithm, the resolution of systems embedded into a loop would lead to an over-resolution (i.e. solving accurately a domain for which the boundary conditions implied by the other domain are not yet stabilized). To prevent this phenomenon, the RELXCA and INBALI parameters that drives the resolution accuracy of the thermal and electrical domains respectively, are automatically adjust as the solution converges.
Note that the domain decomposition process may converge very quickly if the variation of the output powers from the electrical network is linear according to the input temperatures variation which is for example not the case in a solar panel study.

<u>*Damping method*</u>

In the case of non-linear behaviors, the domain decomposition approach may lead to oscillate around the solution and may even not converge. This oscillations phenomenon is represented as follow:

$$\overrightarrow{QI}^{n} \longrightarrow \vec{T}^{n+1} \longrightarrow \overrightarrow{QI}^{n+1} \longrightarrow \vec{T}^{n+2} \longrightarrow \overrightarrow{QI}^{n+2} = \overrightarrow{QI}^{n}$$

To prevent from this behavior, we adjust a damping coefficient on the powers variation computed by the electrical network.

$$\overrightarrow{QI}^{n} \longrightarrow \vec{T}^{n+1} \longrightarrow \overrightarrow{QI}^{n+1/2},$$
$$\overrightarrow{QI}^{n+1} = \overrightarrow{QI}^{n} + \beta.(\overrightarrow{QI}^{n+1/2}\text{-}\overrightarrow{QI}^{n}) \longrightarrow \vec{T}^{n+2}$$

This damping factor goes from 0.75 to 1.0 as we get close to the correct solution.

SUBROUTINE TRANS_U

This subroutine computes the solution of an electrical transient problem.

Driving parameters

- ***Same as STEAD_U*: *For the convergence of the electrical network***
- **TIMEO:** Start time of the simulation
- **TIMEN:** End time of the simulation
- **U_DTIMEI:** Input time-step
- **U_DTMIN:** Minimum time-step allowed
- **OUTINT:** Time-step for outputs

Execution schema

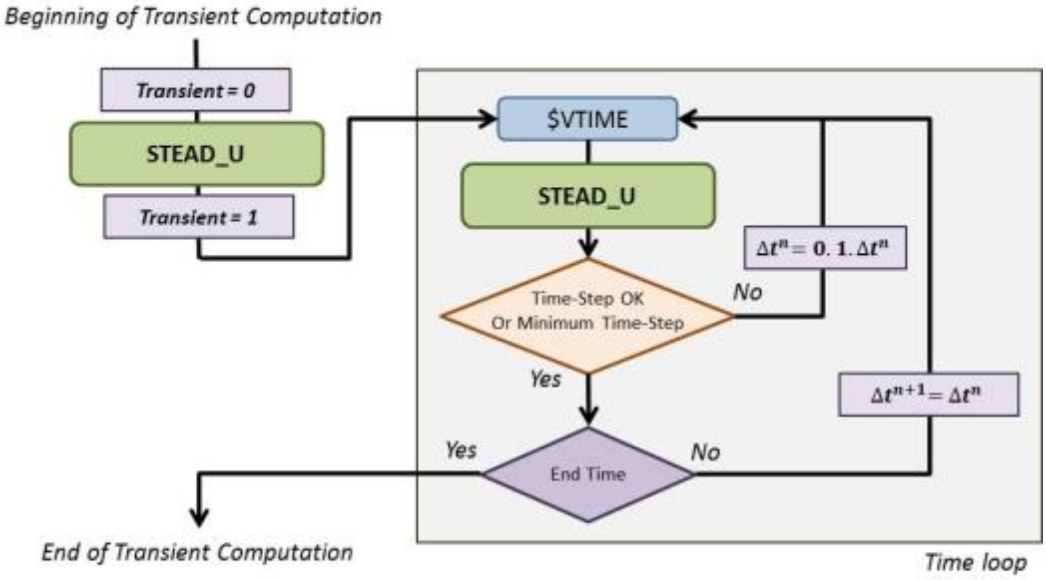


Figure 4.4-7 Execution schema for the TRANS_U solution routine

Transient Analysis

Unlike the thermal network for which the fundamental equation solved changes between the steady-state ($\sum_j Q_{ij} = 0$) and transient analysis ($\sum_j Q_{ij} = C_i \frac{dT_i}{dt}$), the fundamental equation solved for the electrical network remain the same ($\sum_j I_{ij} = 0$) since there is no electrical inertia in the nodes.
In the electrical network, the transient aspect of the simulation is intrinsic to the components. Indeed, solving a transient problem is, at system level, equivalent to solve a steady-state problem for which we use transient equations at component level.
The transient analysis starts by a classical steady-state analysis for which the global variable *Transient* is set to 0. This allows initializing the transient computation with a converged state for the initial time. The *Transient* variable is then switch to 1, meaning that further call to the component codes will be made in transient mode.

Note that the Transient variable is used in component codes to possibly apply different equations depending on the type of analysis (steady-state or transient).

<u>Implicit or Explicit time integration</u>

Since the transient equations are not driven at system level but at the component ones, the numerical approach to integrate time dependent data is embedded into the components codes.
However, since there is a convergence loop in the steady-state system resolution used by the transient analysis, coding implicit time integration is easy.
For example, if we take a capacitance, its equation is: $I = C\frac{dV}{dt}$
which can be numerically approximated by: $I = C\ \frac{V-V_{prev}}{\Delta t}$
where *V* is the capacitance voltage at the current time t^{n+1} and V_{prev} the saved value of this voltage at the previous computed time t^{n}.
Since the value of *V* is adjusted during the convergence loop at the current time t^{n+1}, it makes the resolution implicit, i.e. the solution of *I* at time t^{n+1} depends on parameters that are also evaluated at that time.

<u>Time-Step management</u>

As for thermal transient simulations, the computed times include the declared events and the specified times for the $OUTPUTS call (through the control variable OUTINT, see the Thermisol Solver User Manual).

SUBROUTINE TRANS_UT

This subroutine computes the solution of coupled thermal-electrical transient problem.

<u>Driving parameters</u>

- ***Same as TRANS_U***: *For the convergence of the electrical network*
- ***Same as SCRANK***: *For the convergence of the thermal network*

<u>Execution schema</u>

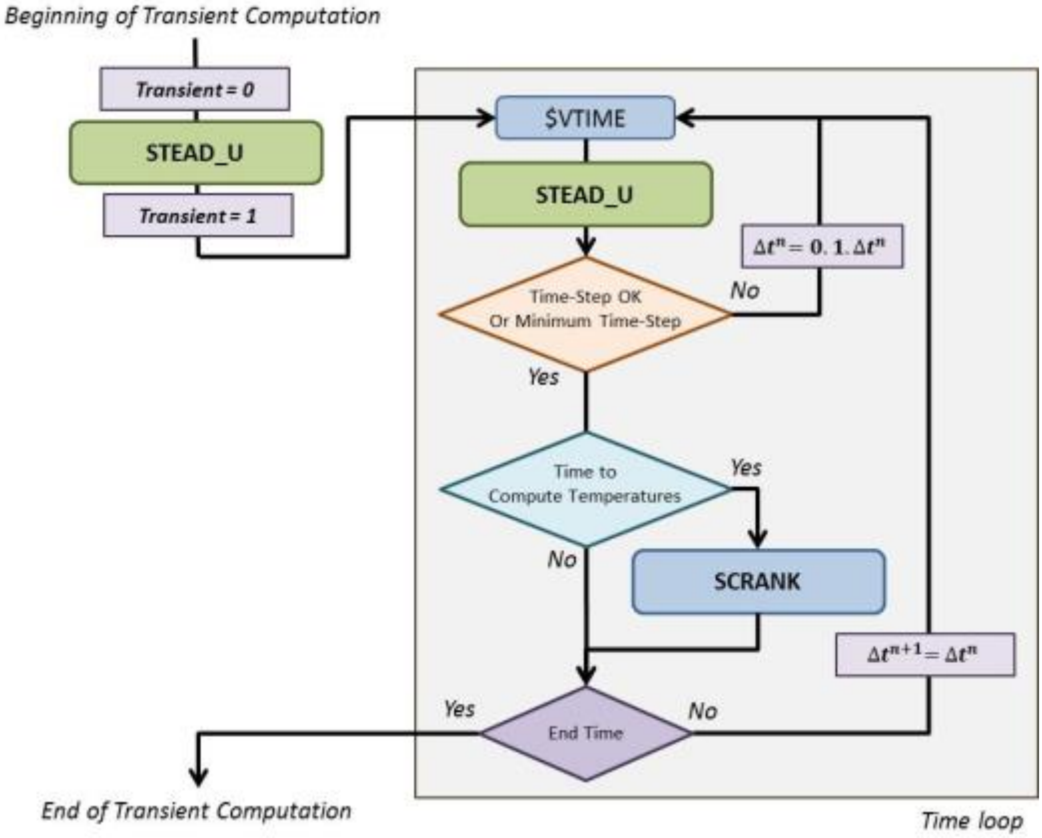
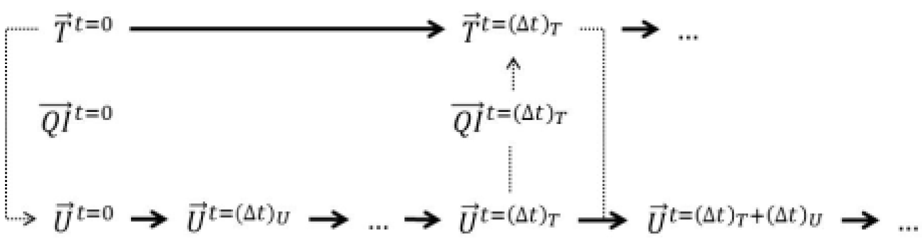


Figure 4.4-8 Execution schema for the TRANS_UT solution routine

<u>*Transient coupled resolution*</u>

The transient coupled simulation has a weaker cross-dependency between the thermal and electrical network. Because of the stronger thermal inertia, the thermal network is updated less often than the electrical one. We consider here that temperatures are constants over a thermal time-step, meaning that the electrical network can be computed without looping on temperatures updates.
The resolution of the electrical network is adjusted so to be synchronized with the thermal time-step.
Whenever the electrical resolution has reached the end of a thermal time-step, the thermal network is resolved according to its initial and final environment (*the thermal resolution is also implicit*).
The newly computed temperatures are then taken into account for the next electrical resolution, i.e. at its next time-step.



This weak thermal-electrical coupling can here be justified by the fact that we suppose the temperature variation over one thermal time-step not too important so that the approximation is acceptable. Secondly, a coupled convergence loop at each synchronized time would lead to greater computation time for which a reduction of the thermal time-step (leading to equivalent computation time) would give even better results.

A transient analysis shall always starts by a converge state of the electrical network. Indeed, an electrical steady-state computation is always performed at the beginning of a transient analysis. It is possible to call a coupled electrical-thermal steady-state resolution from the main $EXECUTION block of the model or to initialize temperatures by other means before calling TRANS_UT.

4.4.2 Data output routines

The data output routines can be used for results or data output.

SUBROUTINE UPRINT

This subroutine prints the electrical node potentials.

Example of a UPRINT output

```
UPRINT
MODULE TRANS_U
U_TIMEN = 0.2 (TIMEND = 0.2)
U_DTIMEU = 0.1 (DTIMEI = 0)
U_LOOPCT = 1 (U_NLOOP = 10000)
ENBALI = 4.4409e-16 (INBALI = 1e-05)
U_RELXCC = 0.23704 (U_RELXCA = 0.0001) at node 2 in model

MODEL    NODE   L      U
EXAMPLE  0 V   Mass   0.000000
EXAMPLE  1 U   Node   1 2.400000
EXAMPLE  2 U   Node   2 0.503704
```

SUBROUTINE IPRINT

This subroutine prints the electrical currents on each component's connector.

Example of a IPRINT output

```
IPRINT
MODULE TRANS_U
U_TIMEN = 0.2 (TIMEND = 0.2)
U_DTIMEU = 0.1 (DTIMEI = 0)
U_LOOPCT = 1 (U_NLOOP = 10000)
ENBALI = 4.4409e-16 (INBALI = 1e-05)
U_RELXCC = 0.23704 (U_RELXCA = 0.0001) at node 2 in model

MODEL     COMPONENT       I1         I2
EXAMPLE  Alim [ 0, 0]  -1.896296  1.896296
EXAMPLE  Res  [ 0, 0]  -1.896296  1.896296
EXAMPLE  Capa [ 0, 0]  -1.896296  1.896296

Unbalance sum : 0
```

SUBROUTINE UIPRINT

Calls both UPRINT and IPRINT.

5 Standard component library

This chapter presents all the standard components that are available within the Power application. For each component is given:

- a short presentation
- the associated symbol that is created in the graphic window
- a description of the connectors
- a description of the input parameters that can be set in the associated edition window
- a description of the output parameters that are computed by the solver
- a description of the specificities and/or limitations of the component

The detailed modelling equations are not given in the User Manual. They can be found in the .powcmp file of each component (SYSTEMA_INSTALL_DIR/applications/Power-XXX/powcmp/).

5.1 Primary sources

5.1.1 Standard Solar Array

This model can represent from a single solar cell to a whole solar array panel. It can be linked to the geometrical model (thanks to the Tfront connector) and it can take into account the solar and thermal fluxes computed with Thermica. It computes the electrical and thermal behavior of the (piece of) solar array based on the following elementary equation:

$$Iout = Isc\ (K - e^{\propto(V - Voc)})$$

where

$K = Losses\ \times Flux/FluxRef$

$\propto = \dfrac{\log\left(1.0 - \dfrac{Imp}{Isc}\right)}{Vmp - Voc}$

$V = Uout - Uin$

- **Symbol**

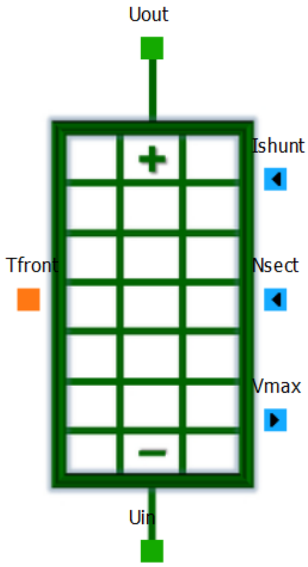


Figure 5.1-1 Solar Array symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
Tfront	Thermal	Solar Array front face node
Ishunt	Logical	Input parameter. Shunted current used for thermal calculation (generally connected to a constant parameter or a shunt regulator)
Nsect	Logical	Input parameter. Solar array sections connected to the bus (generally connected to a constant parameter or a digital regulator)
Vmax	Logical	Output parameter. Maximum instantaneous SA power point voltage (generally connected to the MPPT)

Note: the logical connectors may not be connected. In that case, the default value of the input parameters (Ishunt and Nsect) will be used for the computation. The value of the output parameter (Vmax) will be computed but it will not be directly used as input of another component.

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Nsect	Integer	-	1	SA sections connected to the bus. (Logical connector)
Ishunt	Real	A	0	Shunted current that is used for thermal calculation (Logical connector)
CellS	Integer	-	1	Number of cells in series
CellP	Integer	-	1	Number of cells in parallel
AreaCorrection	Real	-	1	Area correction factor - 1: Only the active part of the panel

				(covered by sun cells) is taken into account. Fluxes that impact inactive parts of the panel are lost. Cell's temperature will then be under-estimated - 0: Tfront is the average temperature of the entire panel. Cell's temperature will then be over-estimated because of local thermal gradient due to the cell's absorption of power.
APanel	Real	m^2	A:Tfront	Panel Area *Note that the substring "Tfront" in the APanel parameter field will be automatically replaced with the corresponding thermal node number when running the Power module.*
NsectTot	Integer	-	1	Total number of SA sections
Flux	Real	W/m^2	(QS:Tfront + QA:Tfront) / A:Tfront / ALP:Tfront	Solar flux associated to the Tfront thermal node. *Note that the substring "Tfront" in the Flux parameter field will be automatically replaced with the corresponding thermal node number when running the Power module.*
KThresh	Real	-	0.001	Thershold (Direct solar flux / Reference flux) below the one there is no output current
Losses	Real	-	1	Loss factor (between 0 and 1, 1 means 0% loss)
ACell	Real	cm^2	1	Cell size
TRef	Real	°C	28	Reference temperature
FluxRef	Real	W/m^2	1367	Reference flux
Isc0	Real	mA/cm^2	16.77	Reference short circuit current density
dIsc0	Real	$mA/K/cm^2$	0.01060	Coefficients of thermal dependence at reference temperature and infinite radiation dose
Voc0	Real	mV	2667	Reference open circuit voltage
dVoc0	Real	mV/K	-6.0	Coefficients of thermal dependence at 0 and infinite radiation dose
Vmp0	Real	mV	2371	Reference maximum power point Voltage
dVmp0	Real	mV/K	-6.1	Coefficients of thermal dependence at 0 and infinite radiation dose
Pmp0	Real	mW/cm^2	38.11	Power at the maximum power point
dPmp0	Real	$mW/K/cm^2$	-0.07284	Coefficients of thermal dependence at 0 and infinite radiation dose

- **Output parameters**

Name	***Type***	***Unit***	***Description***

Vmax	Real	V	SA maximum power point voltage
Imax	Real	A	SA maximum power point current
Pmax	Real	W	SA maximum power point
Isc	Real	A	SA short circuit current
Voc	Real	V	SA open circuit voltage
Pout	Real	W	SA delivered power
SA_Matching_Coef	Real	%	SA matching factor
Pmp	Real	W	SA maximum power point for Flux=FluxRef
Vmp	Real	V	SA maximum power point voltage for Flux=FluxRef
Imp	Real	A	SA maximum power point current for Flux=FluxRef

- **Specificities and limitations**

It is possible to define the cell characteristics directly in the schematic diagram edition window or by using a cell file. For the second option, just select “Yes” in the field “Use a cell file” and choose the cell file that you want to use.

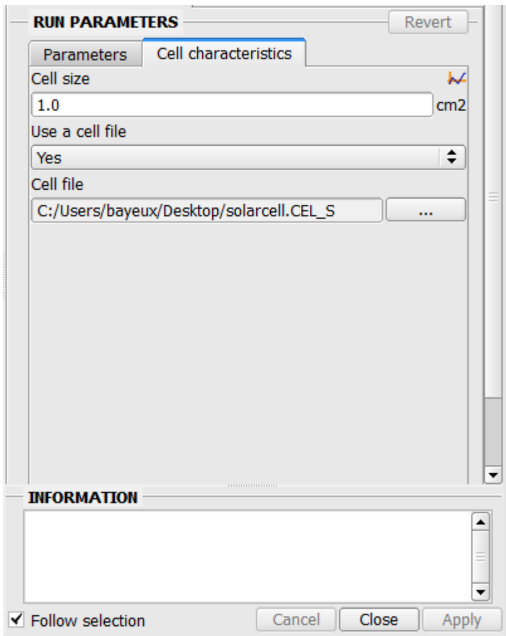


Figure 5.1-2 Cell characteristics definition using a cell file

The cell file is a text file which should have the following format:

```
#
#  -------------------
# |      Comments      |
#  -------------------
#
$PARAMETERS
TREF = 28.0
PHI_REF = 1367
ICC_0 = 16.77
```

```
ICC_D0 = 0.01060
VOC_0 = 2667.0
VOC_D0 = -6.0
VMAX_0 = 2371.0
VMAX_D0 = -6.1
PMAX_0 = 38.11
PMAX_D0 = -0.07284
$END
```

The files should begin with the $PARAMETERS (or $PARAMETRES) instruction and end with a $END instruction. It is possible to add comments using the character #. The cell characteristics should be specified with a specific name which is different from the one used in Power Systema. Here is the correspondence table:

Cell characteristic name in Power Systema	***Cell characteristic name in cell file***
TRef	TREF
FluxRef	PHI_REF
Isc0	ICC_0
dIsc0	ICC_D0
Voc0	VOC_0
dVoc0	VOC_D0
Vmp0	VMAX_0
dVmp0	VMAX_D0
Pmp0	PMAX_0
dPmp0	PMAX_D0

The order in which the parameters are declared in the cell file does not matter. If some parameters are missing in the cell file, the default value of these parameters will be used.

Warning:
The name of the .powcmp file must start with “SolarArray” to be considered as a Solar Array component within the Power application (especially if a cell file is used to set the cell characteristics). SolarArray.powcmp, SolarArray_v2.powcmp and SolarArrayNew.powcmp are three examples of correct name for the powcmp file. However, the correct behavior of the component is not guaranteed if it is called MySolarArray.powcmp or SA.powmcp for instance.

5.2 Secondary sources

5.2.1 Standard battery

This model is a Li-Ion cell model. It can represent from a single battery cell to a whole battery. It computes the electrical behavior of the (piece of) battery based on the following elementary equation:

$$Iin = \frac{Vbat - CellS \times EintCell}{\frac{CellS}{CellP} \times RintCell + Rharness}$$

where

$Vbat = Uout - Uin$

$RintCell = 0.001 * (R0 \; + \; R1 \times DOD \; + \; R2 \times DOD^2 \; + \; R3 \; \times DOD^3)$

- **Symbol**

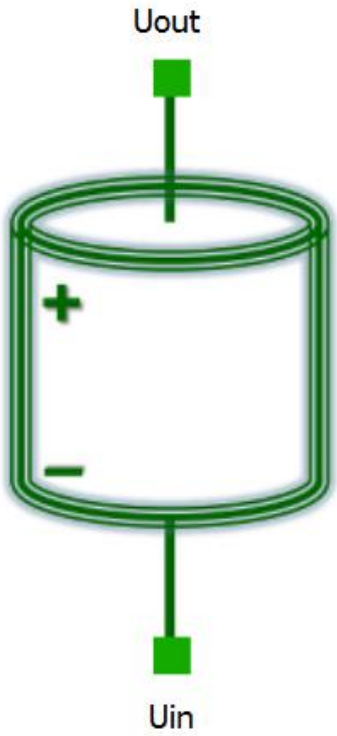


Figure 5.2-1 Battery symbol

- **Connectors**

Name	*Type*	*Description*
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
CellS	Integer	-	1	Number of cells in series
CellP	Integer	-	1	Number of cells in parallel
Qnom	Real	Ah	3	Cell nominal capacity
Rharness	Real	Ω	0	Total resistance of the battery internal wiring
SOCInit	Real	%	100	Initial state of charge value wrt remaining capacity
Ilim	Real	A	150	This is used in case there are 2 solutions of (V,I) leading to a given output power (so the lowest I is taken as solution)
E0	Real	V	4.2	Internal voltage polynomial coefficient
E1	Real	V	-3.1e-03	Internal voltage polynomial coefficient
E2	Real	V	-4e-04	Internal voltage polynomial coefficient
E3	Real	V	9.0e-06	Internal voltage polynomial coefficient
E4	Real	V	-6.0e-08	Internal voltage polynomial coefficient
E5	Real	V	-1.4e-22	Internal voltage polynomial coefficient
E6	Real	V	4.45e-25	Internal voltage polynomial coefficient
R0	Real	mΩ	230.296	Internal resistor polynomial coeffcient
R1	Real	mΩ	-0.627	Internal resistor polynomial coeffcient
R2	Real	mΩ	0.01826	Internal resistor polynomial coeffcient
R3	Real	mΩ	0.0	Internal resistor polynomial coeffcient

- **Output parameters**

Name	***Type***	***Unit***	***Description***
SOC	Real	%	State of charge wrt the remaining capacity
RintBat	Real	Ω	Battery internal resistance
EintCell	Real	V	Internal cell voltage
Vcell	Real	V	Cell voltage
DOD	Real	%	Depth of charge
WhCharged	Real	W	Battery charged energy
WhDischarged	Real	W	Battery discharged energy
PdisBat	Real	W	Instantaneous dissipation of battery
Vbat	Real	V	Battery voltage
RintCell	Real	Ω	Cell internal resistance
Wh	Real	W	Battery instantaneous energy

5.3 Regulators

5.3.1 Shunt Regulator

This model enables to simulate an interface of the type S3R, or a regulator of the type SPOT.

- **Symbol**

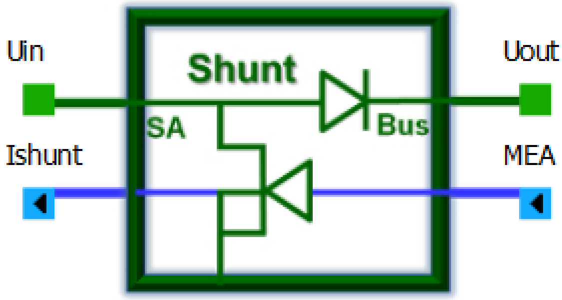


Figure 5.3-1 Shunt Regulator symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal, generally connected to the Solar Array
Uout	Electrical	Output terminal, generally connected to the Bus
MEA	Logical	Input parameter (must be connected). Error voltage piloting signal for the regulator (generally connected to a constant parameter or a linear regulation)
Ishunt	Logical	Output parameter (may be connected). Current to be shunted (generally connected to the Solar Array)

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
MEA	Real	-	Must be connected	Error voltage piloting signal for the regulator
Vd	Real	V	0.6	Voltage drop of the diode
Rs	Real	Ω	0.005	Series resistance
Rshunt	Real	Ω	0.01	Shunt resistance

- **Output parameters**

Name	Type	Unit	Description
Ishunt	Real	A	(if connected)
Pd	Real	W	Instantaneous power dissipation
Pin	Real	W	Input power
Pout	Real	W	Output power
Eff	Real	-	Instantaneous energy efficiency
EffAve	Real	-	Average energy efficiency

5.3.2 MPPT

This model allows connecting a Solar Array component to a bus or to a stockage element (secondary source). It regulates the Solar Array voltage to its maximum power point voltage.

- **Symbol**

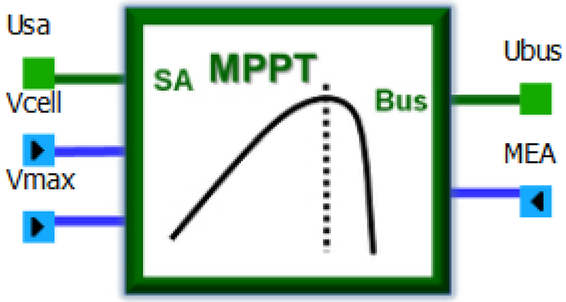


Figure 5.3-2 MPPT symbol

- **Connectors**

Name	Type	Description
Usa	Electrical	Input terminal, generally connected to the Solar Array
Ubus	Electrical	Output terminal, generally connected to the Bus
Vcell	Logical	Input parameter (must be connected to the output terminal of the Solar Array thanks to the “get V” component). Voltage to be regulated by MPPT.
Vmax	Logical	Input parameter (must be connected to Vmax connector of the Solar Array). Maximum power point voltage.
MEA	Logical	Input parameter (must be connected to a constant parameter or a linear regulation for instance). Error voltage piloting signal for the regulator.

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Vcell	Real	V	Must be connected	Voltage to be regulated by MPPT
Vmax	Real	V	Must be connected	Maximum power point voltage
MEA	Real	-	Must be connected	Error voltage piloting signal for the regulator
Vinit	Real	V	0.0	Initialization of Vcell parameter
effMPPT	Real	-	1.0	Efficiency of MPPT - if effMPPT > 0: fixed constant efficiency - else: use efficiency coefficients
a1	Real	-	0.921	Efficiency coefficient (only used if effMPPT≤ 0)
a2	Real	-	0.0	Efficiency coefficient (only used if effMPPT≤ 0)
a3	Real	-	0.0	Efficiency coefficient (only used if effMPPT≤ 0)
b1	Real	-	0.0019	Efficiency coefficient (only used if effMPPT≤ 0)
b2	Real	-	0.920	Efficiency coefficient (only used if effMPPT≤ 0)
b3	Real	-	0.0	Efficiency coefficient (only used if effMPPT≤ 0)
c1	Real	-	0.002	Efficiency coefficient (only used if effMPPT≤ 0)
c2	Real	-	0.022	Efficiency coefficient (only used if effMPPT≤ 0)
c3	Real	-	0.0	Efficiency coefficient (only used if effMPPT≤ 0)
maxPout	Real	W	1000.0	Maximum output power
minVsa	Real	V	0.0	Minimum input voltage of MPPT
maxVsa	Real	V	100.0	Maximum input voltage of MPPT
Kmea	Real	-	100.0	Internal regulation constant
Kinteg	Real	-	-1.0	Coefficient applied to the integral - if Kinteg < 0: shunt type regulation - else: series type regulation

- **Output parameters**

Name	*Type*	*Unit*	*Description*
Pin	Real	W	Input power from Solar Array
Pout	Real	W	Output power to bus
Efficiency	Real	%	Instantaneous efficiency of MPPT
Dissipation	Real	W	Instantaneous dissipation of MPPT
aveEff	Real	-	Average energy efficiency
MEAint	Real	-	Internal MEA signal
Pcte	Real	-	Intermediate variable used only if effMPPT ≤0
K1	Real	-	Intermediate variable used only if effMPPT ≤0
K2	Real	-	Intermediate variable used only if effMPPT ≤0
FlagMea	Integer	-	This flag is set so to switch to a power regulation within a time-step if the MEA becomes positive
Dampt	Real	-	Dampt factor
MEAmean	Real	-	Internal MEA signal

5.3.3 BCR / BDR

This model allows connecting a secondary source to a regulated bus for charge and discharge.

- **Symbol**

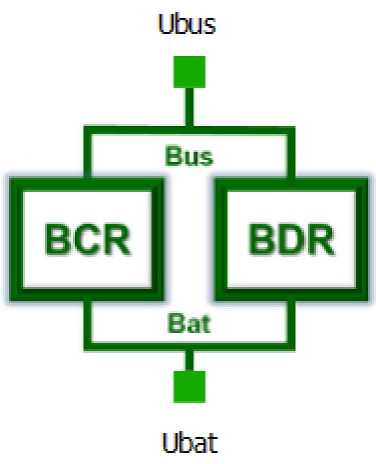


Figure 5.3-3 BCR BDR symbol

- **Connectors**

Name	*Type*	*Description*
Ubat	Electrical	Battery electrical node
Ubus	Electrical	Bus electrical node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Vbus	Real	V	34.0	Regulation bus voltage
effBCR	Real	-	1.0	BCR efficiency
effBDR	Real	-	1.0	BDR efficiency
IbdrLim	Real	A	100.0	Maximum limit of discharge current
K	Real	-	1.0	Regulation constant

- **Output parameters**

Name	*Type*	*Unit*	*Description*
Efficiency	Real	-	Instantaneous efficiency
Dissipation	Real	W	Instantaneous dissipation of the converter
Pbus	Real	W	Bus power
Pbat	Real	W	Battery power

5.4 Current/Voltage regulations

5.4.1 Linear Regulation

This module pilots a regulator taking into account the current and voltage limitations of a secondary source. This regulation is linear.

- **Symbol**

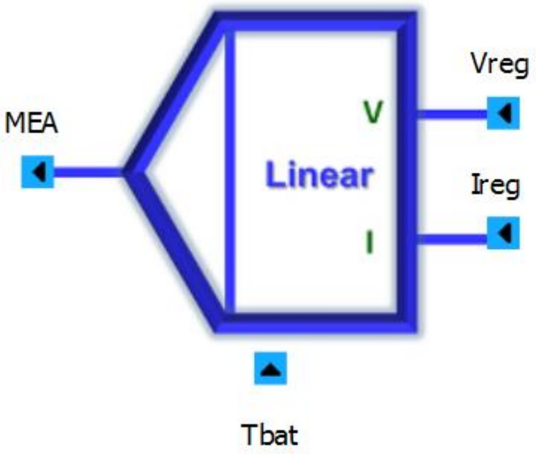


Figure 5.4-1 Linear Regulation symbol

- **Connectors**

Name	*Type*	*Description*
Vreg	Logical	Input parameter (must be connected to output terminal of the battery thanks to a “Get V” component for instance). Voltage to be regulated.
Ireg	Logical	Input parameter (must be connected to a “Resistance Get I” component to get the battery current for instance). Current to be regulated
Tbat	Logical	Input parameter (may be connected to a constant parameter for instance).

MEA	Logical	Battery temperature Output parameter (may be connected to a Regulator, typically a Shunt Regulator or MPPT). Error voltage piloting signal for the regulator.

If Tbat is not connected, the influence of the battery temperature will not be taken into account for the regulation.

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Vreg	Real	V	Must be connected	Voltage to be regulated
Ireg	Real	A	Must be connected	Current to be regulated
Tbat	Real	°C	0 (may be connected)	Battery temperature
Ilim	Real	A	50.0	Current limitation value
VlimABS	Real	V	50.0	Absolute limitation voltage
Vlim0	Real	V	50.0	Voltage limitation at 0°C
dVlim	Real	V/K	0.0	Variation coefficient of limitation voltage depending on temperature with respect to 0°C
epsilon	Real	V	0.05	Epsilon for the threshold of the Taper voltage
deltaV	Real	V	0.5	Delta V for the taper voltage computation
K	Real	-	100.0	Regulation constant

- **Output parameters**

Name	***Type***	***Unit***	***Description***
MEA	Real	-	Error voltage piloting signal for the regulator.
TIlim	Real	min	Current limitation duration
Ttaper	Real	min	Voltage limitation duration
Vlim	Real	V	Regulation voltage
Is_taper	Integer	-	Flag for Taper voltage computation

5.5 Basic components

5.5.1 Resistance

This model simulates a resistor.

- **Symbol**

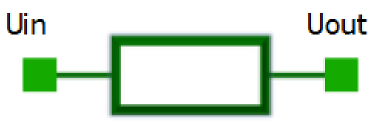


Figure 5.5-1 Resistance symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
R	Real	Ω	1.0	Resistance value

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Resistance voltage
Pdis	Real	W	Resistance dissipation

5.5.2 Resistance (Get I)

This model simulates a resistor. This component is the same as the previous one with an additional connector to get the value of the output current.

- **Symbol**

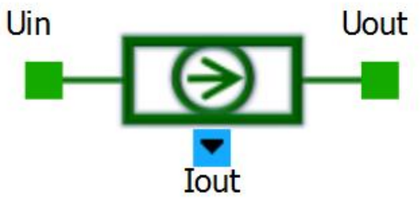


Figure 5.5-2 Resistance (Get I) symbol

- **Connectors**

Name	*Type*	*Description*
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
Iout	Logical	Output parameter (may be connected). Output current value of the resistance.

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
R	Real	Ω	1.0	Resistance value

- **Output parameters**

Name	*Type*	*Unit*	*Description*
V	Real	V	Resistance voltage
Pdis	Real	W	Resistance dissipation

5.5.3 Diode

This model simulates a perfect diode.

- **Symbol**

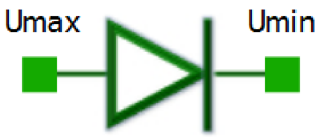


Figure 5.5-3 Diode symbol

- **Connectors**

Name	***Type***	***Description***
Umax	Electrical	Input terminal
Umin	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Vd	Real	V	0.6	Diode's conduction voltage threshold
Ron	Real	Ω	1.0e-6	Diode residual resistance

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Diode voltage
Pdis	Real	W	Diode dissipation

5.5.4 Capacitor

This model simulates an ideal capacitor.

- **Symbol**

Figure 5.5-4 Capacitor symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Capa	Real	F	1.0e-9	Capacity value

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Capacity voltage

5.5.5 Inductor

This model simulates an ideal inductor.

- **Symbol**

Figure 5.5-5 Inductor symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
L	Real	H	1.0e-3	Inductance Value

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Inductor voltage

5.5.6 Vdrop

This model simulates a voltage drop (multiplied by a gain). This is a component which adjusts the resistance value so to get the expected voltage drop.

- **Symbol**

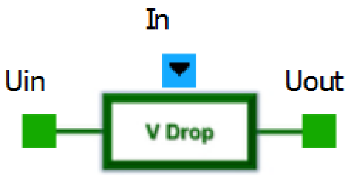


Figure 5.5-6 Vdrop symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
In	Logical	Input parameter (must be connected). Input voltage

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
In	Real	V	1.0 (must be connected)	Input voltage
Gain	Real	-	1.0	Gain

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Differential voltage
Pdis	Real	W	Vdrop dissipation

5.5.7 Switch

This model simulates a command switch with a given resistance. If the input control is greater than 0.6 the switch turns ON (Status = 1). If the input control is lower than 0.4 the switch turns OFF (Status = 0). If the input control is between 0.4 and 0.6 the status remains unchanged. The input current is calculated as follows:

$$Iin = Status * \frac{Uout - Uin}{Ron}$$

- **Symbol**

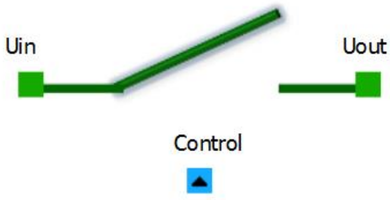


Figure 5.5-7 Switch symbol

- **Connectors**

Name	*Type*	*Description*
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
Control	Logical	Input parameter (must be connected). Command parameter

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Control	Real	-	1.0	Switch control.
Ron	Real	Ω	1.0e-3	Resistance value at ON status

- **Output parameters**

There is no output parameter.

5.5.8 Voltage source

This model generates a given voltage on the electrical line.

- **Symbol**

Figure 5.5-8 Voltage source symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Vs	Real	V	1.0	Source voltage

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Differential voltage

- **Specificities and limitations**

Warning:
The name of the .powcmp file must start with “Vsource” to have the right behavior. Vsource.powcmp, Vsource_v2.powcmp and VsourceNew.powcmp are three examples of correct name for the powcmp file. However, the correct behavior of the component is not guaranteed if it is called MyVsource.powcmp or Vs.powmcp for instance.

5.5.9 Controlled Voltage source

This model generates a signal controlled voltage on the electrical line.

- **Symbol**

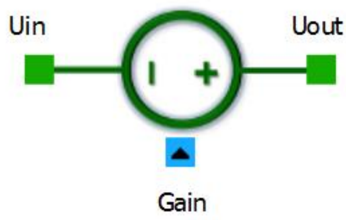


Figure 5.5-9 Controlled Voltage Source symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
Gain	Logical	Input parameter (may be connected). Gain

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Gain	Real	-	1.0 (may be connected)	Gain
Vs	Real	V	1.0	Source voltage

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Differential voltage

- **Specificities and limitations**

Warning:
The name of the .powcmp file must start with “Vsource” to have the right behavior. Vsource.powcmp, Vsource_v2.powcmp and VsourceNew.powcmp are three examples of correct name for the powcmp file. However, the correct behavior of the component is not guaranteed if it is called MyVsource.powcmp or Vs.powmcp for instance.

5.5.10 Current source

This model generates a given current in the electrical line.

- **Symbol**

Figure 5.5-10 Current Source symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Is	Real	A	1.0	Generated current

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Differential voltage

5.5.11 Controlled Current source

This model generates a signal controlled current in the electrical line.

- **Symbol**

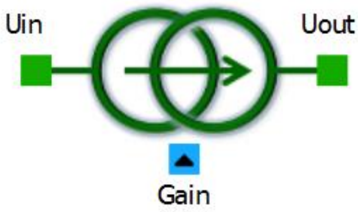


Figure 5.5-11 Controlled Current Source symbol

- **Connectors**

Name	***Type***	***Description***
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal
Gain	Logical	Input parameter (may be connected). Gain

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Gain	Real	-	1.0 (may be connected)	Gain
Is	Real	A	1.0	Generated current

- **Output parameters**

Name	***Type***	***Unit***	***Description***
V	Real	V	Differential voltage

5.5.12 Power load

This model consumes a given power (fixed or tabulated) augmented by an offset on the electrical line. This is a non-linear component which adjusts the current and voltage so to get the expected power load.

- **Symbol**

Figure 5.5-12 Power Load symbol

- **Connectors**

Name	*Type*	*Description*
Uin	Electrical	Input terminal
Uout	Electrical	Output terminal

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
P	Real	W	1.0	Load power consumption value
Offset	Real	W	0.0	Offset applied to the load power consumption value

- **Output parameters**

Name	*Type*	*Unit*	*Description*
Ptot	Real	W	Total load power consumption value

- **Specificities**

The parameter P can be set with a contact value or with a power profile using a txt input file. For the second option, just select “Yes” in the field “Use a txt file” and choose the txt file that you want to use. Then select the interpolation variable (only the time is available).

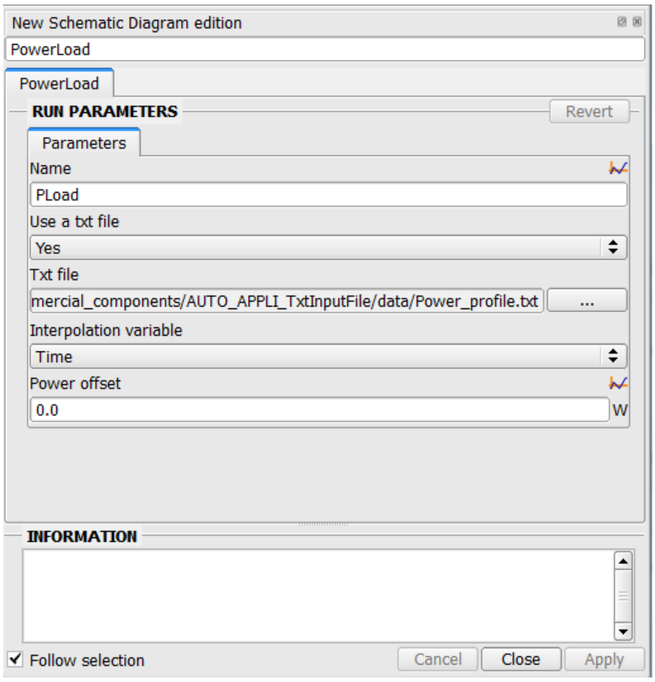


Figure 5.5-13 Power profile definition using a txt file

The txt file should have the following format (2 columns separated by a tab or a space, the first one is the time, the second one is the load power consumption value):

```
# Time       power  (some comments, for example name of each column)
0            300
10           301
20           302
30           303
40           304
50           305
60           304
70           303
80           302
90           301
100          300
200          305
300          310
400          315
500          320
600          315
700          310
800          305
900          300
```

If a txt file is used, the solver will compute the load power consumption value at each time by interpolating in the table thanks to the INTRP1 Thermisol routine.

5.5.13 Mass

This model allows fixing reference voltage (ground, 0V).

- **Symbol**

Figure 5.5-14 Mass symbol

- **Connectors**

Name	***Type***	***Description***
V0	Electrical	Reference electrical node

- **Input parameters**

No input parameter.

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Do not rename the component. The component will not have the right behavior if it is renamed.

5.6 Basic operators

5.6.1 Sum

This model allows adding two signals and multiplying the result by a gain.

$$Out = Gain \times (In1 + In2)$$

- **Symbol**

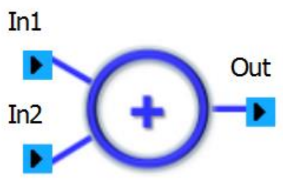


Figure 5.6-1 Sum symbol

- **Connectors**

Name	*Type*	*Description*
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.6.2 Diff (subtracter)

This model allows subtracting a signal to another signal and multiplying the result by a gain.

$$Out = Gain\ \times (In1 - In2)$$

- **Symbol**

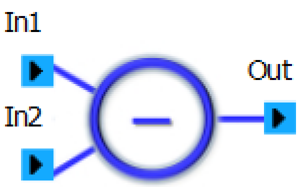


Figure 5.6-2 Diff symbol

- **Connectors**

Name	*Type*	*Description*
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.6.3 Product

This model allows multiplying two signals and multiplying the result by a gain.

$$Out = Gain \times In1 \times In2$$

- **Symbol**

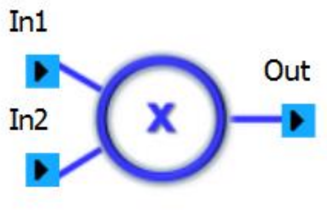


Figure 5.6-3 Product symbol

- **Connectors**

Name	***Type***	***Description***
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.6.4 Division

This model allows dividing a signal by another signal and multiplying the result by a gain.

$$Out = Gain \times \frac{In1}{In2}$$

- **Symbol**

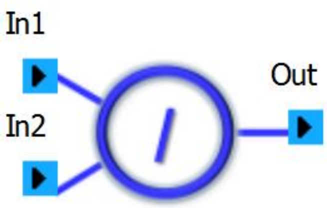


Figure 5.6-4 Division symbol

- **Connectors**

Name	*Type*	*Description*
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.6.5 Min

This model allows getting the minimum value between two signals and multiplying it by a gain.

$$Out = Gain \times MIN(In1, In2)$$

- **Symbol**

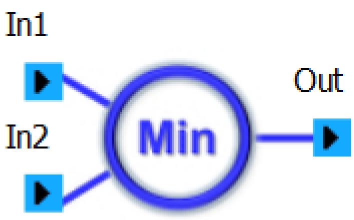


Figure 5.6-5 Min symbol

- **Connectors**

Name	***Type***	***Description***
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.6.6 Max

This model allows getting the maximum value between two signals and multiplying it by a gain.

$$Out = Gain \times MAX(In1, In2)$$

- **Symbol**

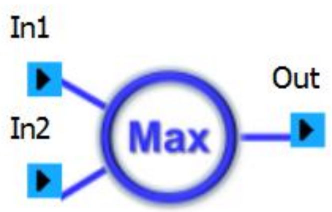


Figure 5.6-6 Max symbol

- **Connectors**

Name	***Type***	***Description***
In1	Logical	First input parameter node
In2	Logical	Second input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

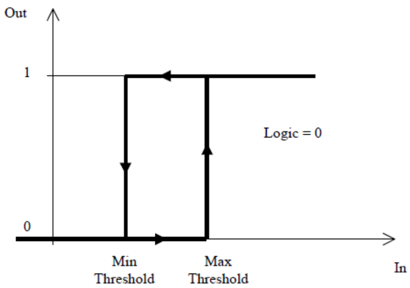
Component name must not be changed.

5.7 Other operators

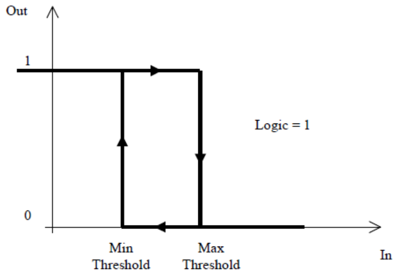
5.7.1 Comparator

This model allows generating a comparison signal with hysteresis effect.

If Logic is 0 (normal logic), Out is given in the following way depending on In:
If In > Max Threshold, Out = 1
If In < Min Threshold, Out = 0,
Otherwise Out is unchanged



If Logic is 1 (reversed logic), Out is given in the following way depending on In:
If In > Max Threshold, Out = 0
If In < Min Threshold, Out = 1,
Otherwise Out is unchanged



- **Symbol**

Figure 5.7-1 Comparator symbol

- **Connectors**

Name	*Type*	*Description*
In	Logical	Input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
In	Real	-	Must be connected	Input value to compare
MaxThresh	Real	-	1.0	Maximum threshold
MinThresh	Real	-	0.0	Minimum threshold
Logic	Integer	-	0	Normal (0) or reverse (1) logic command

- **Output parameters**

Name	*Type*	*Unit*	*Description*
Out	Real	-	Output logic value of the comparator

5.7.2 Integrator

This model allows generating an integration of a signal (for example the energy calculation).

$$Out = Initial + Gain \int_0^t In.dt$$

- **Symbol**

Figure 5.7-2 Integrator symbol

- **Connectors**

Name	***Type***	***Description***
In	Logical	Input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
In	Real	-	Must be connected	Input value to integrate
Gain	Real	-	1.0	Constant gain of the integrator
Initial	Real	-	0.0	Initial value of the integrator output

- **Output parameters**

Name	***Type***	***Unit***	***Description***
Out	Real	-	Output logic value of the comparator

5.7.3 Table Interpolation

This model allows linking a signal with another signal by a chosen tabulation. The tabulation should be given in one array of two columns or two arrays of one column. The first column/array defines In, the second column/array provides the corresponding Out value. Between two points, an interpolation is realized thanks to a Thermisol routine.

- **Symbol**

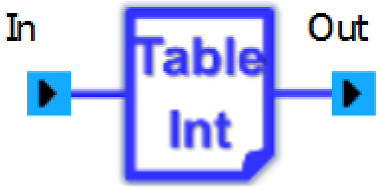


Figure 5.7-3 Table Interpolation symbol

- **Connectors**

Name	*Type*	*Description*
In	Logical	Input parameter node
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Routine	String	-	INTCYC	Thermisol interpolation routine (INTRP1, INTCY1 or INTCYC)
Order	Integer	-	1	Interpolation order
Period	Real	s	100.0	Period (only if the interpolation routine is INTCY1 or INTCYC
Array	String	-	ARR	Name of the array(s) used for interpolation

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.8 Parameters

5.8.1 Constant

This model fixes a parameter value to a constant value. This constant is declared in the $LOCAL paragraph. This type of parameter cannot be modified at any time (see the Thermisol User Manual for more details about the constant declaration in the $LOCAL paragraph).

- **Symbol**

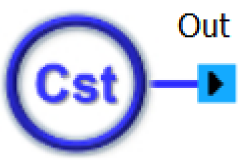


Figure 5.8-1 Constant symbol

- **Connectors**

Name	*Type*	*Description*
Out	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Value	Real	-	0.0	Constant value

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.8.2 Variable

This model fixes a parameter value. This vairable is declared in the $VARIABLES paragraph. This type of parameter can be modified at any time by the user (see the Thermisol User Manual for more details about the variable declaration in the $VARIABLES paragraph).

- **Symbol**

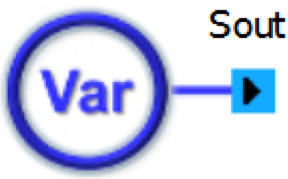


Figure 5.8-2 Variable symbol

- **Connectors**

Name	*Type*	*Description*
Sout	Logical	Output parameter node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Value	Real	-	0.0	Variable value

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.9 Interfaces

5.9.1 Get T

This element issues the value (to be connected to logical connectors) of the temperature of a thermal node. This value can be multiplied by a gain.

- **Symbol**

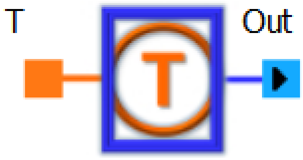


Figure 5.9-1 Get T symbol

- **Connectors**

Name	***Type***	***Description***
T	Thermal	Input thermal node
Out	Logical	Output logical node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Output gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.9.2 Get V

This element issues the value (to be connected to logical connectors) of the voltage of an electrical node. This value can be multiplied by a gain.

- **Symbol**

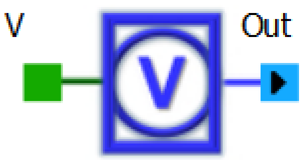


Figure 5.9-2 Get V symbol

- **Connectors**

Name	***Type***	***Description***
V	Electrical	Input electrical node
Out	Logical	Output logical node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Output gain	Real	-	1.0	Gain

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.10 Thermal components

5.10.1 GL

This model allows simulating a conductive exchange between two thermal nodes.

- **Symbol**

Figure 5.10-1 GL symbol

- **Connectors**

Name	*Type*	*Description*
T1	Thermal	First thermal node
T2	Thermal	Second thermal node

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Coupling value	Real	W/K	1.0	Conductive coupling value

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.10.2 GR

This model allows simulating a radiative exchange between two thermal nodes.

- **Symbol**

Figure 5.10-2 GR symbol

- **Connectors**

Name	***Type***	***Description***
T1	Thermal	First thermal node
T2	Thermal	Second thermal node

- **Input parameters**

Name	***Type***	***Unit***	***Default value***	***Description***
Coupling value	Real	m^2	1.0	Radiative coupling value

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

5.10.3 Wsource

This model allows simulating an internal dissipation applied on a thermal node.

- **Symbol**

Figure 5.10-3 Wsource symbol

- **Connectors**

Name	*Type*	*Description*
T1	Thermal	thermal node on which the dissipation is applied

- **Input parameters**

Name	*Type*	*Unit*	*Default value*	*Description*
Thermal power	Real	W	0.0	Internal dissipation

- **Output parameters**

No output parameter.

- **Specificities and limitations**

Component name must not be changed.

6 Create your own components

It is possible to create or use customized components.
To add a new component to the library, the following files are needed:

File Name	***Description***	***Location***
XXXX.jpeg	Picture of the component used by the schematic editor	in the “components” folder (by default in the installation directory or elsewhere if it has been modified by the user through the settings or the environment variable SYSTEMA_COMPONENTS_PATH)
XXXX.sysapp	XML file to configure the component edition in the GUI	in the “components” folder (by default in the installation directory or elsewhere if it has been modified by the user through the settings or the environment variable SYSTEMA_COMPONENTS_PATH)
XXXX.powcmp	text file containing the model (equation) of the component	in the “powcmp” folder (by default in the installation directory or elsewhere if it has been modified by the user through the environment variable POWCMP_DIR) or in the working directory

6.1 Sysapp file

The structure of a sysapp file is divided in 3 main parts:

- The global definition of the component (circled in dark below)
- The definition of connectors (circled in red below)
- The definition of input parameters (circled in green below)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SYSML CONTENTS="SCHEMBOX_DEFINITION" SYSML_VERSION="1.1">
    <SCHEMBOX_DEF NAME="Resistance" EXE="Resistance" TYPE="Electrical" SUBTYPE="Basic components" SYMBOL="./Resistance.jpg">
        <DESCRIPTION></DESCRIPTION>
        <CONNECTOR_DEF TYPE="ELEC" SIDE="LEFT" ROLE_TITLE="Uin" ROLE_DESC="Uin"/>
        <CONNECTOR_DEF TYPE="ELEC" SIDE="RIGHT" ROLE_TITLE="Uout" ROLE_DESC="Uout"/>
        <RUN_PARAMETERS_DEF>
            <PARAM_CATEGORY_DEF NAME="Parameters">

                <PARAMETER_DEF ID="idName" NAME="Name" TYPE="STRING" UNIT="">
                    <ELEM ID="Default_Value" TYPE="STRING" STR="Res%d"/>
                </PARAMETER_DEF>

                <PARAMETER_DEF ID="R" NAME="Resistance" TYPE="VALUE" UNIT="Ohm">
                    <ELEM ID="Default_Value" TYPE="VALUE" VALUE="1.0"/>
                </PARAMETER_DEF>
            </PARAM_CATEGORY_DEF>
        </RUN_PARAMETERS_DEF>
    </SCHEMBOX_DEF>
</SYSML>
```

Figure 6.1-1 Structure of the .sysapp file

6.1.1 Global definition

The first part of the sysapp file gives the name and the type (Electrical, Thermal or Logical) of the component. It also makes the link with the .jpg and .powcmp corresponding files.

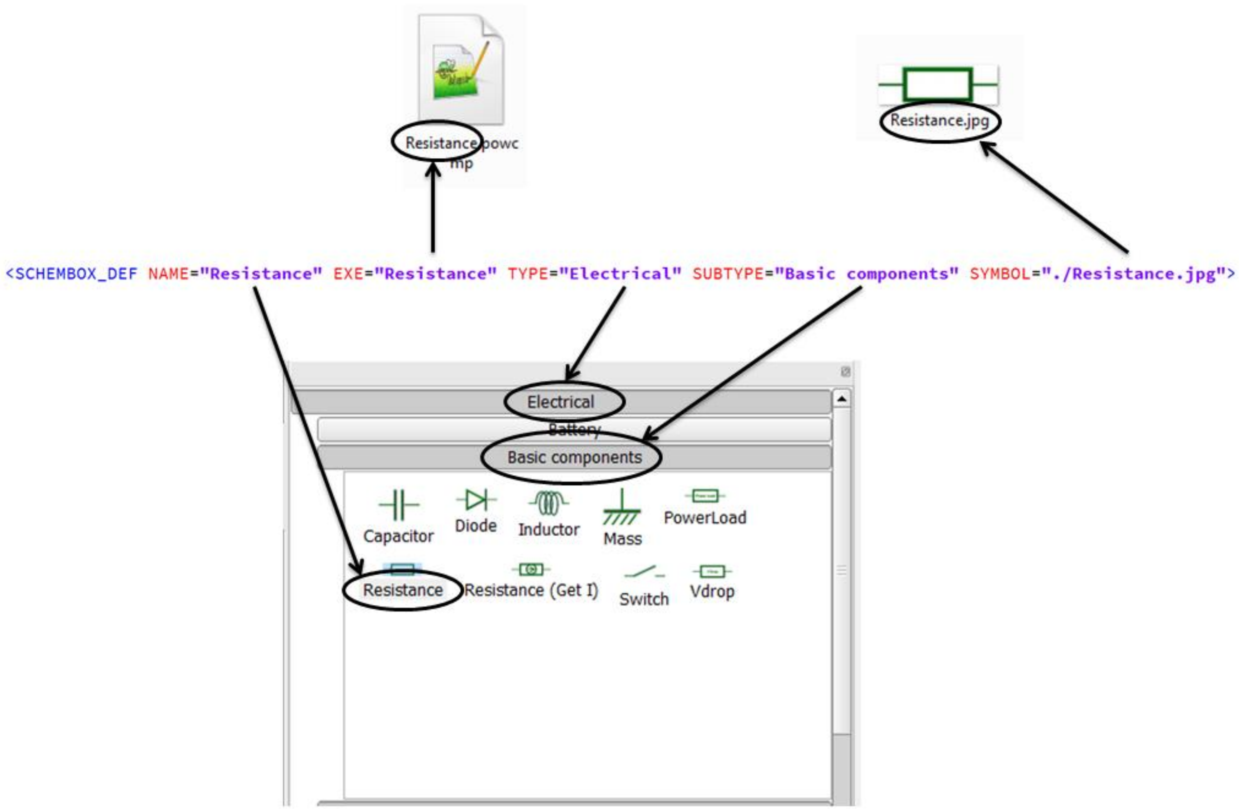


Figure 6.1-2 Example of a global definition of a component

The model of the component (powcmp file) is associated to the graphical part of the component (sysapp file) through the “EXE” parameter. This parameter should be equal to the first part of the corresponding powcmp file name (without the “.powcmp” suffix).

6.1.2 Connectors definition

To define a new connector for the component in the sysapp file, the following information are needed:

	Description	***Possible values***
TYPE	type of the connector. It should be electrical, thermal or logical	ELEC THERMIC LOGIC

INOUT *(only for a logical connector)*	Specify if it is an input or an output connector	INPUT OUTPUT
SIDE	Define the location of the connector in the component drawing	LEFT RIGHT TOP BOTTOM
ROLE_TITLE	Name of the connector which is displayed in the GUI	(should correspond to the connector name in the powcmp file)
ROLE_DESC	Description of the connector	-

Here is an example of the connectors definition for the Solar Array component:

```
<CONNECTOR_DEF TYPE="THERMIC" SIDE="LEFT" ROLE_TITLE="Tfront" ROLE_DESC="Front Thermal Node"/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="OUTPUT" SIDE="RIGHT" ROLE_TITLE="Vmax" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="INPUT" SIDE="RIGHT" ROLE_TITLE="Ishunt" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="INPUT" SIDE="RIGHT" ROLE_TITLE="Nsect" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="ELEC" SIDE="BOTTOM" ROLE_TITLE="Uin" ROLE_DESC="Uin"/>
<CONNECTOR_DEF TYPE="ELEC" SIDE="TOP" ROLE_TITLE="Uout" ROLE_DESC="Uout"/>
```

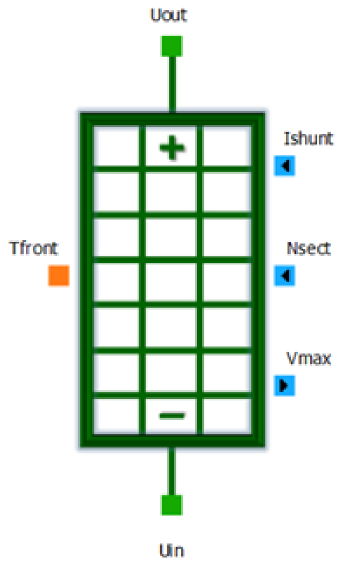


Figure 6.1-3 Connectors definition for the Solar Array component

6.1.3 Input Parameters definition

The sysapp file should contain all the input parameters that will be used in the component equations (powcmp file), except the current, voltage and temperature that are automatically defined. In addition, the first parameter shall always be the name of the component instance.

To define a new component input parameter in the sysapp file, the following information are needed:

	Description	***Possible values/Comments***
ID	Parameter name that will be used in the powcmp file	The ID must absolutely be unique in a syapp file. The ID of the other component parameters should be different.
NAME	Description of the parameter that is displayed in the GUI	The NAME must absolutely be unique in a syapp file. The NAME of the other component parameters should be different.
TYPE	Type of the parameter in the GUI (different from the type of the parameter in the powcmp file)	STRING VALUE (The type can be a STRING in the GUI whereas it is a REAL or INTEGER in the powcmp file)
UNIT	Unit of the parameter that is displayed in the GUI	-

It is possible to define a default value for each parameter. The type of this default value should be the same as the type of the parameter (STRING or VALUE). Here are two examples of parameters with a default value:

```xml
<RUN_PARAMETERS_DEF>
    <PARAM_CATEGORY_DEF NAME="Parameters">

        <PARAMETER_DEF ID="idName" NAME="Name" TYPE="STRING" UNIT="">
            <ELEM ID="Default_Value" TYPE="STRING" STR="Res"/>
        </PARAMETER_DEF>

        <PARAMETER_DEF ID="R" NAME="Resistance" TYPE="VALUE" UNIT="Ohm">
            <ELEM ID="Default_Value" TYPE="VALUE" VALUE="1.0"/>
        </PARAMETER_DEF>
    </PARAM_CATEGORY_DEF>
</RUN_PARAMETERS_DEF>
```

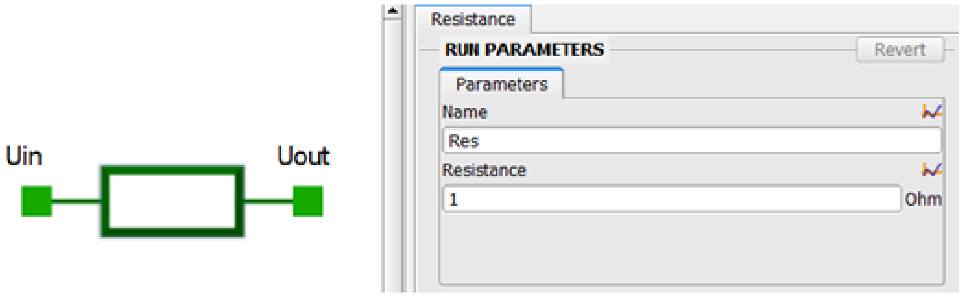


Figure 6.1-4 Parameters definition with default value for the Resistance

Some constraints can be added if the type “VALUE” is used for the component parameter. In that case, the GUI will prohibit values that do not respect the defined constraints. The example below gives the syntax for adding constraints. Here, the resistance value should be strictly greater than 0 Ohm and less than or equal to 50 Ohm:

```xml
<RUN_PARAMETERS_DEF>
    <PARAM_CATEGORY_DEF NAME="Parameters">

        <PARAMETER_DEF ID="idName" NAME="Name" TYPE="STRING" UNIT="">
            <ELEM ID="Default_Value" TYPE="STRING" STR="Res%d"/>
        </PARAMETER_DEF>

        <PARAMETER_DEF ID="R" NAME="Resistance" TYPE="VALUE" UNIT="Ohm">
            <ELEM ID="Default_Value" TYPE="VALUE" VALUE="1.0"/>
            <CONSTRAINT TYPE="RANGE">
                <ELEM ID="Min_Value" TYPE="VALUE" VALUE="0"/>
                <ELEM ID="Equal_Min" TYPE="STRING" STR="FALSE"/>
                <ELEM ID="Max_Value" TYPE="VALUE" VALUE="50"/>
                <ELEM ID="Equal_Max" TYPE="STRING" STR="TRUE"/>
            </CONSTRAINT>
        </PARAMETER_DEF>
    </PARAM_CATEGORY_DEF>
</RUN_PARAMETERS_DEF>
```

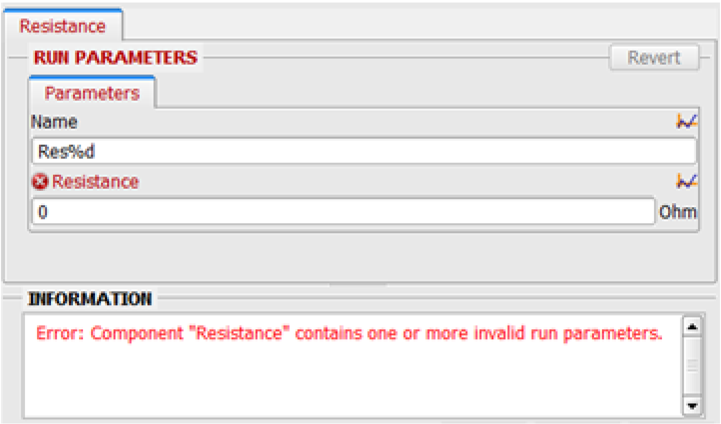


Figure 6.1-5 Parameters definition with constraints for the Resistance

Note that it is also possible to define only a minimum value or only a maximum value.

6.1.4 Parameters using a txt input file

If the parameter is supposed to take values that are time-dependent for example, it is possible to add an option “Use a txt file” to choose if a txt file should be load for this parameter or not.

Constant value for the load power consumption

Parametric value for the load power consumption

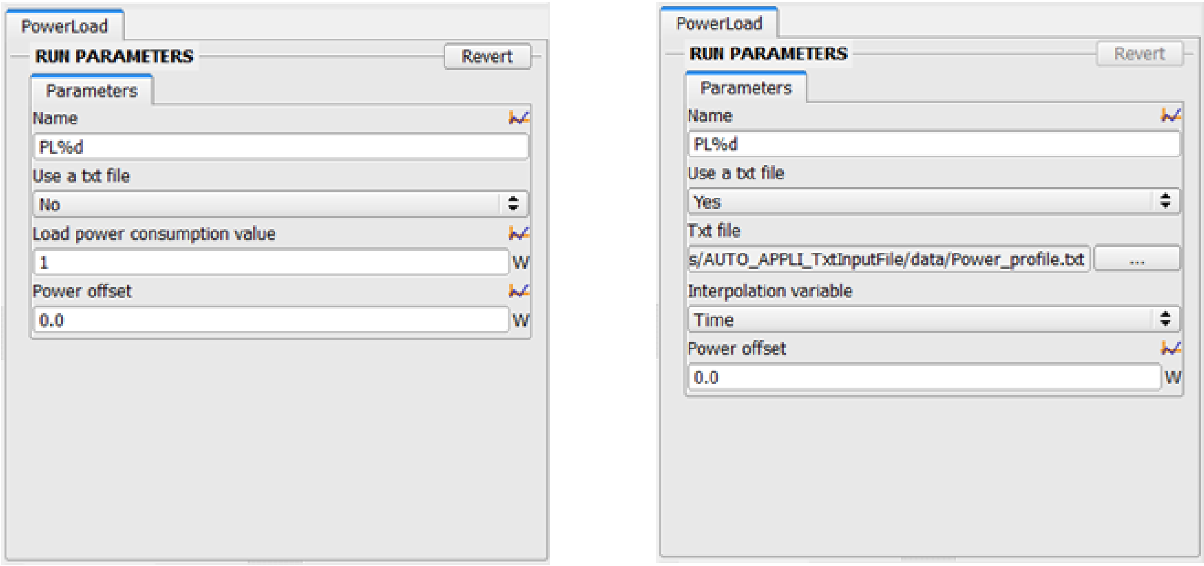


Figure 6.1-6 Create an option to add a txt input file

If the user selects “No” for the option “Use a txt file”, he can specify a constant value for the parameter.

If the user selects “Yes”, he can give a txt file containing a table of the time dependent values that the parameter can take. During the simulation, the solver will compute the load power consumption value at each time by interpolating in the table thanks to the INTRP1 Thermisol routine. The txt file should have 2 columns separated by a tab or a space, the first one is the interpolation variable (the time), the second one is the component parameter value.

A very specific syntax is needed in the sysapp file to add the option “Use a txt file” for a component parameter. In particular, three additional parameters (with specific names) should be defined in the sysapp file (we supposed here that the component parameter is named “ParamName”):

Additional Parameter Name	***Description***	***Possible values/Comments***
ParamName_UseInputFile	Selector to choose if a txt input file is needed	Yes No
ParamName_InputFile	Browser to select the path of the txt input file	This field appears only if the selector is put to “Yes”
ParamName_Interpolator	Selector to choose the interpolation variable	For now, only the time is available. This field appears only if the selector is put to “Yes”

ParamName	Field to specify a constant value for the component parameter	This field appears only if the selector is put to “No”

An example can be found in the PowerLoad.sysapp file for the parameter “P”:

```
<PARAMETER_DEF ID="P_UseInputFile" NAME="Use a txt file" TYPE="STRING" UNIT="">
    <CONSTRAINT TYPE="CHOICE">
        <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
        <ELEM NAME="No"  TYPE="STRING" STR="No"/>
    </CONSTRAINT>
    <ELEM ID="Default_Value" TYPE="STRING" STR="No"/>
</PARAMETER_DEF>

<PARAMETER_DEF ID= "P_InputFile" NAME="Txt file" TYPE="FILE_DESCRIPTION">
  <EXISTENCE>
    <CONDITION PARAMID="P_UseInputFile">
      <CONSTRAINT TYPE="CHOICE">
        <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
      </CONSTRAINT>
    </CONDITION>
  </EXISTENCE>
</PARAMETER_DEF>

<PARAMETER_DEF ID="P_Interpolator" NAME="Interpolation variable" TYPE="STRING" UNIT="">
    <CONSTRAINT TYPE="CHOICE">
        <ELEM NAME="Time" TYPE="STRING" STR="Time"/>
    </CONSTRAINT>
    <EXISTENCE>
        <CONDITION PARAMID="P_UseInputFile">
            <CONSTRAINT TYPE="CHOICE">
                <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
            </CONSTRAINT>
        </CONDITION>
    </EXISTENCE>
    <ELEM ID="Default_Value" TYPE="STRING" STR="Time"/>
</PARAMETER_DEF>


<PARAMETER_DEF ID="P" NAME="Load power consumption value" TYPE="STRING" UNIT="W">
    <ELEM ID="Default_Value" TYPE="STRING" STR="1"/>
   <EXISTENCE>
     <CONDITION PARAMID="P_UseInputFile">
       <CONSTRAINT TYPE="CHOICE">
         <ELEM NAME="No" TYPE="STRING" STR="No"/>
       </CONSTRAINT>
     </CONDITION>
   </EXISTENCE>
</PARAMETER_DEF>
```

Figure 6.1-7 Parameters using a txt input file

To add the option “Use a txt file” in a new component, the user can copy and paste the following code in his sysapp file and only customize the red text:

```
<PARAMETER_DEF ID="ParamName_UseInputFile" NAME="Description (1)" TYPE="STRING" UNIT="">
     <CONSTRAINT TYPE="CHOICE">
          <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
          <ELEM NAME="No"  TYPE="STRING" STR="No"/>
     </CONSTRAINT>
     <ELEM ID="Default_Value" TYPE="STRING" STR="No"/>
</PARAMETER_DEF>

<PARAMETER_DEF ID= "ParamName_InputFile" NAME="Description (2)" TYPE="FILE_DESCRIPTION">
      <EXISTENCE>
           <CONDITION PARAMID=" ParamName _UseInputFile">
                <CONSTRAINT TYPE="CHOICE">
                     <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
                </CONSTRAINT>
           </CONDITION>
      </EXISTENCE>
   </PARAMETER_DEF>

<PARAMETER_DEF ID="ParamName_Interpolator" NAME="Description (3)" TYPE="STRING" UNIT="">
     <CONSTRAINT TYPE="CHOICE">
          <ELEM NAME="Time" TYPE="STRING" STR="Time"/>
     </CONSTRAINT>
     <EXISTENCE>
          <CONDITION PARAMID="ParamName_UseInputFile">
               <CONSTRAINT TYPE="CHOICE">
                    <ELEM NAME="Yes" TYPE="STRING" STR="Yes"/>
               </CONSTRAINT>
          </CONDITION>
     </EXISTENCE>
     <ELEM ID="Default_Value" TYPE="STRING" STR="Time"/>
</PARAMETER_DEF>

<PARAMETER_DEF ID="ParamName" NAME="Description (4)" TYPE="STRING" UNIT="ParamUnit">
     <ELEM ID="Default_Value" TYPE="STRING" STR="DefaultValue"/>
     <EXISTENCE>
           <CONDITION PARAMID="ParamName_UseInputFile">
               <CONSTRAINT TYPE="CHOICE">
                     <ELEM NAME="No" TYPE="STRING" STR="No"/>
                </CONSTRAINT>
           </CONDITION>
     </EXISTENCE>
</PARAMETER_DEF>
```

Note that description 1 to 4 must all be different.

6.1.5 Limitations

The sysapp files should be edited with caution to avoid compatibility issues. In particular, it is not recommended to re-open an old schematic after modifying a sysapp file of a component that is used in this schematic. In that case, it is strongly advised to rebuild the schematic from scratch in order to avoid compatibility issues.

Please also note that Systema should be restarted in order to take into account any change in a sysapp file.

6.2 Powcmp file

The structure of a powcmp file is based on two specific sections:

- The component description
- The component equations

6.2.1 Component description

The description of the component is used to declare the name of the electrical and thermal connectors, the variables names, types and default values. Those declarations are made using the following Thermisol-like blocks:

$U Connectors

$T Connectors

$Variables

$Locals

The language used is MORTRAN.

Note: The block names are not case sensitive.
Comment may be written using the # character.
The logical connectors are declared in the $Variables block.

The electrical and thermal connectors are simply declared with their names, one connector by line as shown in the following example.

Example of $U Connectors paragraph

```
$U Connectors
Uin      # Input connector of the component
Uout     # Output connector of the component
```

Note that the connector names in the powcmp file should correspond to the connector names in the sysapp file (ROLE_TITLE). The logical connectors should be declared in the $VARIABLES block.

[sysapp file]

```
<CONNECTOR_DEF TYPE="THERMIC" SIDE="LEFT" ROLE_TITLE="Tfront" ROLE_DESC="Front Thermal Node"/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="OUTPUT" SIDE="RIGHT" ROLE_TITLE="Vmax" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="INPUT" SIDE="RIGHT" ROLE_TITLE="Ishunt" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="INPUT" SIDE="RIGHT" ROLE_TITLE="Nsect" ROLE_DESC=""/>
<CONNECTOR_DEF TYPE="ELEC" SIDE="BOTTOM" ROLE_TITLE="Uin" ROLE_DESC="Uin"/>
<CONNECTOR_DEF TYPE="ELEC" SIDE="TOP" ROLE_TITLE="Uout" ROLE_DESC="Uout"/>
```

[powcmp file]

```
$U Connectors
  Uin     # Negative terminal
  Uout    # Positive terminal
$T Connectors
  Tfront  # SA front face temperature
```

Figure 6.2-1 Component connectors - Connection between the sysapp and powcmp files

It is important to notice that electrical connector names shall always start by the letter **U** and thermal connectors by the letter **T**.
In the component's code, it will be possible to access to the voltage of a node by using the electrical connector's name and also to the electrical current crossing this connector using that same name but a letter **I** instead of the first **U** (for example Iout is the current going to the Uout connector).

Reminder: a positive current is leaving the component and a negative one is entering it.

For thermal node, the temperature is directly accessed by the connector's name. The dissipation can also be accessed by using the letter **Q** instead of the first **T** in the name. Note that the dissipation of a thermal node is actually linked with the **QI** sub-category of thermal node powers.

A connector may be optional, meaning that it may be used or not at component's instantiation.
To declare an optional connector the keyword [FACULTATIVE] shall be added after the connector's name declaration, like in the following example.

Example of an optional connector

```
$T Connectors
Tdis [FACULTATIVE]   # Optional Thermal node to output dissipation
```

Whenever an optional connector is used, it is important to check its existence in the component's code using the keyword Exist_Name:

Example of an optional connector existence check

```
$EQ:End
{
   /* Output dissipation */
   If (Exist_Tdis)
   {
      Qdis = (Uin-Uout) * Iout;
   }
}
```

Variables may be declared in a $VARIABLES or a $LOCALS block depending on if they are accessible from the model or not. $LOCALS variables may be used to initiate variables values from the INIT code so to reuse them in the SOLVE code without the need to re-valuate them every time the solve code is called. Those local variables will however not be visible from the model.

The variables can be of the two types REAL (which is in fact a double precision floating value, the same as Thermisol) or INTEGER. A default value shall always be given at declaration.

Example of component variables declaration

```
$Variables
REAL    R      = 1.0   # Resistance value
INTEGER Status = 0     # Switch status
```

It is also possible to indicate that a variable is dynamic, i.e. time dependent, for specific transient cases. This will be described in the next paragraph.

All the component input parameters declared in the corresponding sysapp file should be declared in the $VARIABLES block (except the name of the component and the additional parameters used for the "Use a txt file" option). These are the input variables. The variable name declared in the powcmp file should correspond to the parameter name (=ID) declared in the sysapp file. The value of the input variables is specified by the user through the GUI.

Variables declared in the $VARIABLES block of the powcmp file but not in the sysapp file are the output parameters of the component. Their value should be calculated in the component's code.

[sysapp file]

```
<RUN_PARAMETERS_DEF>
    <PARAM_CATEGORY_DEF NAME="Parameters">

        <PARAMETER_DEF ID="idName" NAME="Name" TYPE="STRING" UNIT="">
            <ELEM ID="Default_Value" TYPE="STRING" STR="Res%d"/>
        </PARAMETER_DEF>

        <PARAMETER_DEF ID="R" NAME="Resistance" TYPE="STRING" UNIT="Ohm">
            <ELEM ID="Default_Value" TYPE="STRING" STR="1.0"/>
        </PARAMETER_DEF>
    </PARAM_CATEGORY_DEF>
</RUN_PARAMETERS_DEF>
```

[powcmp file]

```
$VARIABLES

 # Input Parameters
  REAL R        =    1.0       # Resistance value (Ohm) (should be > 0)

 # Output Parameters
  REAL V        =     0.0      # Resistance voltage (V)
  REAL Pdis     =     0.0      # Resistance dissipation (W)
```

Figure 6.2-2 Component parameters - Connection between sysapp and powcmp files

The logical (input and output) connectors defined in the sysapp file should also be defined in the $VARIABLES block. Their name should correspond.

[sysapp file]

```
<CONNECTOR_DEF TYPE="ELEC" SIDE="LEFT" ROLE_TITLE="Uin" ROLE_DESC="Input voltage generally connected to the Solar Array"/>
<CONNECTOR_DEF TYPE="ELEC" SIDE="RIGHT" ROLE_TITLE="Uout" ROLE_DESC="Output voltage generally connected to the Bus"/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="OUTPUT" SIDE="LEFT" ROLE_TITLE="Ishunt" ROLE_DESC="Current to be shunted"/>
<CONNECTOR_DEF TYPE="LOGIC" INOUT="INPUT" SIDE="RIGHT" ROLE_TITLE="MEA" ROLE_DESC="Error voltage piloting signal for the regulator"/>
```

[powcmp file]

```
# Input Interface
REAL(DYN)    MEA       = 0.0      # Error voltage piloting signal for the regulator (-)

# Output Interface
REAL         Ishunt    = 0.0      # Current to be shunted (A)
```

Figure 6.2-3 Logical connectors - Connection between sysapp and powcmp file

Note that the value of all the variables declared in the $VARIABLES block (input or output variables) is exported at each computation step in the temp.h5 generated by the solver. The values of all these variables will be accessible at the end of the calculation. However, the value of the variables declared in the $LOCALS paragraph are not exported.

6.2.2 Component equations

The second part of the powcmp file contains the component equations. Unlike the input model, the component equations are coded in C and not in MORTRAN. This is because the solver kernel is actually coded in C.

The component code is structure around 3 main parts:

$EQ:Init

$EQ:Solve

$EQ:End

The Init block is used as a pre-execution of a resolution (steady-state or transient time-step) and the End block as a post-execution.

Usually, the Init code is used to value variables that do not depend on the component's voltages and currents but on thermal environment and/or time.

The End block is generally used to post-process the electrical convergence, such as setting thermal dissipations or integrating time-dependent parameters.

The main purpose of the component code is to return the value of the current I at each connector and the jacobian matrix dI/dU. The jacobian terms can be accessed through the variables derived from the connectors' names like in the following example:

Example of jacobian instantiation

```
$EQ:Solve
{
[…]
double jacobian = -1. / R;
dIin_dUin = dIout_dUout = jacobian;
dIin_dUout = dIout_dUin = -jacobian;
}
```

In some cases, it is possible that a component shall not compute the current on a connector but needs to get the incoming or outgoing current. The function getIin() for a connector called Uin may then be used. This function will automatically compute the external current sum on the connected node and will affect its opposite value to the connector's current.

6.2.3 Transient components and Dynamic parameters

Some components have transient behaviors and need to use the value of a variable at the previous time step.

During a transient resolution (through the routines TRANS_U or TRANS_UT), the previous values of the electrical connectors voltages and currents are automatically saved. It is then possible to access those data through their names followed by the suffix _prev.

It is also possible to access to variable previous values by declaring the variables (external or locals) as dynamic variables. This is done by adding (DYN) after the variable type.

It is important to notice that a steady-state convergence is always performed to initiate the electrical network at the simulation starting time. During this call, the transient mode is not yet set and previous values have no means. As a consequence, any code using previous values of U, I or any dynamic parameter shall be written under a test condition if(Transient). An alternative code may be written for specific steady-state purpose.

In addition of previous values, it is possible to access time related data in transient mode. Those data are:

- `curTime`: current time of the computation
- `prevTime`: previous time computed
- `dTime`: current time-step value (equal to `curTime-prevTime`)

Here is the code of a capacitance as an example of transient components. The steady-state behavior coded here is converged state of the capacitance (fully charged or discharged according to its voltage gradient).

Example of transient component

```
$U Connectors
Uin
Uout

$VARIABLES
REAL Capa = 1.0

$EQ:Solve
{
   double jacobien;
   double V = Uin - Uout;
   double Vprev = Uin_prev - Uout_prev;
   if (Transient)
   {
      Iout = Capa * (V - Vprev) / dTime;
      Iin = -Iout;
      jacobien = Capa / dTime;
   }
   else
   {
```

```
      Iout = 0.0;
      Iin  = 0.0;
      jacobien = 0.0;
   }
   dIin_dUin   = -jacobien;
   dIout_dUout = -jacobien;
   dIin_dUout  =  jacobien;
   dIout_dUin  =  jacobien;
}
```

6.2.4 Components used as functions

Dealing with components offers a new way of managing an input model compared to the classical Thermisol usage. It is then possible to create components for linking parameters without being a true component (i.e. connecting electrical nodes).

Adding an integrator to a model could then be made through an integrator component such as:

Example of component used as a function

```
$Variables
REAL(DYN) X         # input signal to be integrated
REAL(DYN) integX    # integration of signal

$EQ:Solve
{
   if (Transient)
   {
      integX = integX_prev + (X + X_prev) * dTime / 2. ;
   }
   else
   {
      integX = 0.0 ;
   }
}
```

7 Power post-processing

7.1 H5 output file

The Thermisol solver generates an output file temp.h5 in HDF5 format. This file contains general data (program version, user name, start date and time of the simulation…), simulation data (nodes identification and models identification, number of couplings), model properties (areas, capacitance, couplings…), power and thermal results (Voltage, current, component parameters, temperature, external fluxes…) and analysis data (minmax analyses, flux budgets…). The frequency at which the results are stored in the h5 file can be modified thanks to the H5_FREQ variable. For more general information concerning the h5 output file, please refer to the Thermisol User Manual.

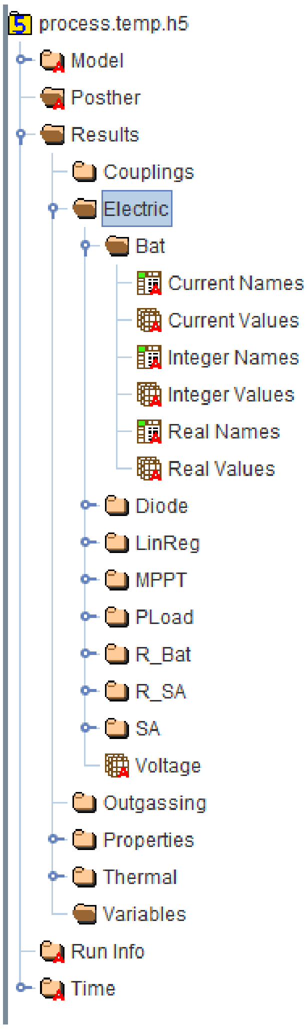
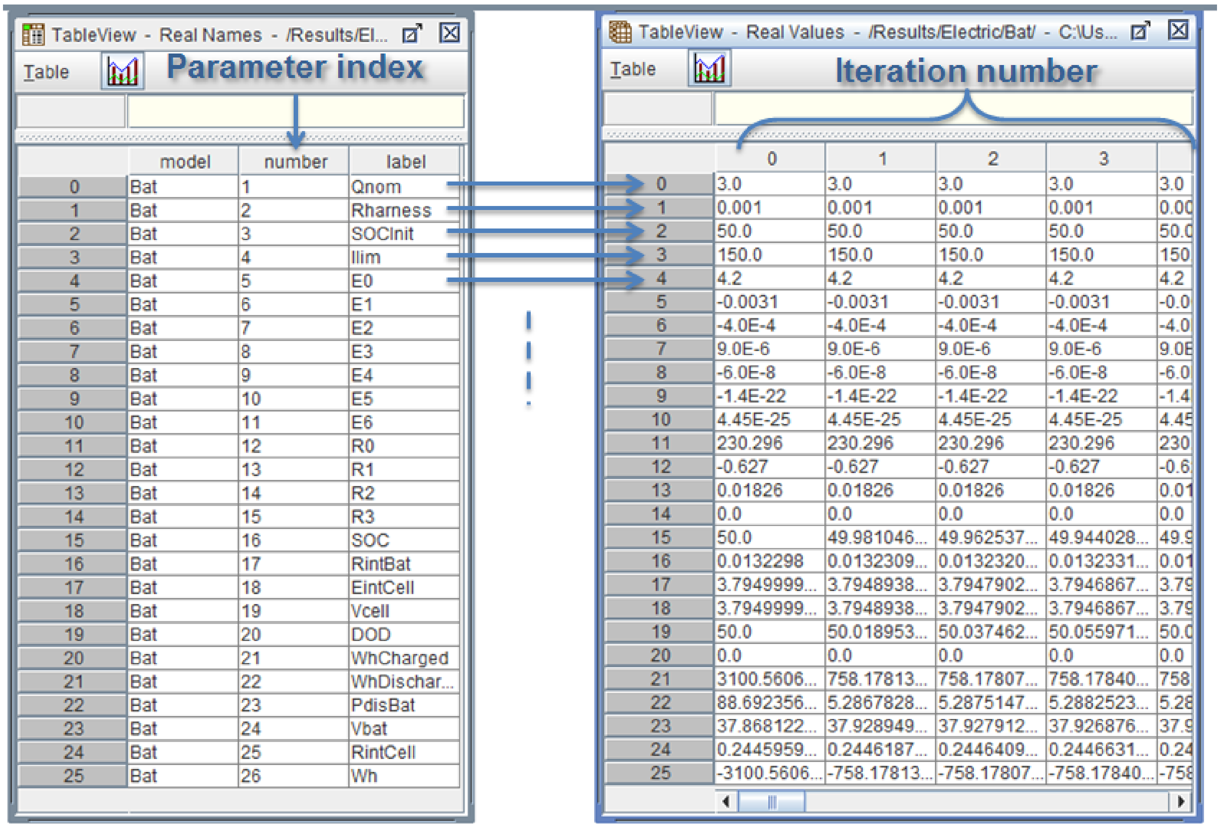


Figure 7.1-1 Example of an H5 file structure

Concerning the power part, the h5 file contains one group per component instance used in the electrical architecture. Each group can contain up to 6 datasets:

- The “Current Names” dataset gives the labels of the currents leaving and entering the component.
- The “Current Values” dataset gives the values of the currents leaving and entering the component
- The “Integer Names” dataset gives the labels of all the component parameters (input and output) that are defined as “INTEGER” in the powcmp file.
- The “Integer Values” dataset gives the values of all the component parameters (input and output) that are defined as “INTEGER” in the powcmp file.
- The “Real Names” dataset gives the labels of all the component parameters (input and output) that are defined as “REAL” in the powcmp file.
- The “Real Values” dataset gives the values of all the component parameters (input and output) that are defined as “REAL” in the powcmp file.

The couple of datasets “Names” and “Values” are organized in the same order in such a way that it is possible to associate the values with the labels by crossing the tables.

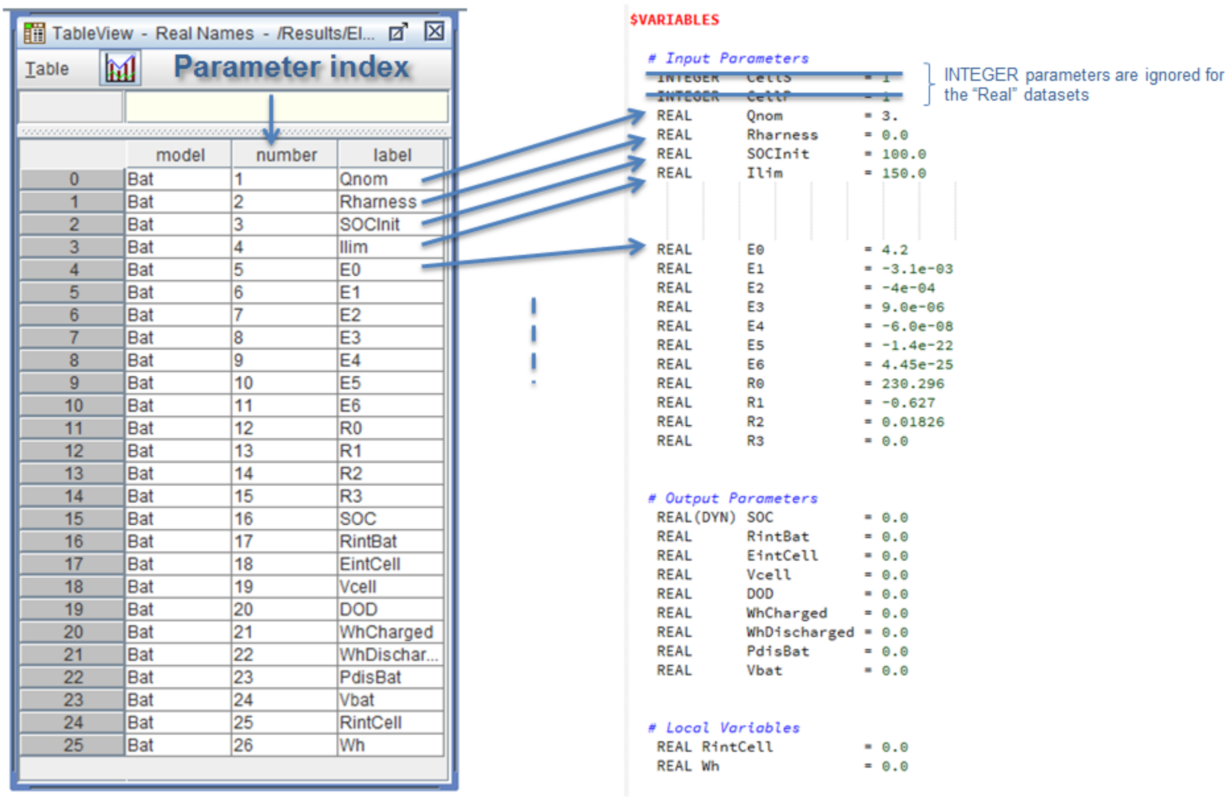


	model	number	label
0	Bat	1	Qnom
1	Bat	2	Rharness
2	Bat	3	SOCInit
3	Bat	4	Ilim
4	Bat	5	E0
5	Bat	6	E1
6	Bat	7	E2
7	Bat	8	E3
8	Bat	9	E4
9	Bat	10	E5
10	Bat	11	E6
11	Bat	12	R0
12	Bat	13	R1
13	Bat	14	R2
14	Bat	15	R3
15	Bat	16	SOC
16	Bat	17	RintBat
17	Bat	18	EintCell
18	Bat	19	Vcell
19	Bat	20	DOD
20	Bat	21	WhCharged
21	Bat	22	WhDischar...
22	Bat	23	PdisBat
23	Bat	24	Vbat
24	Bat	25	RintCell
25	Bat	26	Wh

	0	1	2	3	
0	3.0	3.0	3.0	3.0	3.0
1	0.001	0.001	0.001	0.001	0.00
2	50.0	50.0	50.0	50.0	50.0
3	150.0	150.0	150.0	150.0	150
4	4.2	4.2	4.2	4.2	4.2
5	-0.0031	-0.0031	-0.0031	-0.0031	-0.0
6	-4.0E-4	-4.0E-4	-4.0E-4	-4.0E-4	-4.0
7	9.0E-6	9.0E-6	9.0E-6	9.0E-6	9.0E
8	-6.0E-8	-6.0E-8	-6.0E-8	-6.0E-8	-6.0
9	-1.4E-22	-1.4E-22	-1.4E-22	-1.4E-22	-1.4
10	4.45E-25	4.45E-25	4.45E-25	4.45E-25	4.45
11	230.296	230.296	230.296	230.296	230
12	-0.627	-0.627	-0.627	-0.627	-0.6
13	0.01826	0.01826	0.01826	0.01826	0.01
14	0.0	0.0	0.0	0.0	0.0
15	50.0	49.981046...	49.962537...	49.944028...	49.9
16	0.0132298	0.0132309...	0.0132320...	0.0132331...	0.01
17	3.7949999...	3.7948938...	3.7947902...	3.7946867...	3.79
18	3.7949999...	3.7948938...	3.7947902...	3.7946867...	3.79
19	50.0	50.018953...	50.037462...	50.055971...	50.0
20	0.0	0.0	0.0	0.0	0.0
21	3100.5606...	758.17813...	758.17807...	758.17840...	758
22	88.692356...	5.2867828...	5.2875147...	5.2882523...	5.28
23	37.868122...	37.928949...	37.927912...	37.926876...	37.9
24	0.2445959...	0.2446187...	0.2446409...	0.2446631...	0.24
25	-3100.5606...	-758.17813...	-758.17807...	-758.17840...	-758

Figure 7.1-2 Correspondence between "Names" and "Values" datasets

In each dataset, the component parameters are stored in the order in which they are declared in the corresponding powcmp file. For convenience, section 7.3 “List of parameter indexes” gives the index of each component parameter for the standard component library.



Real Names dataset for the battery

Powcmp file of the battery

Figure 7.1-3 Correspondence between datasets and powcmp file

The h5 file contains also a dataset named “Voltage”. This dataset gives the electric potential values of each electrical node. This dataset is organized in the same order as the dataset “UNodes” (in the Model group). It is possible to associate the electric potential values to the node numbers by crossing the tables.

It is possible to post-process the results contains in the h5 file by using the “post-processing” tab of Systema (min/max computation, comparison, difference or sum, extract in CSV H5 or XLSX, Power budget, etc …). For the general usage of the post-processing tab, please refer to the Systema User Manual (sections “Post-processing management” and “Post-processing technical annex”).

7.2 Display curves in Systema

It is possible to visualize results contained in the h5 file thanks to the Graph View accessible from the Modeler tab using the View/Add/Graph View option of the menu bar or clicking on the button. The graph can be configured by opening the configuration window (in the right click menu, select configuration window or use the button).

The General tab of the configuration window allows you to select the electrical or thermal nodes (through the node number) or components parameters (through the parameter index) for which you want to display results.

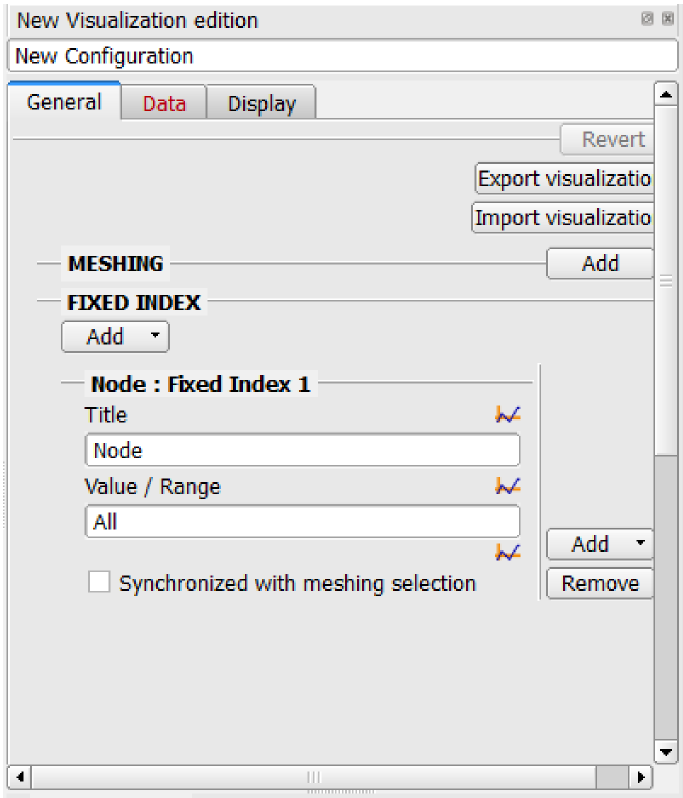


Figure 7.2-1 General tab of the configuration window

The selection should be written in the “Value/Range” field. To specify a component parameter, the syntax is as follows:

```
<ComponentName>:<ParameterIndex>
```

Examples of definition of node or parameter index:

- one node: 100
- one parameter: Bat:16 (corresponds to the SOC of the battery named “Bat”)
- more than one node: 100,2
- more than one parameter: Bat:2, Bat:5, Res:1

- nodes between two values: 100-300
- parameters of the same component between two index values: Bat:2-5
- multi-definition: 2,Res:1,100-300,Bat:2-5
- All nodes and parameters (of all components): All

To find out the index corresponding to each parameter of a component, please refer to section 7.3 “List of parameter indexes”.

The Data tab of the configuration window allows you to select the h5 file(s) containing the data to be observed and the entity that you want to display.

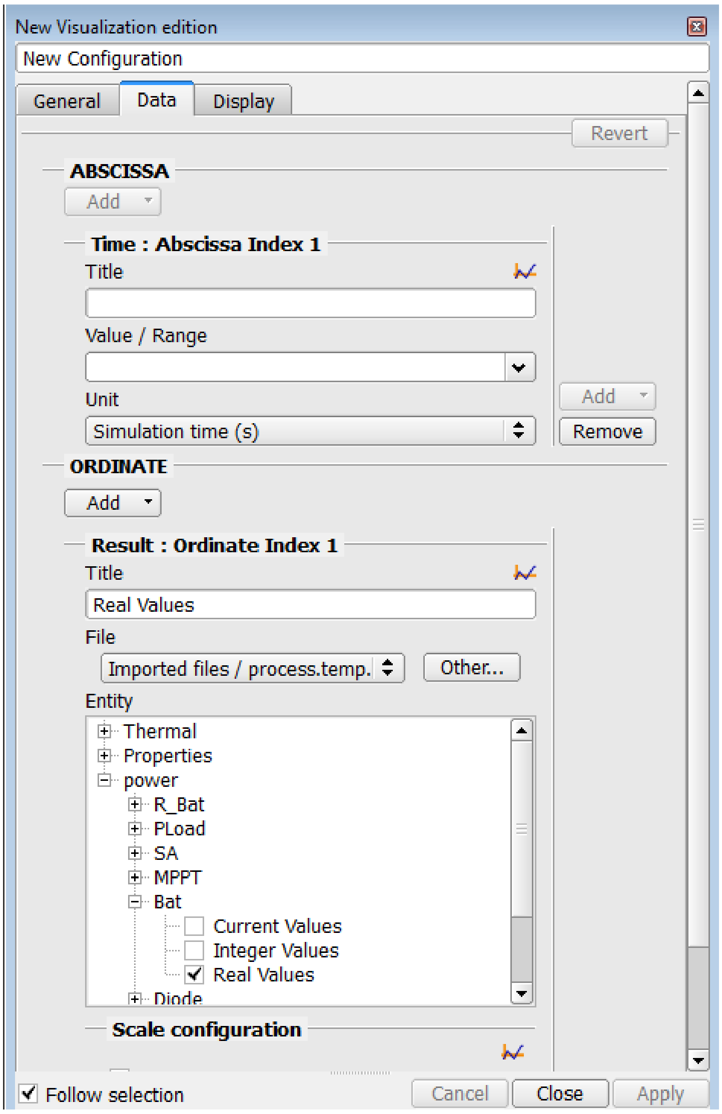


Figure 7.2-2 Data tab of the configuration window

The entities correspond to the datasets of the h5 file. So you have to select:

- Current Values to display the current leaving and entering a component

- Integer Values to display the component parameters defined as INTEGER in the powcmp file of the component
- Real Values to display the component parameters defined as REAL in the powcmp file of the component
- Electric Potentials to display the electric potential values of an electrical node

Note that it is not possible to select several entities per “Ordinate Index”. To select another entity, click on Add/result at the bottom right of the window.

Section 1.4 gives an example of visualization of the results.
For the general usage of the Graph View, please refer to the Systema User Manual.

7.3 List of parameter indexes

This section gives the list of the parameter indexes for the standard component library. These parameter indexes shall be used to refer to a component parameter in the Graph View and the Post-processing tab.

The syntax to call a component parameter is the following:

```
<ComponentName>:<ParameterIndex>
```

For example, let’s suppose that an electrical diagram contains a Solar Array component instance named “SA”. Then SA:2 is used to refer to :

- The current leaving the component (Iout) if the “Current Values” dataset is selected
- The number of cells in series (CellS) if the “Integer Values” dataset is selected
- The Area correction factor (AreaCorrection) if the “Real Values” dataset is selected

7.3.1 Parameter indexes of the Standard Solar Array

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Nsect	SA sections connected to the bus. (Logical connector)
2	CellS	Number of cells in series
3	CellP	Number of cells in parallel
4	NsectTot	Total number of SA sections

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Ishunt	Shunted current that is used for thermal calculation (Logical connector)
2	AreaCorrection	Area correction factor - 1: Only the active part of the panel (covered by sun cells) is taken into account. Fluxes that impact inactive parts of the panel are lost. Cell's temperature will then be under-estimated - 0: Tfront is the average temperature of the entire panel. Cell's temperature will then be over-estimated because of local thermal gradient due to the cell's absorption of power.
3	APanel	Panel Area
4	Flux	Solar flux associated to the Tfront thermal node
5	KThresh	Thershold (Direct solar flux / Reference flux) below the one there is no output current
6	Losses	Loss factor (between 0 and 1, 1 means 0% loss)
7	ACell	Cell size
8	TRef	Reference temperature
9	FluxRef	Reference flux
10	Isc0	Reference short circuit current density
11	dIsc0	Coefficients of thermal dependence at reference temperature and infinite radiation dose
12	Voc0	Reference open circuit voltage
13	dVoc0	Coefficients of thermal dependence at 0 and infinite radiation dose
14	Vmp0	Reference maximum power point Voltage
15	dVmp0	Coefficients of thermal dependence at 0 and infinite radiation dose
16	Pmp0	Power at the maximum power point
17	dPmp0	Coefficients of thermal dependence at 0 and infinite radiation dose
18	DiodeI0	Parameter for inverse diode modelling
19	DiodeAlpha	Parameter for inverse diode modelling
20	Vmax	SA maximum power point voltage
21	Imax	SA maximum power point current
22	Pmax	SA maximum power point
23	Isc	SA short circuit current
24	Voc	SA open circuit voltage
25	Pout	SA delivered power

26	SA_Matching_Coef	SA matching factor
27	Pmp	SA maximum power point for Flux=FluxRef
28	Vmp	SA maximum power point voltage for Flux=FluxRef
29	Imp	SA maximum power point current for Flux=FluxRef

7.3.2 Parameter indexes of the Standard Battery

- **Current Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

Parameter index	*Parameter Name*	*Description*
1	CellS	Number of cells in series
2	CellP	Number of cells in parallel

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Qnom	Cell nominal capacity
2	Rharness	Total resistance of the battery internal wiring
3	SOCInit	Initial state of charge value wrt remaining capacity
4	Ilim	This is used in case there are 2 solutions of (V,I) leading to a given output power (so the lowest I is taken as solution)
5	E0	Internal voltage polynomial coefficient
6	E1	Internal voltage polynomial coefficient
7	E2	Internal voltage polynomial coefficient
8	E3	Internal voltage polynomial coefficient
9	E4	Internal voltage polynomial coefficient
10	E5	Internal voltage polynomial coefficient
11	E6	Internal voltage polynomial coefficient
12	R0	Internal resistor polynomial coeffcient
13	R1	Internal resistor polynomial coeffcient
14	R2	Internal resistor polynomial coeffcient

15	R3	Internal resistor polynomial coeffcient
16	SOC	State of charge wrt the remaining capacity
17	RintBat	Battery internal resistance
18	EintCell	Internal cell voltage
19	Vcell	Cell voltage
20	DOD	Depth of charge
21	WhCharged	Battery charged energy
22	WhDischarged	Battery discharged energy
23	PdisBat	Instantaneous dissipation of battery
24	Vbat	Battery voltage
25	RintCell	Cell internal resistance
26	Wh	Battery instantaneous energy

7.3.3 Parameter indexes of the Shunt Regulator

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No Integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	MEA	Error voltage piloting signal for the regulator
2	Ishunt	Current to be shunted
3	Vd	Voltage drop of the diode
4	Rs	Series resistance
5	Rshunt	Shunt resistance
6	Pd	Instantaneous power dissipation
7	Pin	Input power
8	Pout	Output power
9	Eff	Instantaneous energy efficiency
10	EffAve	Average energy efficiency

7.3.4 Parameter indexes of the MPPT

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Isa	Current entering the component
2	Ibus	Current leaving the component

- **Integer Values dataset**

Parameter index	***Parameter Name***	***Description***
1	FlagMea	This flag is set so to switch to a power regulation within a time-step if the MEA becomes positive

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Vcell	Voltage to be regulated by MPPT
2	Vmax	Maximum power point voltage
3	MEA	Error voltage piloting signal for the regulator
4	Vinit	Initialization of Vcell parameter
5	effMPPT	Efficiency of MPPT - if effMPPT > 0: fixed constant efficiency - else: use efficiency coefficients
6	a1	Efficiency coefficient (only used if effMPPT≤ 0)
7	a2	Efficiency coefficient (only used if effMPPT≤ 0)
8	a3	Efficiency coefficient (only used if effMPPT≤ 0)
9	b1	Efficiency coefficient (only used if effMPPT≤ 0)
10	b2	Efficiency coefficient (only used if effMPPT≤ 0)
11	b3	Efficiency coefficient (only used if effMPPT≤ 0)
12	c1	Efficiency coefficient (only used if effMPPT≤ 0)
13	c2	Efficiency coefficient (only used if effMPPT≤ 0)
14	c3	Efficiency coefficient (only used if effMPPT≤ 0)
15	maxPout	Maximum output power
16	minVsa	Minimum input voltage of MPPT
17	maxVsa	Maximum input voltage of MPPT
18	Kmea	Internal regulation constant
19	Kinteg	Coefficient applied to the integral - if Kinteg < 0: shunt type regulation - else: series type regulation
20	Pin	Input power from Solar Array
21	Pout	Output power to bus

22	Efficiency	Instantaneous efficiency of MPPT
23	Dissipation	Instantaneous dissipation of MPPT
24	aveEff	Average energy efficiency
25	MEAint	Internal MEA signal
26	Pcte	Intermediate variable used only if effMPPT ≤0
27	K1	Intermediate variable used only if effMPPT ≤0
28	K2	Intermediate variable used only if effMPPT ≤0
29	Dampt	Dampt factor
30	MEAmean	Internal MEA signal

7.3.5 Parameter indexes of the Linear Regulation

- **Current Values dataset**

No current values (logical component).

- **Integer Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Is_taper	Flag for Taper voltage computation

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Vreg	Voltage to be regulated
2	Ireg	Current to be regulated
3	Tbat	Battery temperature
4	MEA	Error voltage piloting signal for the regulator.
5	Ilim	Current limitation value
6	VlimABS	Absolute limitation voltage
7	Vlim0	Voltage limitation at 0°C
8	dVlim	Variation coefficient of limitation voltage depending on temperature with respect to 0°C
9	epsilon	Epsilon for the threshold of the Taper voltage
10	deltaV	Delta V for the taper voltage computation
11	K	Regulation constant
12	TIlim	Current limitation duration
13	Ttaper	Voltage limitation duration
14	Vlim	Regulation voltage

7.3.6 Parameter indexes of the Resistance and the Resistance (Get I)

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	R	Resistance value
2	V	Resistance voltage
3	Pdis	Resistance dissipation

7.3.7 Parameter indexes of the Diode

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Imin	Current leaving the component
2	Imax	Current entering the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Vd	Diode’s conduction voltage threshold
2	Ron	Diode residual resistance
3	V	Diode voltage
4	Pdis	Diode dissipation

7.3.8 Parameter indexes of the Capacitor

- **Current Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Capa	Capacity value
2	V	Capacity voltage

7.3.9 Parameter indexes of the Inductor

- **Current Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	L	Inductance Value
2	V	Inductor voltage

7.3.10 Parameter indexes of the Vdrop

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	In	Input voltage
2	Gain	Gain
3	V	Differential voltage
4	Pdis	Vdrop dissipation

7.3.11 Parameter indexes of the Switch

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Control	Switch control.
2	Ron	Resistance value at ON status

7.3.12 Parameter indexes of the Voltage Source and the Controlled Voltage Source

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Gain	Gain
2	Vs	Source voltage
3	V	Differential voltage

7.3.13 Parameter indexes of the Current Source and the Controlled Current Source

- **Current Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	Gain	Gain
2	Is	Generated current
3	V	Differential voltage

7.3.14 Parameter indexes of the PowerLoad

- **Current Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Iin	Current entering the component
2	Iout	Current leaving the component

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	P	Load power consumption value
2	Offset	Offset applied to the load power consumption value
3	Ptot	Total load power consumption value

7.3.15 Parameter indexes of the Comparator

- **Current Values dataset**

No current values (logicial component).

- **Integer Values dataset**

Parameter index	*Parameter Name*	*Description*
1	Logic	Normal (0) or reverse (1) logic command

- **Real Values dataset**

Parameter index	*Parameter Name*	*Description*
1	In	Input value to compare
2	Out	Output logic value of the comparator
3	MaxThresh	Maximum threshold
4	MinThresh	Minimum threshold

7.3.16 Parameter indexes of the Integrator

- **Current Values dataset**

No current values (logicial component).

- **Integer Values dataset**

No integer parameters.

- **Real Values dataset**

Parameter index	***Parameter Name***	***Description***
1	In	Input value to integrate
2	Out	Output logic value of the comparator
3	Gain	Constant gain of the integrator
4	Initial	Initial value of the integrator output