The main Working Areas for designing in EicasLab™

The Plant Area

Via Vincenzo Vela, 27 10128 Torino - ITALY (IT)
Tel. +39 011 56 23 798 +39 011 56 23 088
Fax +39 011 43 60 679
www.eicas.it

TABLE OF CONTENT

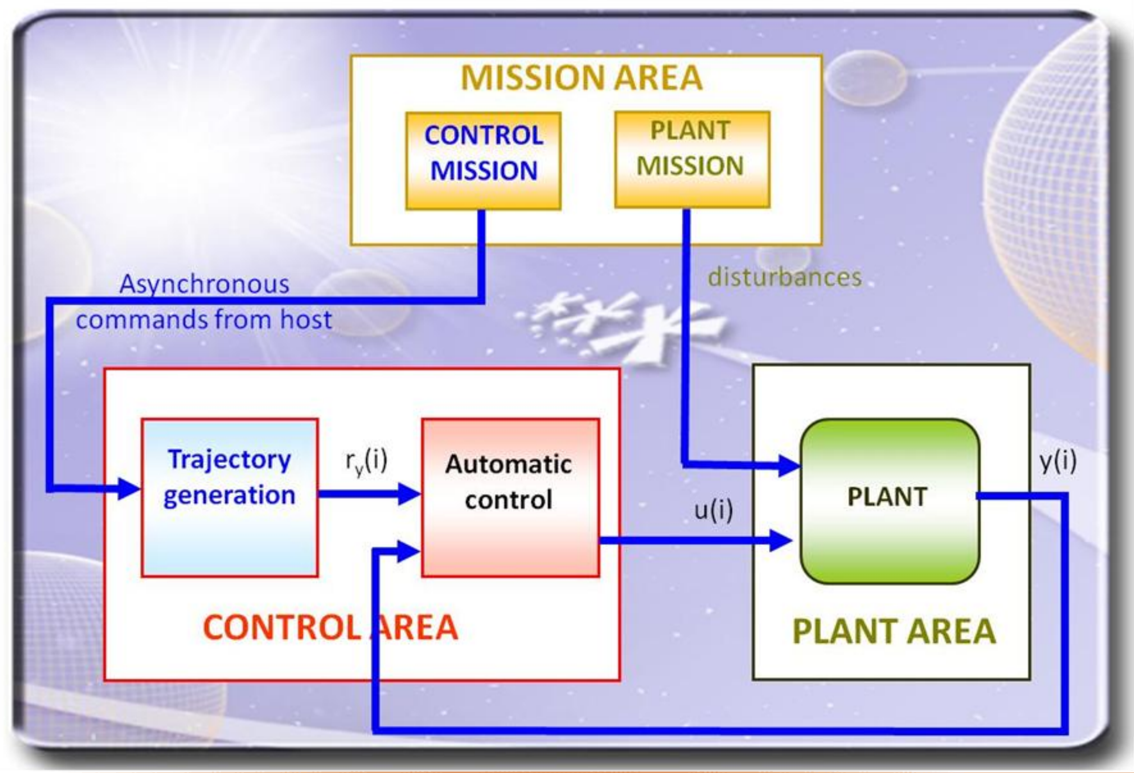
The three main Working Areas

EICASLAB™ has been conceived and developed as a professional software suite supporting the automatic control design and allows to develop and test embedded control system architectures at different hierarchical levels.

Three main Working Areas are available in EICASLAB:

- the **Plant Area**,
- the **Control Area**,
- the **Mission Area**,

specifically devoted and customized to program the different parts of your project.



The Plant Area concept

The Plant modeling is a fundamental task for the control system development.

In the first phase of control design, the control system is tested in a whole simulated environment in which the plant to be controlled is fully simulated.

The Plant Area in EICASLAB is specifically devoted to offer all the necessary features for modeling your Plant.

Typically the Plant model must consider all the aspects neglected in the control algorithm design– such as friction, hysteresis and other non linearities – but that may act on the frequency band of the control.

If the Plant model is not accurate enough you can obtain good simulations but then the control could work not correctly on field.

On the other side, a Plant model too accurate may lead to useless and long computation and simulation time without providing significant results.

The Plant Area environment

EICASLAB offers a pre-organized environment devoted to the design and the implementation of the Plant Area, which allows an accurate and efficient development of your Plant models.
You have at disposal a set of **libraries** devoted to simulate the crucial aspects of the Plant, including general and accurate models for the typical non-linearities such as the **hysteresis** and the **friction** or other non-linearities that typically are neglected in the control algorithm design.

The Plant Area simulation

EICASLAB adopts a fully original and proprietary procedure for the **integration of the differential equations** of the Continuous Plant model.
The procedure has been specifically developed for overcoming the frequent difficulties met as a consequence of the numerical errors, which can not be avoided even with the best techniques of numerical integration. It can be handled in a way such to make their effects negligible.
The procedure requires a "resolution value" for each state variable. The "resolution value" is strictly linked to the physical meaning of the state variable and corresponds to the precision with which you want to compute the variable value at each sampling step.

Plant Categories in the Plant Area

The following categories of plant may be programmed in the Plant Area:

CP

the **Continuous Plant**:
it is the mathematical fine model of the plant to be controlled.
It is a dynamic system - with state and outputs variables - that can be represented through a system of differential equations,

DP

the **Discrete Plant**:
it allows to simulate a Plant by means of a set of finite differences equations (the model uses a discrete time approach),

ED

the **Experimental Data**:
it allows to substitute the Plant model with a set of data collected on field during experimental trials.
It is then possible to perform simulations using directly the on field data instead of data computed by means of a Continuous or a Discrete Plant.

HYB

the **Hybrid Plant**:
it is an advanced container that can collect blocks representing:

- a Continuous Plant, a set of Discrete Plants and Experimental Data,
- missions that allow to model disturbances acting on the Plant,
- A/D and D/A converters.

The Programming modes of the Plant Area

You can develop your Plant:

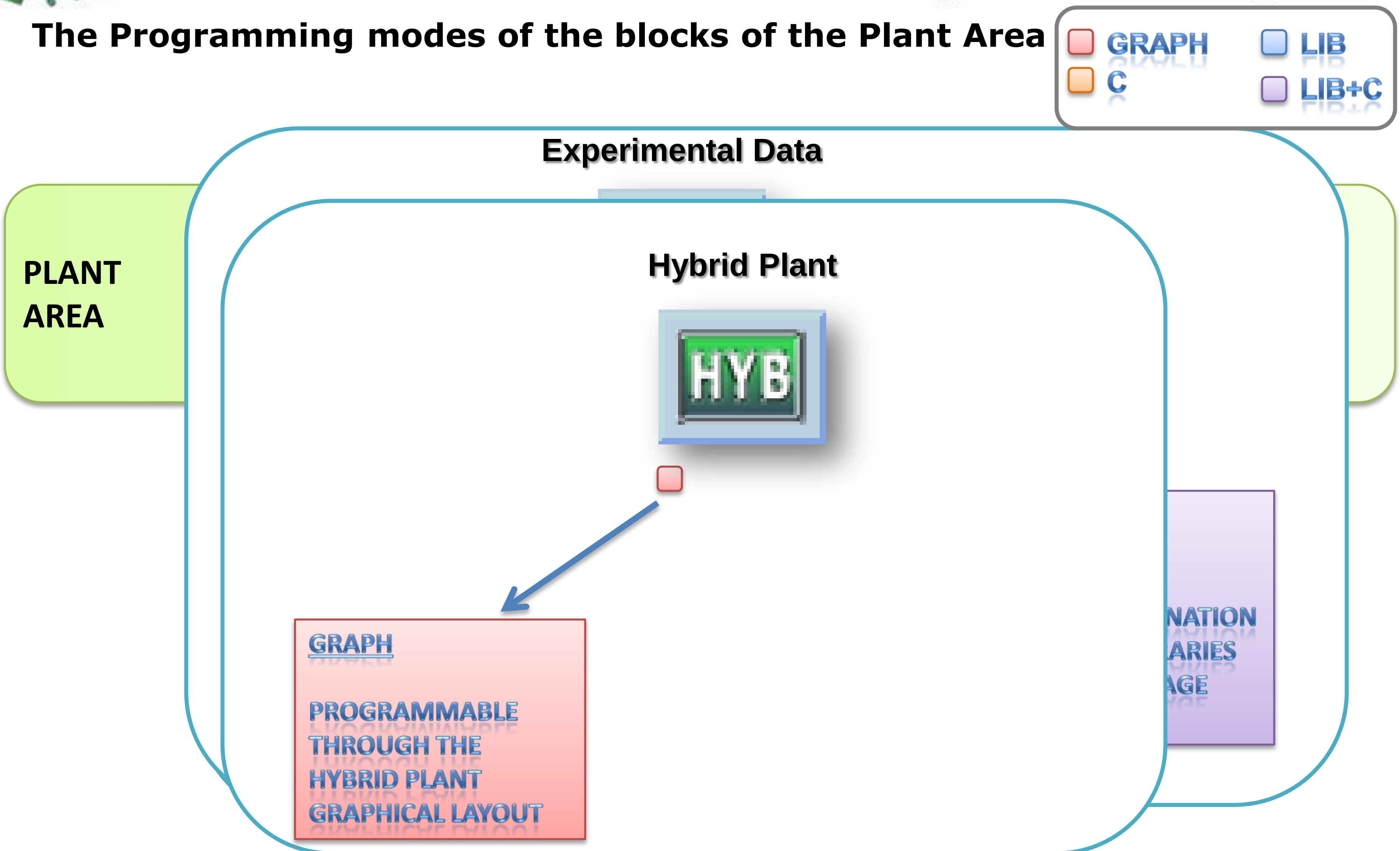
- **graphically** programming:
  you work on **graphical layouts** equipped with specific and oriented **libraries** that contain a set of suitable pre-defined blocks,

- programming with **ANSI C language**:
  EICASLAB allows an easy programming in ANSI C language
  by means of an open and customizable pre-organized structure
  that allows you to focus just on specific and crucial aspects of the system to be programmed.
  You have at disposal a set of template files and libraries,

- using **pre-defined libraries**,

- using a combination of pre-defined libraries and ANSI C language programming.

The Programming modes of the blocks of the Plant Area



CP

The Continuous Plant

Differential state equations and EICASLAB integration procedure

The Continuous Plant is a dynamic system described by a set of **state variables** that can be represented through a system of **differential state equations**.

The differential state equations provide the state derivative as a function of the state and the inputs of the Continuous Plant and are called **state equations**:

$$dx/dt = f(x,u,t;par)$$

(having indicated: x: states, u inputs, t: current time, par: parameters).

The integration of the differential state equations is carried out through a smart proprietary integration procedure embedded in the **EICASLAB SIM** tool.
The procedure has been specifically developed for overcoming the frequent difficulties met as a consequence of the numerical errors, which can not be avoided even with the best techniques of numerical integration. It can be handled in a way such to make their effects negligible.
The procedure requires a “resolution value” for each state variable.

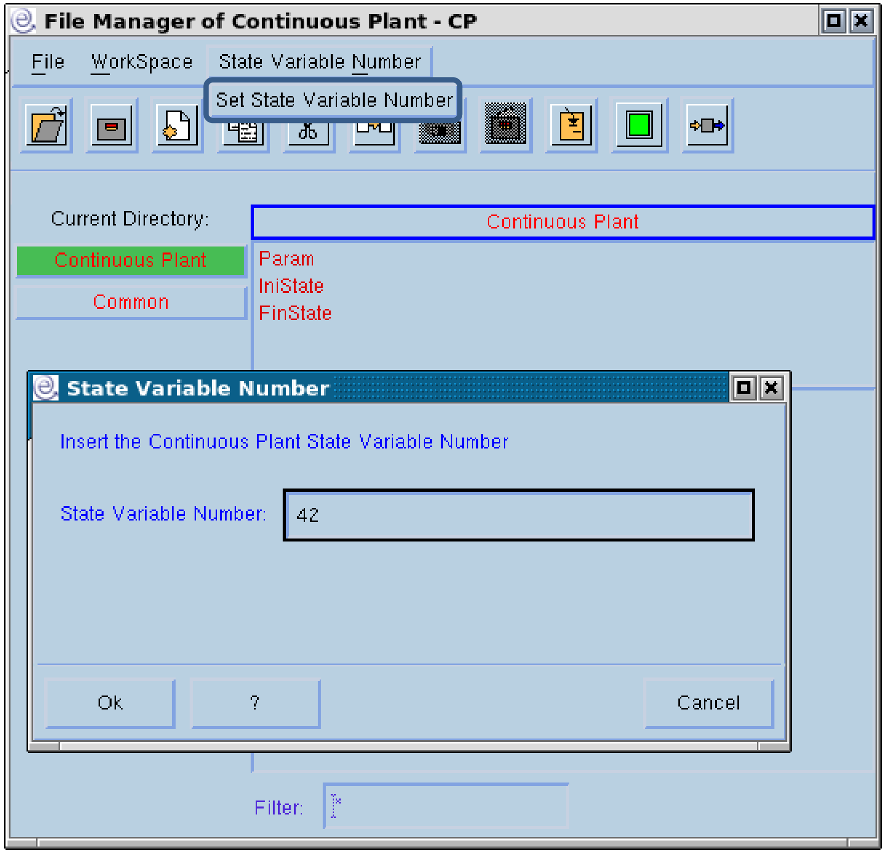
The Continuous Plant

The state variable number

The procedure for integrating the differential equations representing the model of a Continuous Plant requires the knowledge of the **state variable number**:

- **Graphical** Continuous Plant:
  the state variable number is automatically available in EICASLAB based on the graphical representation

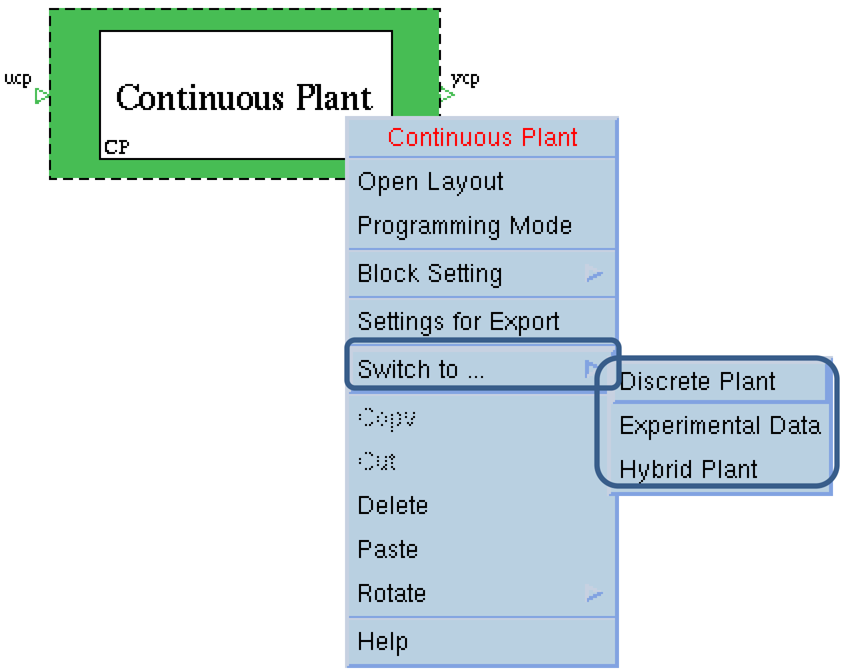
- Continuous Plant programmed in **ANSI C**:
  the user has to explicitly provide the state variable number.

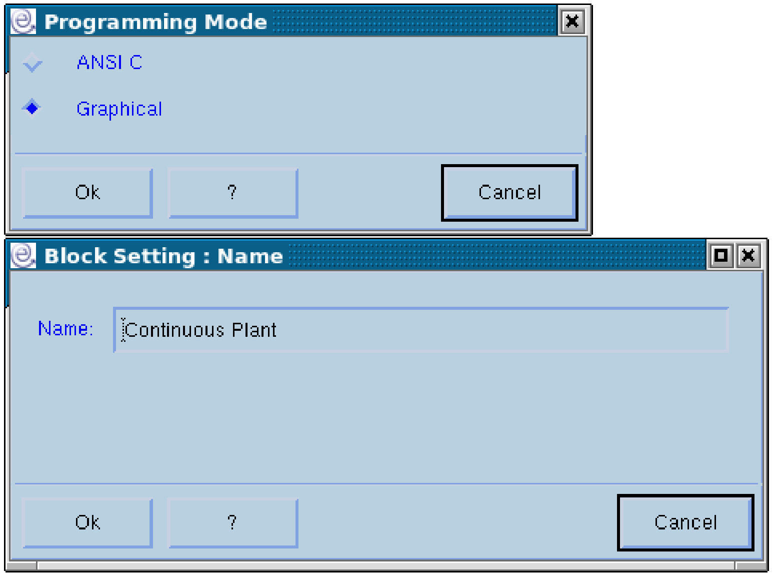


The Continuous Plant

Associated popup menu

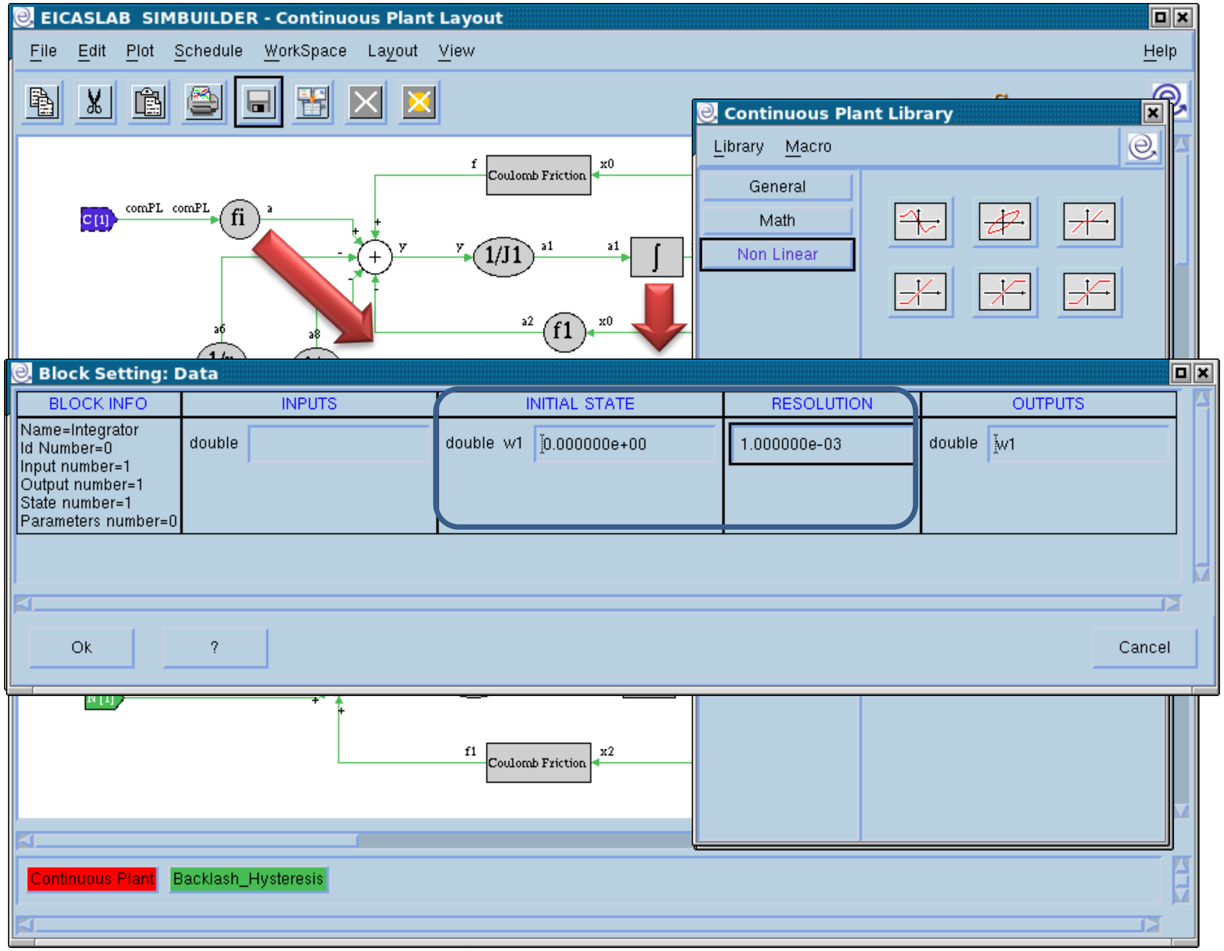
The Continuous Plant is by default graphically programmed.





The Continuous Plant graphically programmed

The Continuous Plant Layout



The Continuous Plant Layout allows to graphically program the Continuous Plant.

You can build your plant model by using the blocks available in the Continuous Plant Library window,

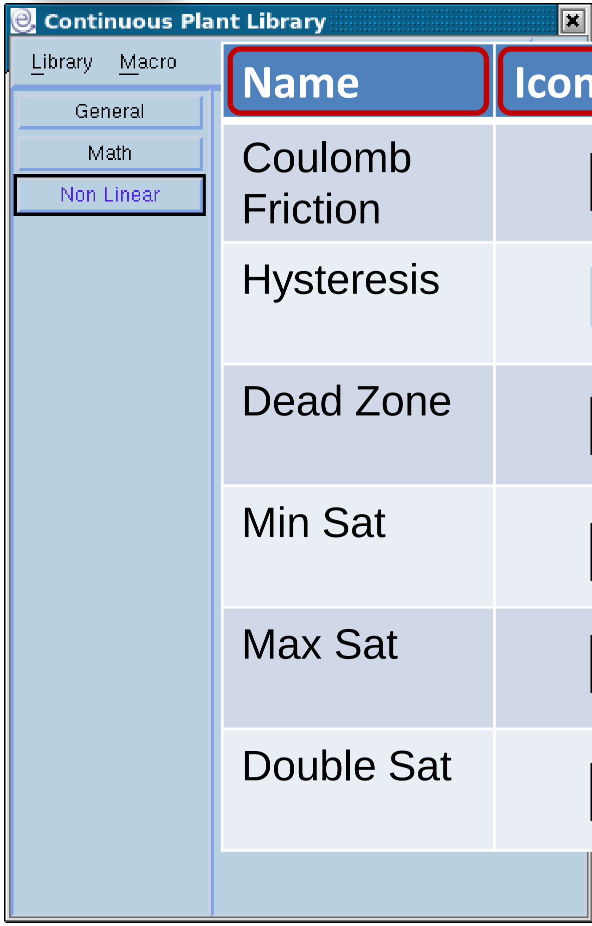
and by setting their:

- outputs,
- parameters,
- resolution (dynamic blocks),
- initial states (dynamic blocks).

CP

The Continuous Plant graphically programmed

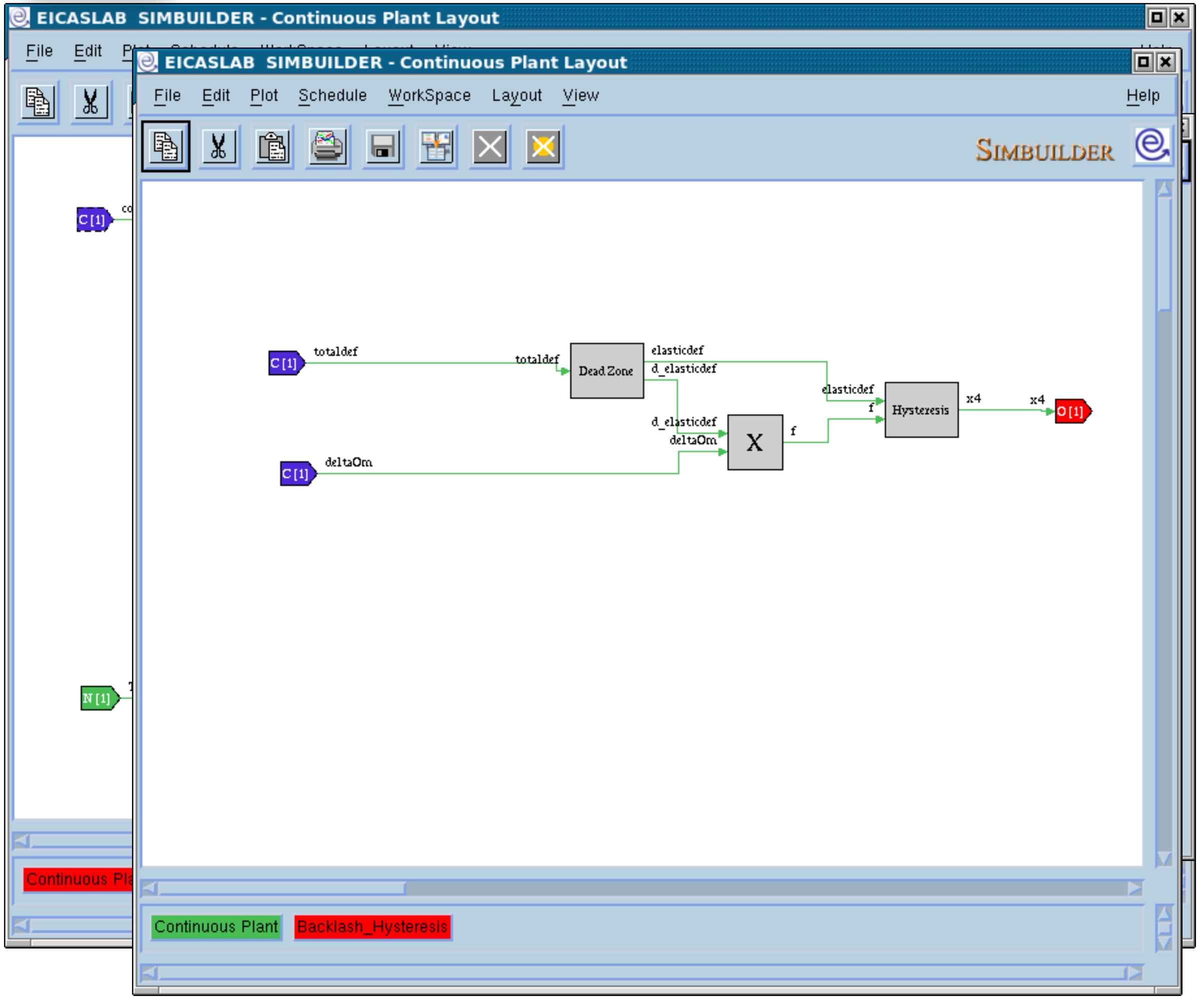
The non-linear library



Name	Icon in library	Block in the layout	decription
Coulomb Friction		Coulomb Friction f1	Generate output according to a coulomb friction model
Hysteresis		Hysteresis f1	Generate output according to an hysteresis model
Dead Zone		Dead Zone f1 f2	Generate output according to a backlash model
Min Sat		Min Sat s1	Limit the lower value of a signal
Max Sat		Max Sat s1	Limit the upper value of a signal
Double Sat		Double Sat s1	Limit the range of a signal

The Continuous Plant graphically programmed

The subsystems

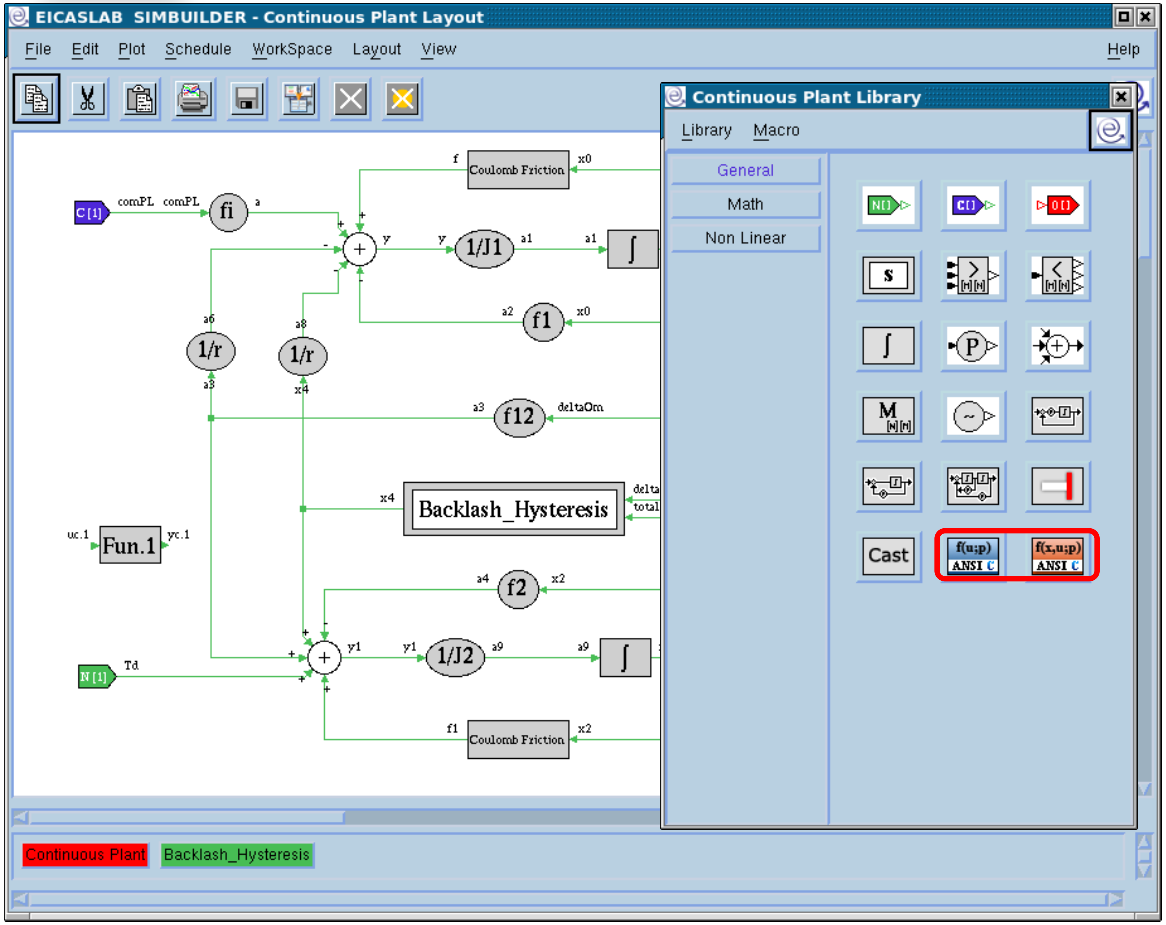


You can simplify the representation of your system by collecting parts of your block diagram in a block called **Subsystem**.

Double clicking on the subsystem opens the *Subsystem* layout, where you can use all the blocks available in the related library.
You can also create other subsystems in order to build a hierarchical block diagram.

The Continuous Plant graphically programmed

The ANSI C blocks



It is possible to use special blocks programmable in ANSI C language.

There are two types of blocks, allowing you to program in ANSI C language:

- static functions
  in this case the C block implements the function:
  y= f(u;par);
- dynamic functions
  in this case the C block implements the function:
  y= f(x,u;par);

(having indicated:
y: outputs, u inputs, x: states, par: parameters)

The Continuous Plant graphically programmed

The macros

The Continuous Plant library window is **customizable** with user blocks called **‘macros’**.

The macros are created by the user in order to complete the library according to the user needs.

The macros can be programmed:

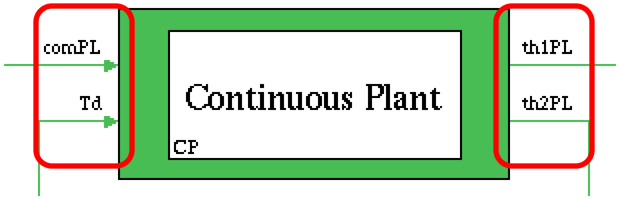
- **graphically** (working on the Graphical Macro layout) or
- **in ANSI C language**.

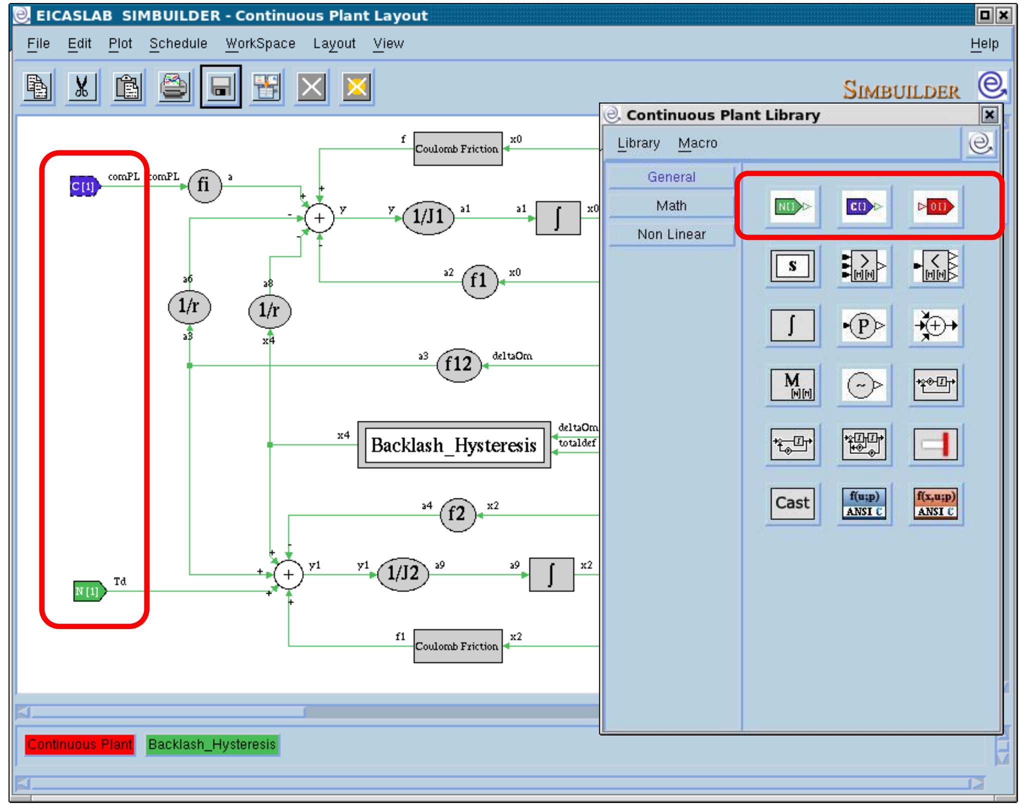
They are then available in the library window of the layout, as all the other blocks and can be used in the current project.

They can also be exported and then used in other projects.

The Continuous Plant graphically programmed

The Input/Output variables





In order to define the inputs and the outputs of a graphically programmed block:

insert
inside the graphical layout the input – outputs blocks.

N[] Plant Noise Input

C[] Plant Command Input

O[] Plant Output

The Continuous Plant programmed with ANSI C language

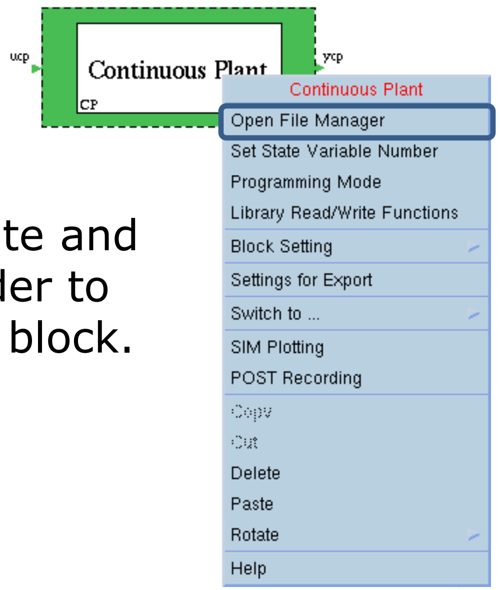
The Continuous Plant file manager

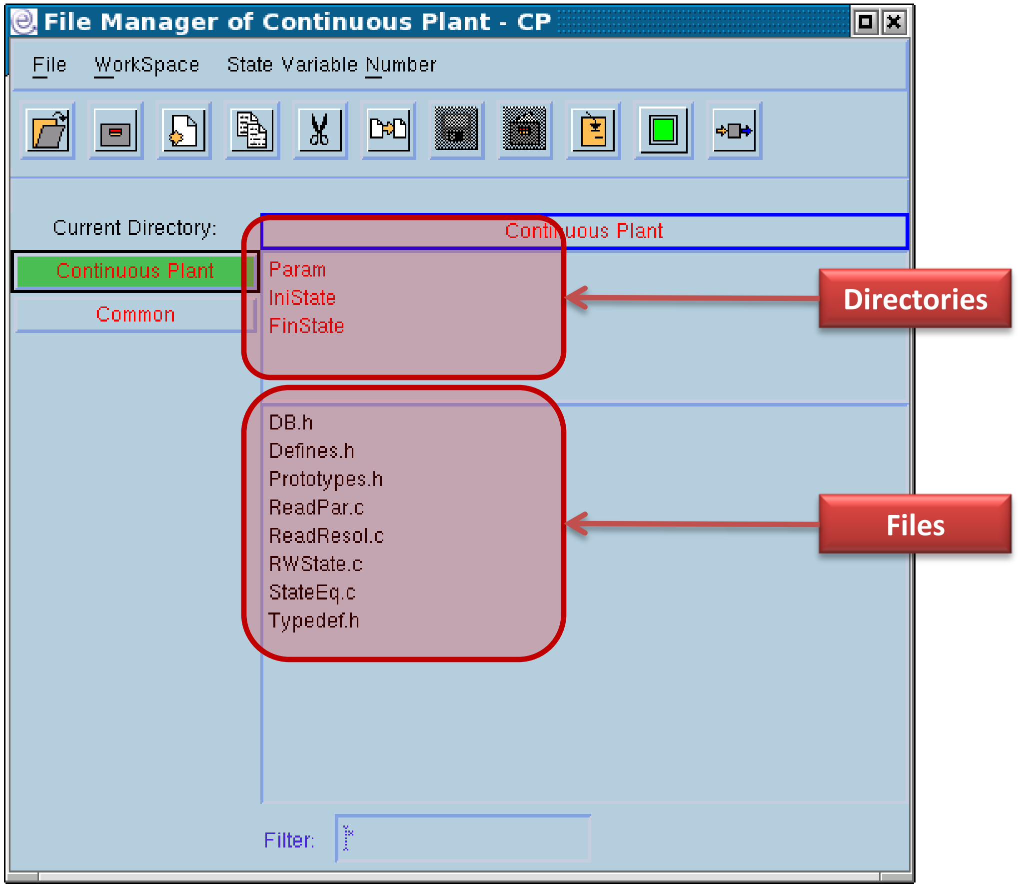
The Continuous Plant programmed with ANSI C language has its own file manager through which it is possible to program the block.

EICASLAB provides a pre-organised structure: a set of template files subdivided in:

- data files,
- header files,
- C files,

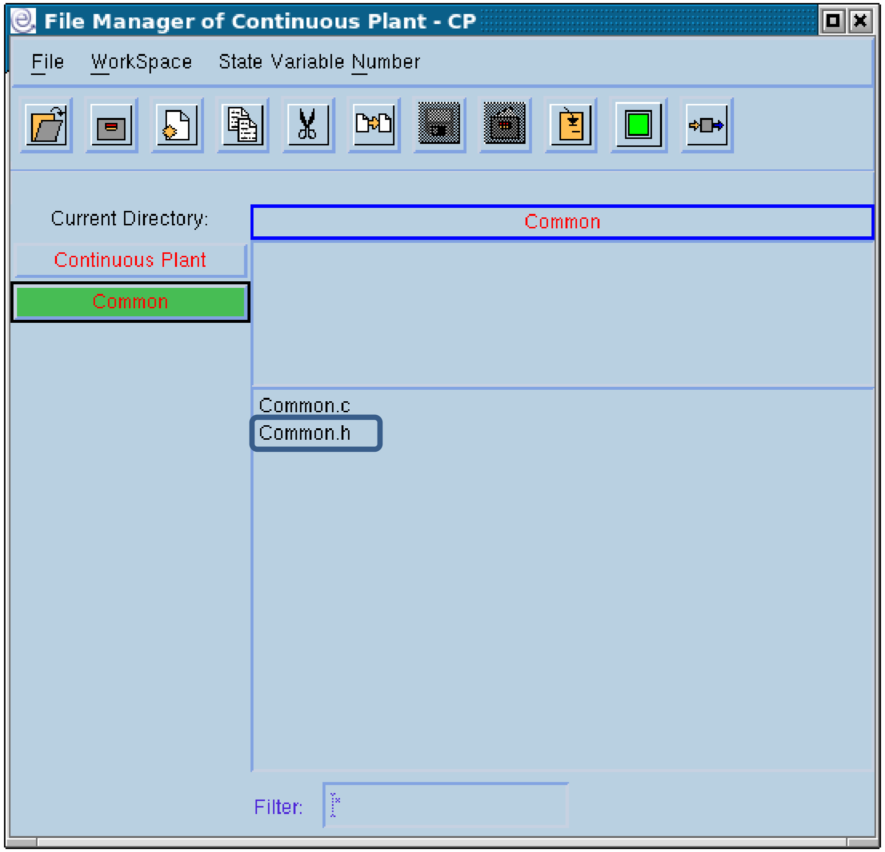
that you can write and customize in order to implement your block.





The Continuous Plant programmed with ANSI C language

The header files



Header files of the pre-organised structure that are written by the user.

Defines.h	Definition of user constants
Typedef.h	Definition of user structures
DB.h	Definition / declaration of user variables
Prototypes.h	Declaration of the function prototypes
Common.h	Available for all the blocks programmed in C

The Continuous Plant programmed with ANSI C language

Initialization functions

Name	Description	C File	Data File
CP_ReadPar	Parameter file reading	ReadPar.c	ContPlant.par
CP_R adState	Initial state le reading	RWS ate.c	ContPl t.inistate
CP_ReadRes	Resolution file reading	ReadResol.c	Resolution.par
CP_Ini	User initialisation function	StateEq.c	---

The Continuous Plant programmed with ANSI C language

Execution functions

Name	Description	C File
CP_StateEq	Computation of the state derivative as a function of the current state and the inputs	StateEq.c
CP_Out	Computation of the outputs of the Continuous Plant as a function of its current state	StateEq.c

The Continuous Plant programmed with ANSI C language

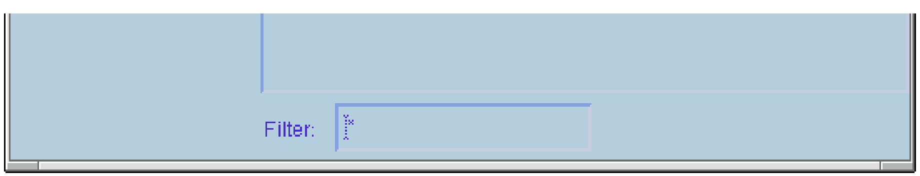
Final functions

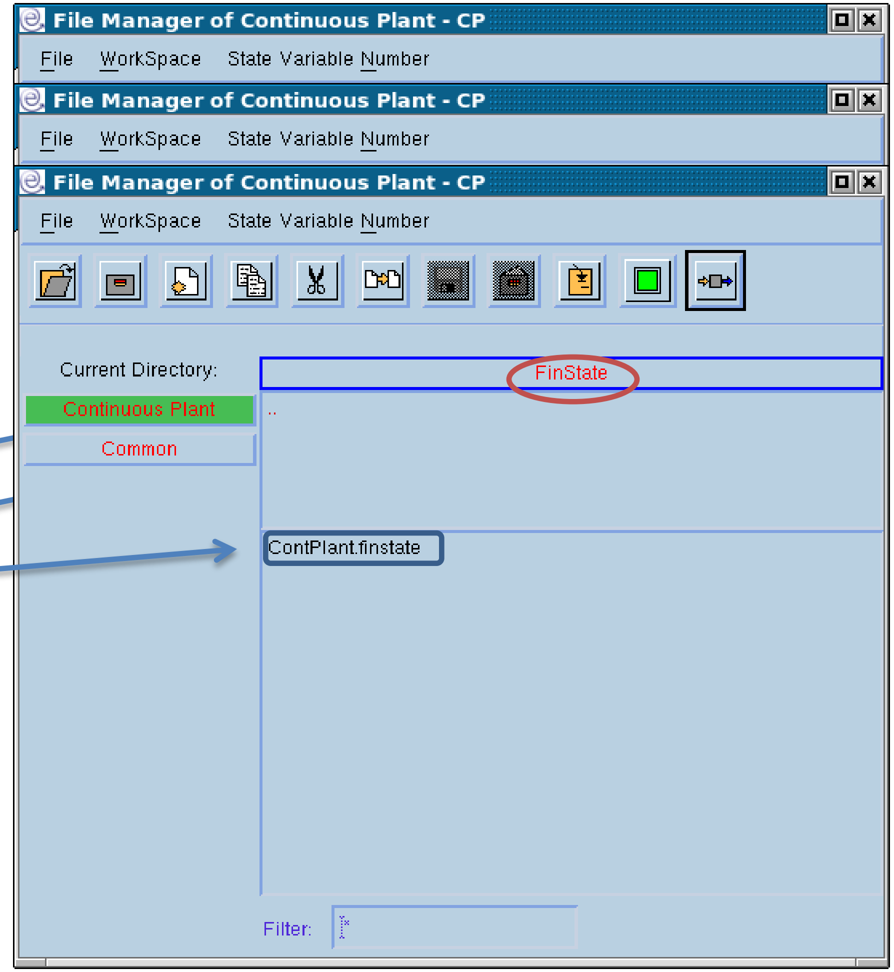
Name	Description	C File	Data File
CP_Fin	User final function	StateEq.c	ContPlant.par
CP_WriteState	Final state file writing	RWState.c	ContPlant.finstate

The Continuous Plant programmed with ANSI C language

Data file management

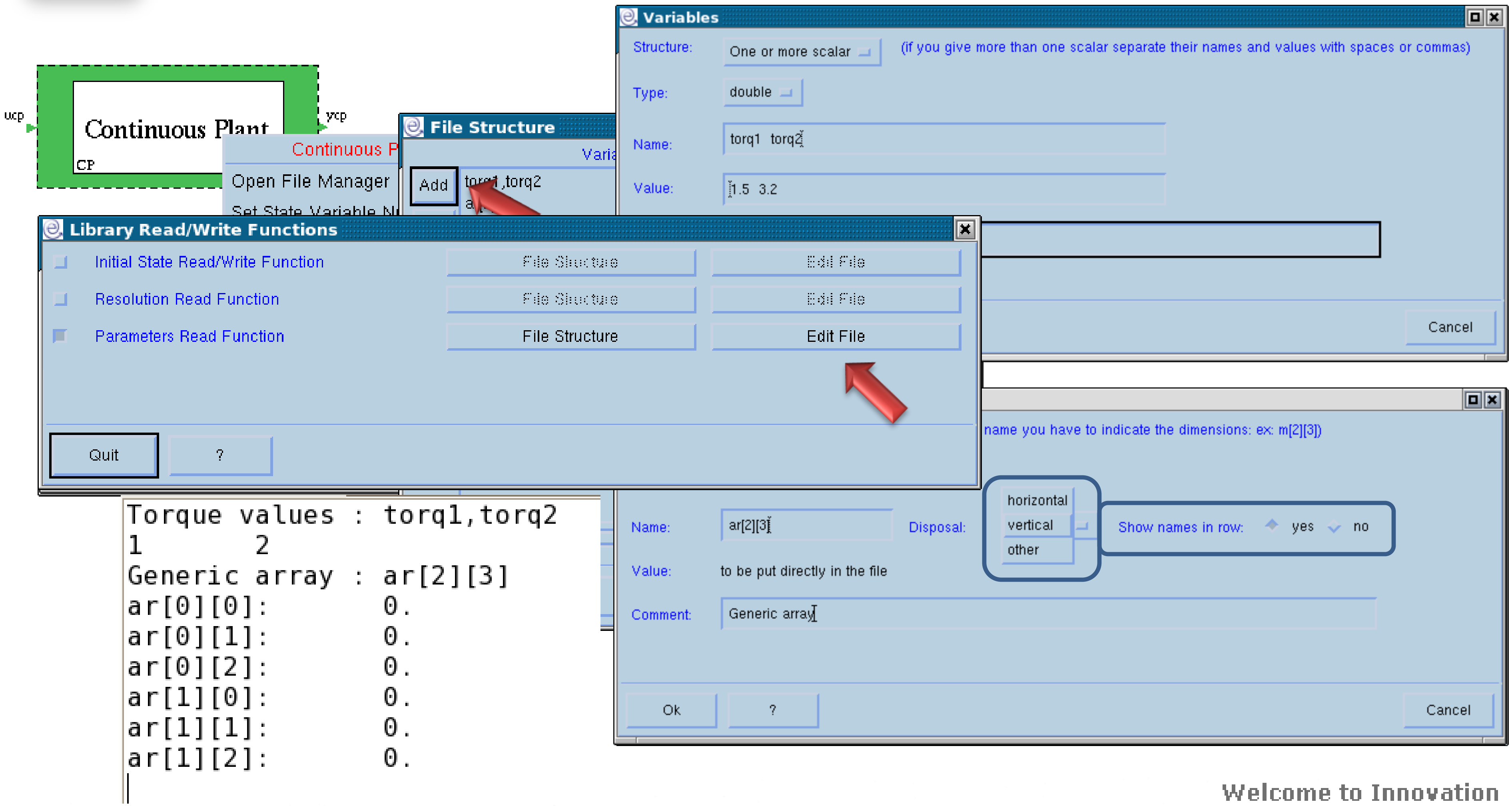
```
/***************************/
void    CP_ReadPar(FILE *fp)
/*
INPUTS:
fp      file pointer to the file ContPlant.par

OUTPUTS:
value of the Continuous Plant parameters

OBJECTIVES:
The function can read the parameter set of the plant mathematical model,
from the file ContPlant.par

All the parameters should be defined in:
        DB.h            database of the Continuous Plant Module

SCHEDULE:
The function is called by the EICASLAB simulator nucleus,
once at the beginning of the simulation,
before the function CP_ReadResol CP_ReadState CP_Ini.
*/
{


return;
}
/***************************/
```





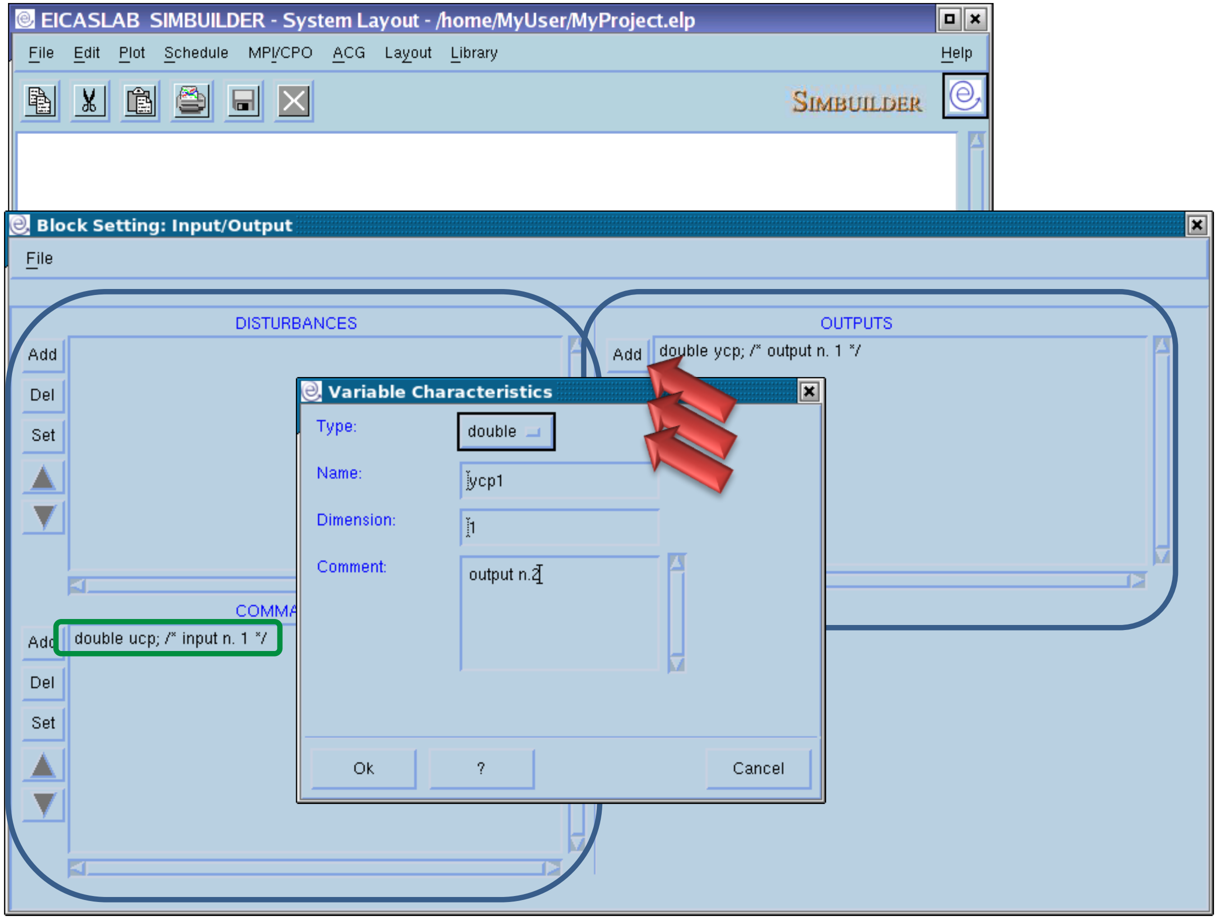
The Continuous Plant programmed with ANSI C language

The Library Read/Write Functions



The Continuous Plant programmed with ANSI C language

The Input/Output variables



The input/output variables of the block are defined by means of an appropriate window.

The input/output variables are ANSI C variables that can be used in any ANSI C function of the block.

The scheduling of the Continuous Plant functions

The Continuous Plant functions

The Continuous Plant may be programmed through a set of activities (functions):

- **Graphical** Continuous Plant:
  all the functions are entirely created and managed by EICASLAB and depend on the graphical scheme of the Continuous Plant Layout and on the data (e.g. parameters, resolution, states) directly inserted by the user.

- Continuous Plant programmed in **ANSI C**:
  all the functions have a template provided by EICASLAB and are managed by the user.

The scheduling of the Continuous Plant functions

The functions belong to three main categories:

Continuous Plant Activities

- Initialization functions → called just once at the beginning of the simulation
- Execution functions → periodically scheduled
- Final functions → called just once at the end of the simulation

The user has to fix a **simulation step**,
which represents the time resolution applied in the simulation of the overall project.

The outputs and the state variables of the Continuous Plant are updated
at each simulation step.

The scheduling of the Continuous Plant functions

Initialization functions

- **Graphical** Continuous Plant:
  → functions entirely created and managed by EICASLAB,
- Continuous Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

The initial functions are called just once at the beginning of the simulation, in the following order:

1) Parameter file reading,

2) Resolution file reading,

3) Initial state file reading,

4) User initialisation function (Only when programmed in ANSI C language).

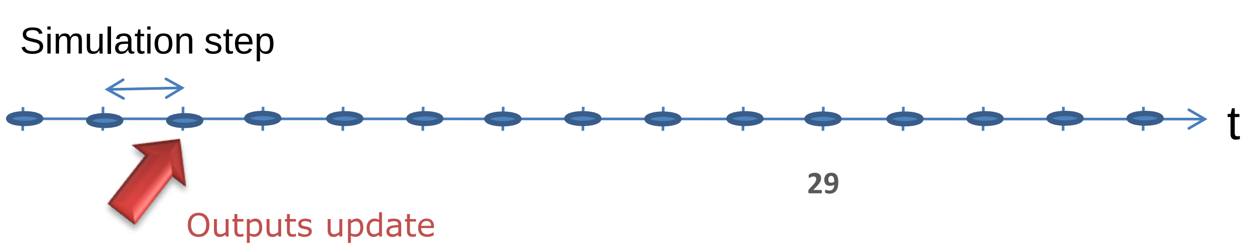
The scheduling of the Continuous Plant functions

Execution functions

- **Graphical** Continuous Plant:
  → functions entirely created and managed by EICASLAB,
- Continuous Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

State equation function	Computation of the state derivative	It is called by the EICASLAB routine that solves the system of differential equations
Output function	Computation of the outputs of the Continuous Plant (as a function of its current state)	It is called at each simulation step

The outputs of the Continuous Plant are updated at each simulation step.



The scheduling of the Continuous Plant functions

Final functions

- **Graphical** Continuous Plant:
  → functions entirely created and managed by EICASLAB,
- Continuous Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

The final functions are called just once at the end of the simulation in the following order:

1) User final function (Only when programmed in ANSI C language),

2) Final state file writing.

DP

The Discrete Plant

State equations

The Discrete Plant is a dynamic system described by a set of **state variables** that can be represented through a set of finite differences equations (the model uses a discrete time approach).

At each sample step the state of the dynamic system is computed as a function of the previous state and of the inputs through the finite differences equations that are called **state equations**:

$$x(i+1) = f(x(i),u(i))$$

The output of the Discrete Plant is computed as a function of its state:

$$y(i) = f(x(i))$$

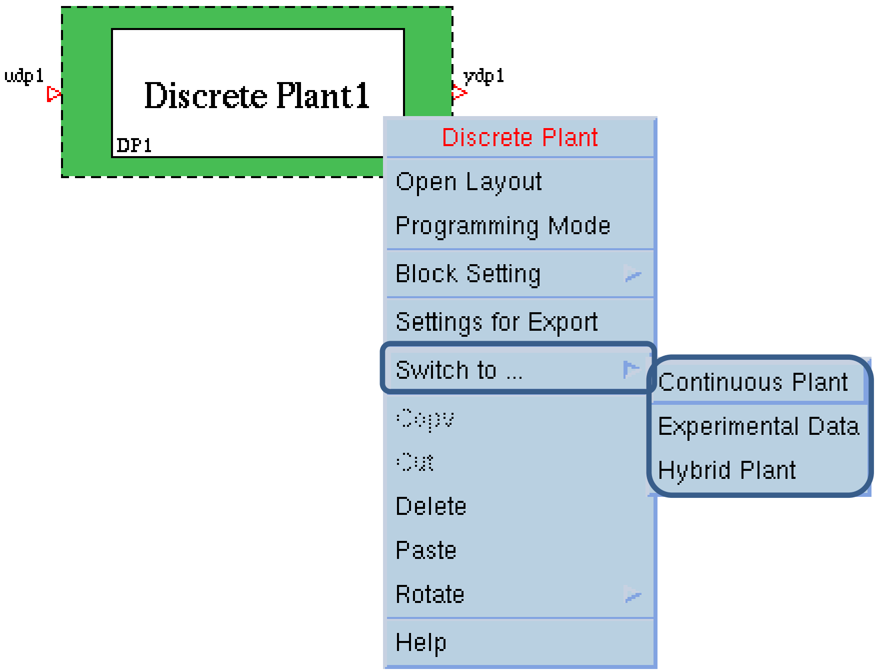
(having indicated:y: outputs, u inputs, x: states, par: parameters)

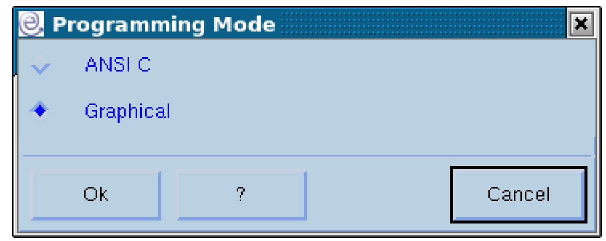
DP

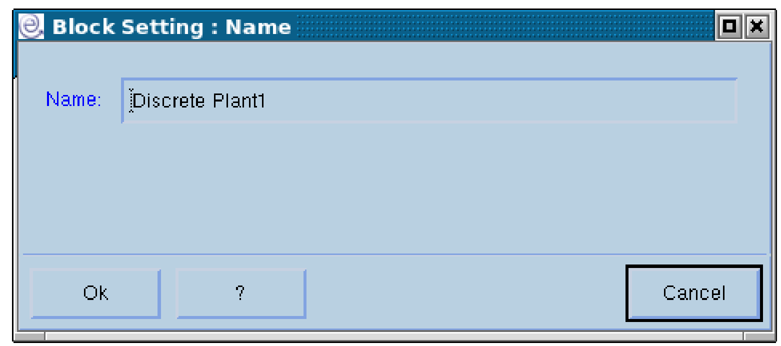
The Discrete Plant

Associated popup menu

The Discrete Plant is by default graphically programmed.



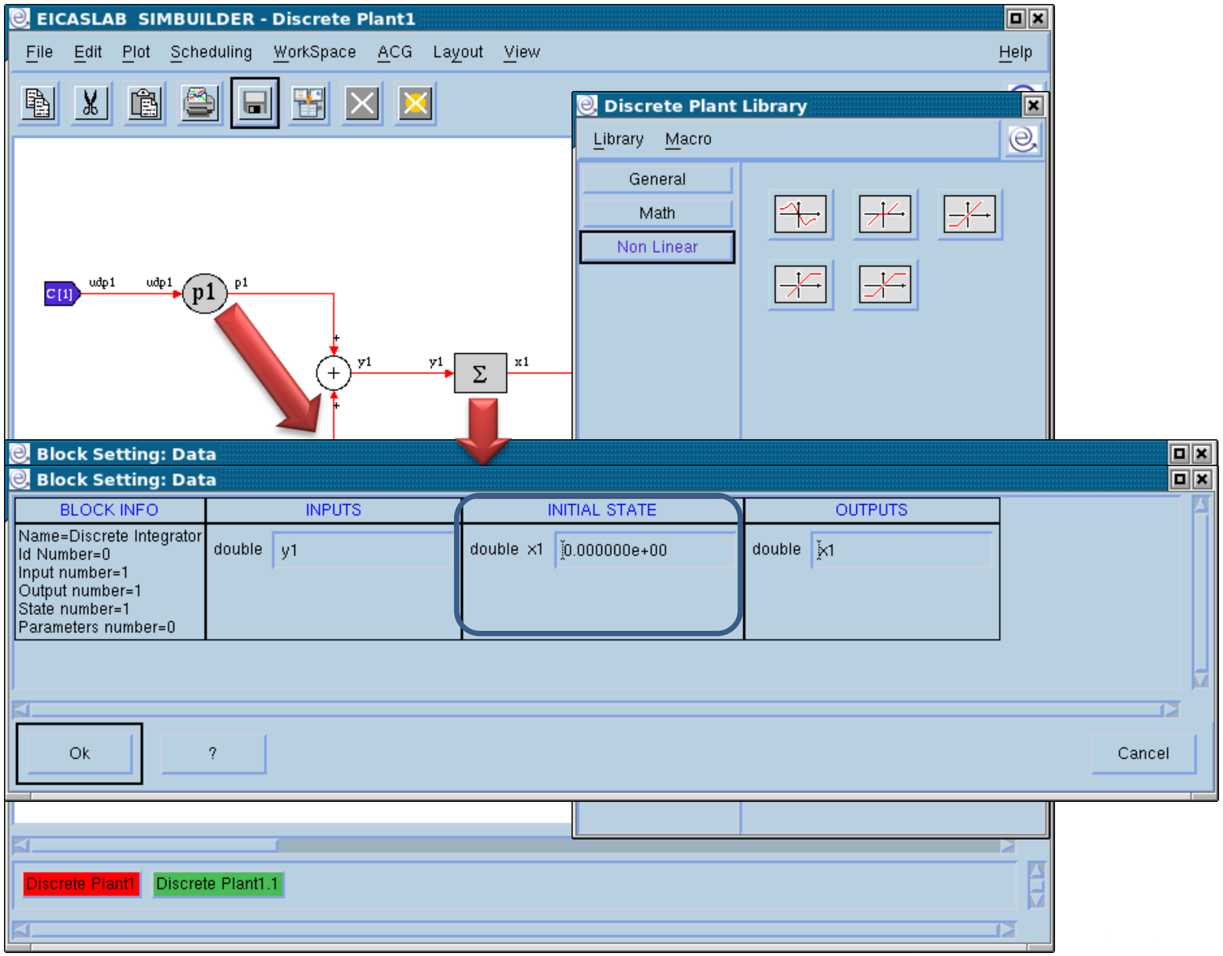




DP

The Discrete Plant graphically programmed

The Discrete Plant Layout



The Discrete Plant layout allows to graphically program the Discrete Plant.

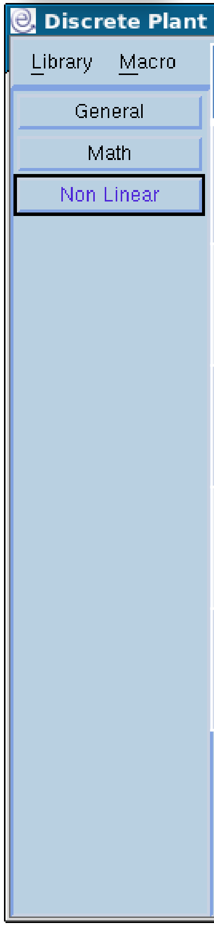
You can build your plant model by using the blocks available in the Discrete Plant Library window,

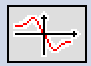
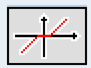
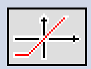
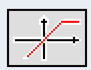
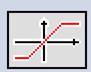
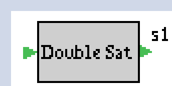
and by setting their:

- outputs,
- parameters,
- initial states (dynamic blocks).

The Discrete Plant graphically programmed

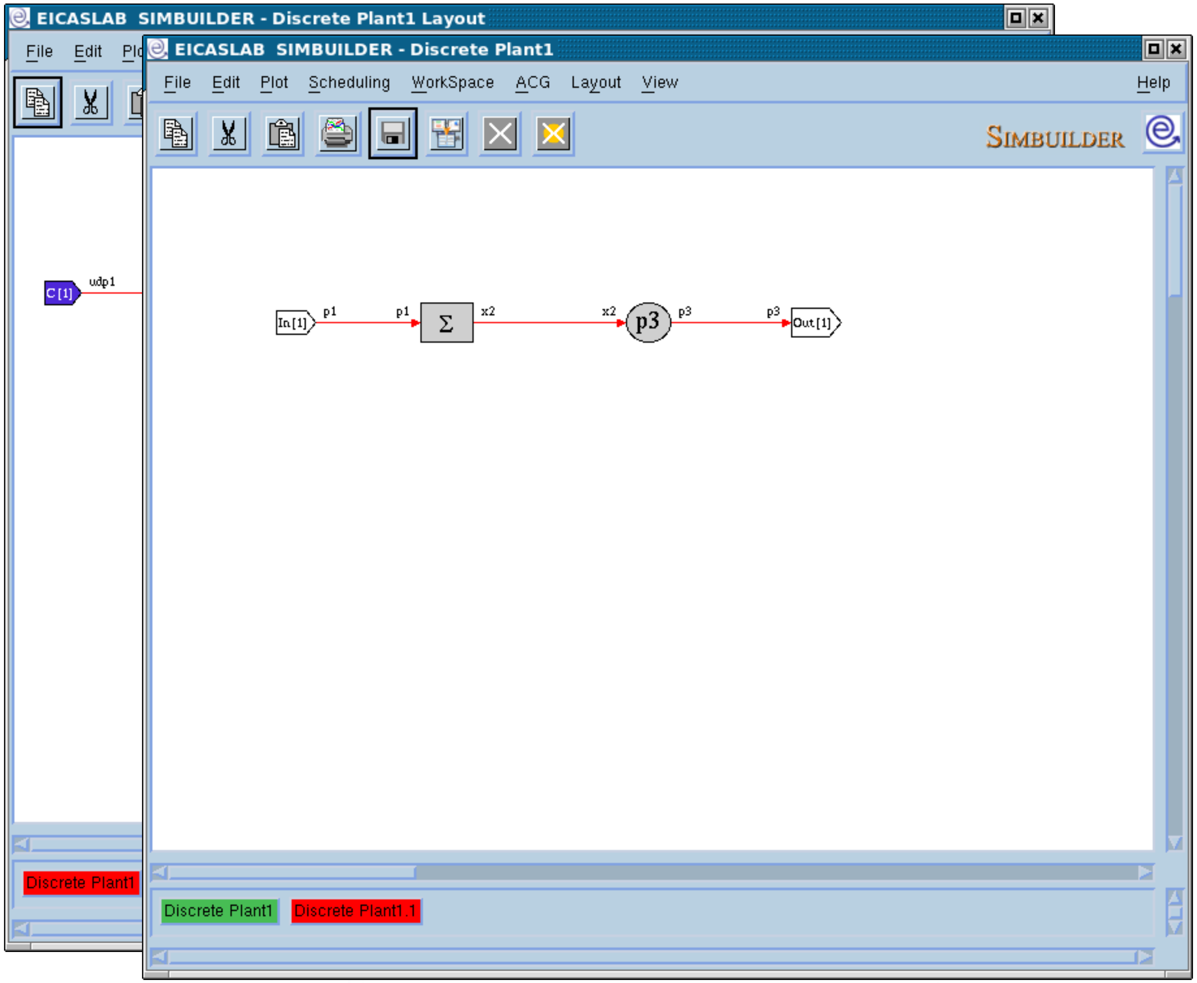
The non-linear library



Name	Icon in library	Block in the layout	decription
Coulomb Friction			Generate output according to a coulomb friction model
Dead Zone			Generate output according to a backlash model
Min Sat			Limit the lower value of a signal
Max Sat			Limit the upper value of a signal
Double Sat			Limit the range of a signal

The Discrete Plant graphically programmed

The subsystems

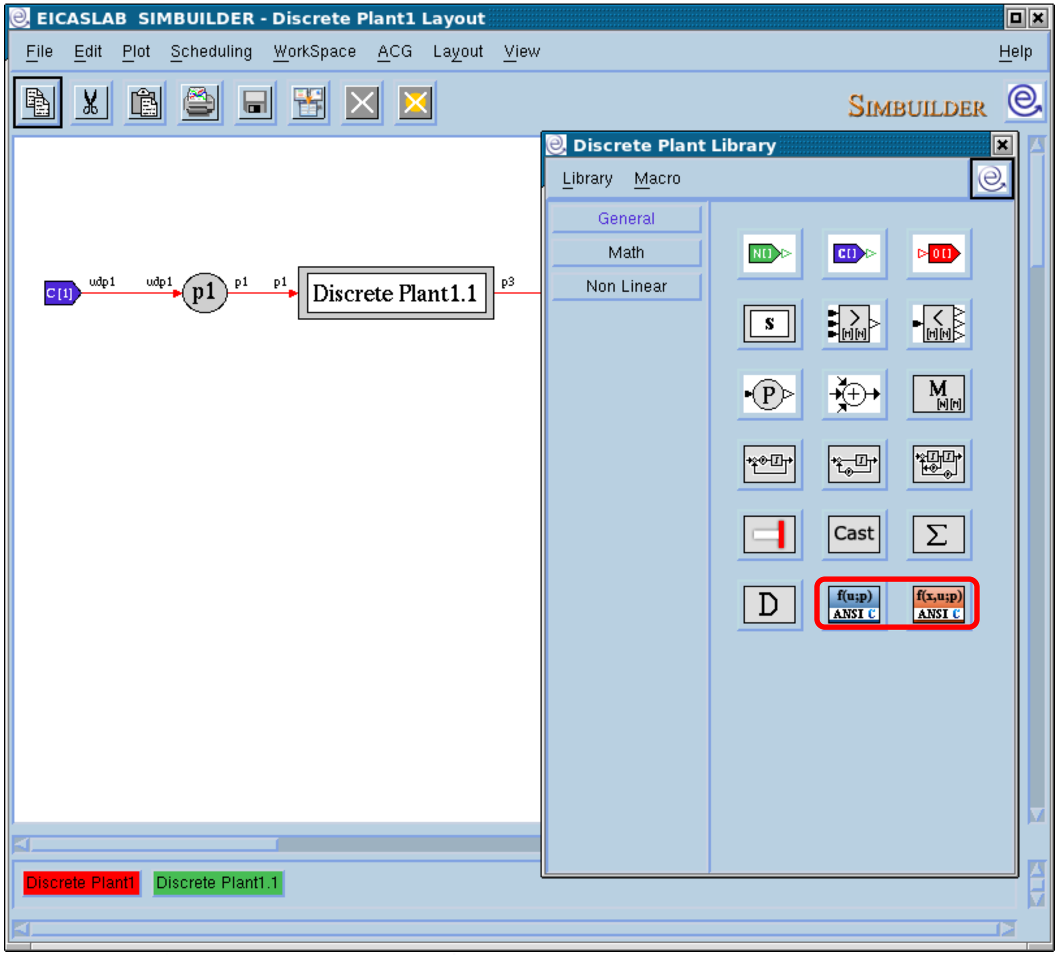


You can simplify the representation of your system by collecting parts of your block diagram in a block called **Subsystem**.

Double clicking on the subsystem opens the *Subsystem* layout, where you can use all the blocks available in the related library.
You can also create other subsystems in order to build a hierarchical block diagram.

The Discrete Plant graphically programmed

The ANSI C blocks



It is possible to use special blocks programmable in ANSI C language.

There are two types of blocks, allowing you to program in ANSI C language:

- static functions
  in this case the C block implements the function:
  y= f(u;par);

- dynamic functions
  in this case the C block implements the function:
  y= f(x,u;par);

(having indicated:
y: outputs, u inputs, x: states, par: parameters)

The Discrete Plant graphically programmed

The macros

The Discrete Plant library window is **customizable** with user blocks called **'macros'**.

The macros are created by the user in order to complete the library according to the user needs.

The macros can be programmed:

- **graphically** (working on the Graphical Macro layout) or
- **in ANSI C language**.

They are then available in the library window of the layout, as all the other blocks and can be used in the current project.

They can also be exported and then used in other projects.

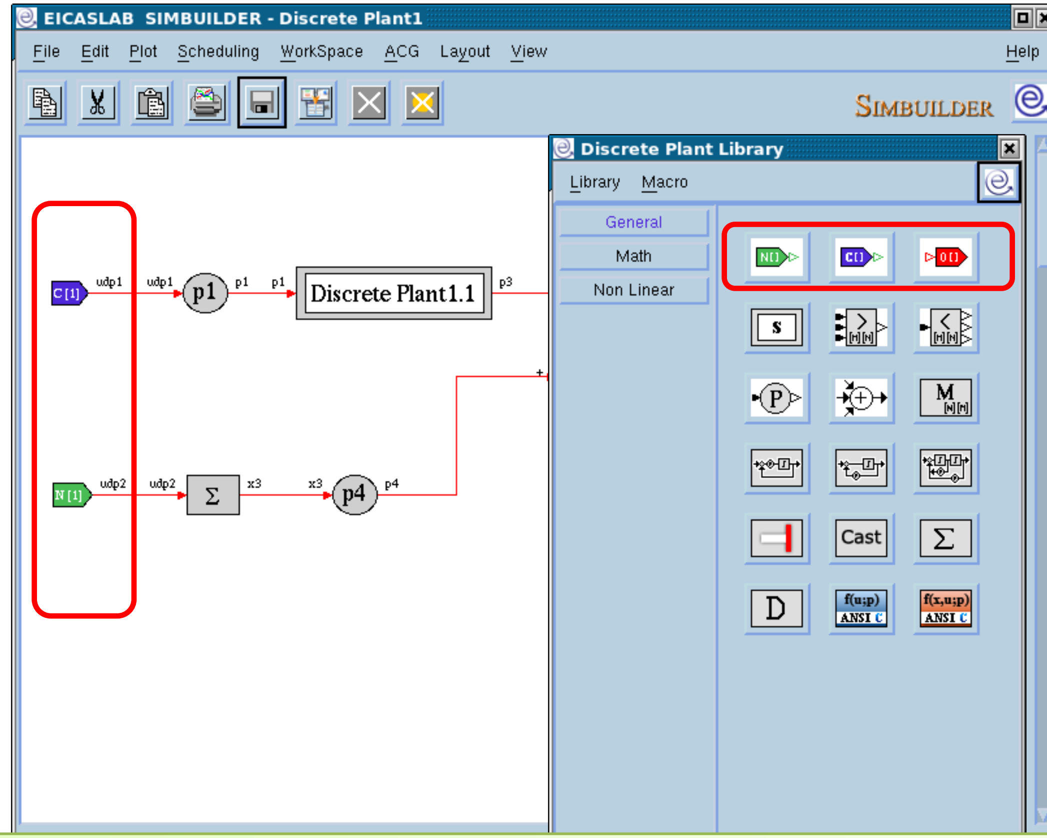
DP

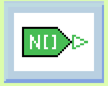
The Discrete Plant graphically programmed

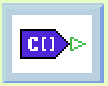
The Input/Output variables

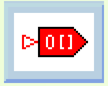
In order to define the inputs and the outputs of a graphically programmed block:

insert
inside the graphical layout
the input – outputs blocks.




Plant Noise Input


Plant Command Input


Plant Output

The Discrete Plant programmed with ANSI C language

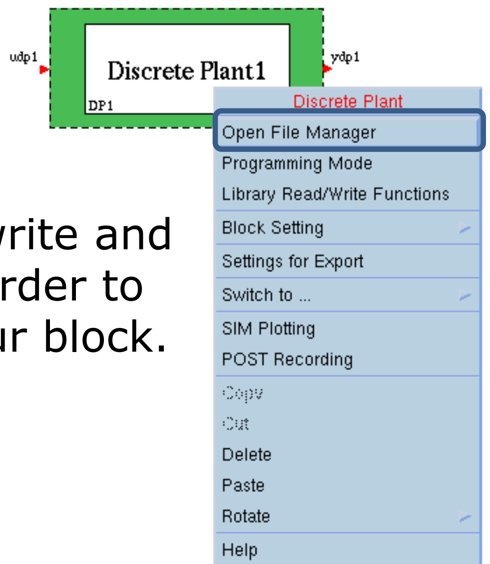
The file manager

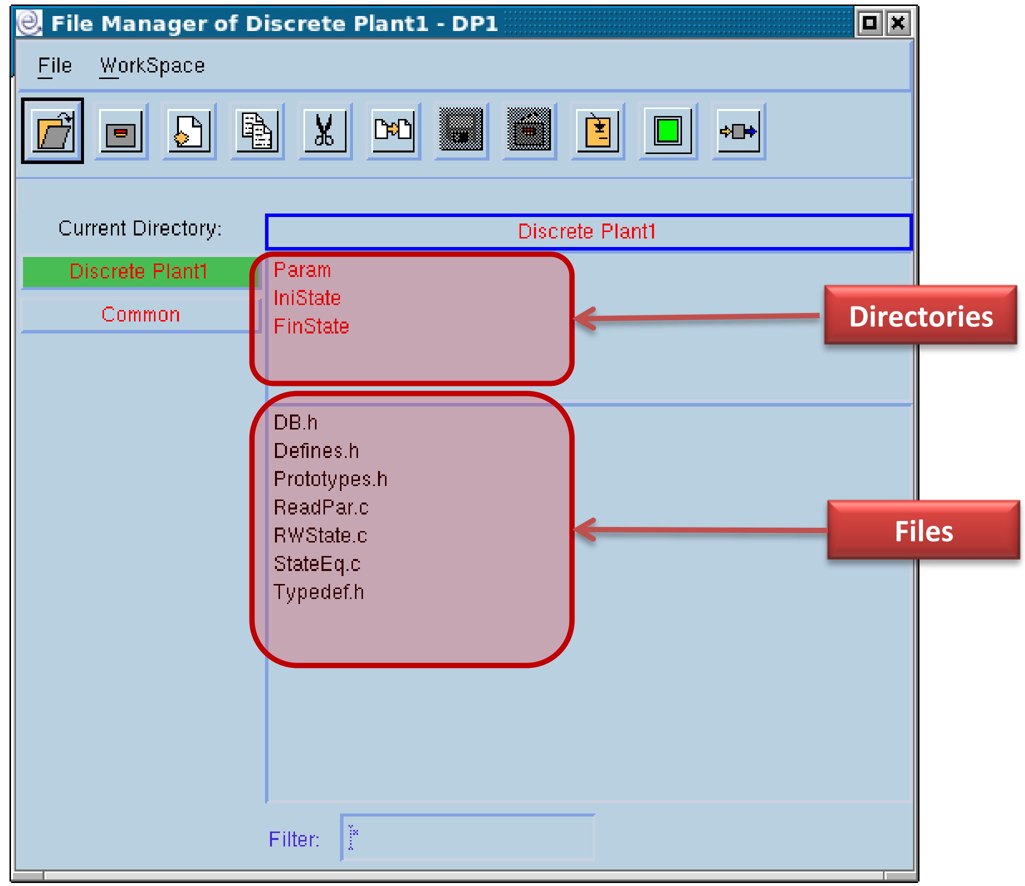
The Discrete Plant programmed with ANSI C language has its own file manager through which it is possible to program the block.

EICASLAB provides a pre-organised structure: a set of template files subdivided in:

- data files,
- header files,
- C files,

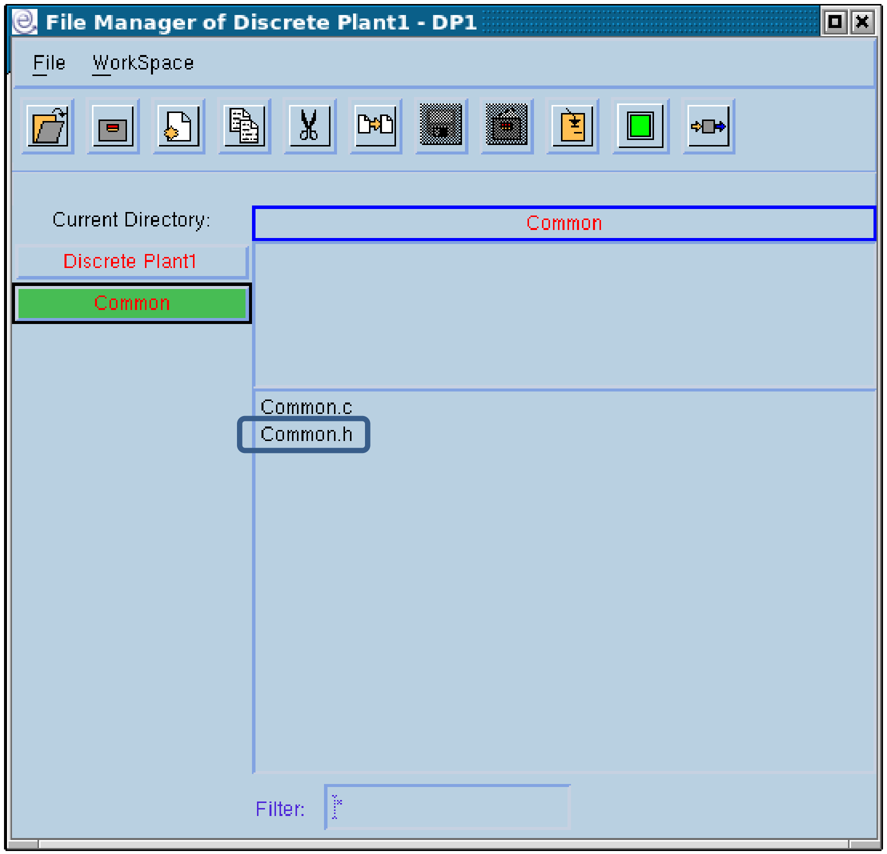
that you can write and customize in order to implement your block.





The Discrete Plant programmed with ANSI C language

The header files



Header files of the pre-organised structure that are written by the user.

Defines.h	Definition of user constants
Typedef.h	Definition of user structures
DB.h	Definition / declaration of user variables
Prototypes.h	Declaration of the function prototypes
Common.h	Available for all the blocks programmed in C

The Discrete Plant programmed with ANSI C language

Initialization functions

Name	Description	C File	Data File
DP#_ReadPar	Parameter file reading	ReadPar.c	DiscrPlant.par
DP#_ReadState	Initial state file reading	RWState.c	DiscrPlant.inistate
DP#_Ini	User initialisation function	StateEq.c	---

The Discrete Plant programmed with ANSI C language

Execution functions

Name	Description	C File
DP#_StateEq	Computation of the next state of the Discrete Plant as a function of its current state and of its inputs	StateEq.c
DP#_Out	Computation of the outputs of the Discrete Plant as a function of its current state	StateEq.c

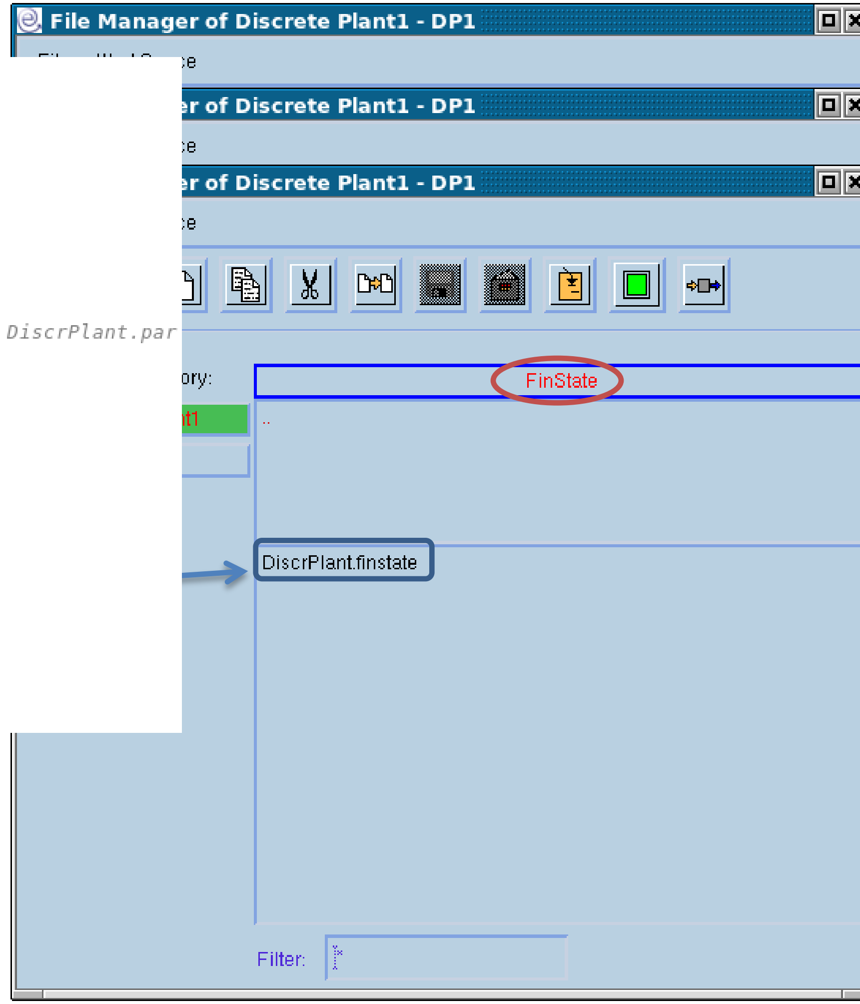
The Discrete Plant programmed with ANSI C language

Final functions

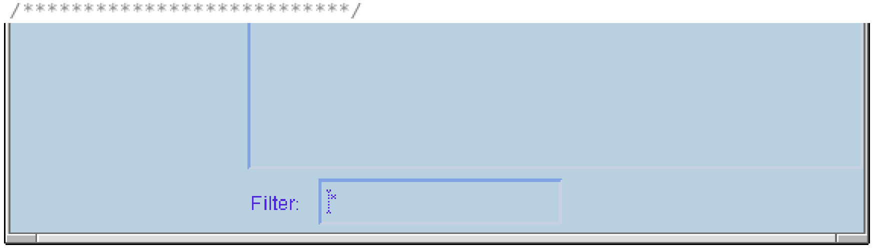
Name	Description	C File	Data File
DP#_Fin	User final function	StateEq.c	-
DP#_WriteState	Final state file writing	RWState.c	DiscrPlant.finstate

The Discrete Plant programmed with ANSI C language

Data file management

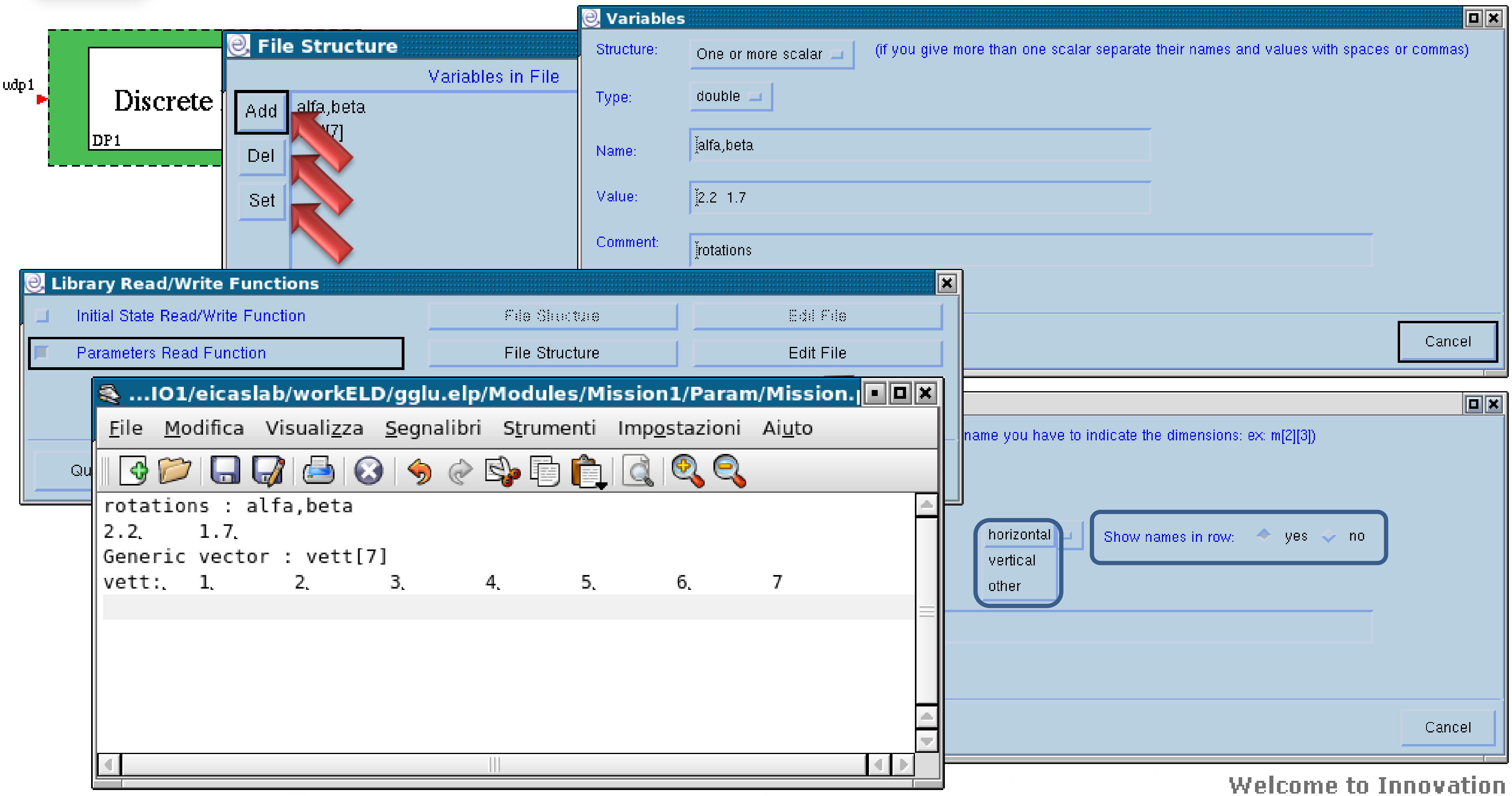



```
/***************************/
void    DP1_ReadPar(FILE *fp)
/*
INPUTS:
fp      file pointer to the file DiscrPlant.par

OUTPUTS:
value of the discrete plant parameters

OBJECTIVES:
The function can read the parameter set of the Discrete Plant1, from the file DiscrPlant.par

All the parameters should be defined in:
        DB.h            database of the Discrete Plant1 Module

SCHEDULE:
The function is called  by the EICASLAB simulator nucleus,
once at the beginning of the simulation session,
before the functions DP1_ReadState and DP1_Ini.
*/
{


return;
}
/***************************/
```



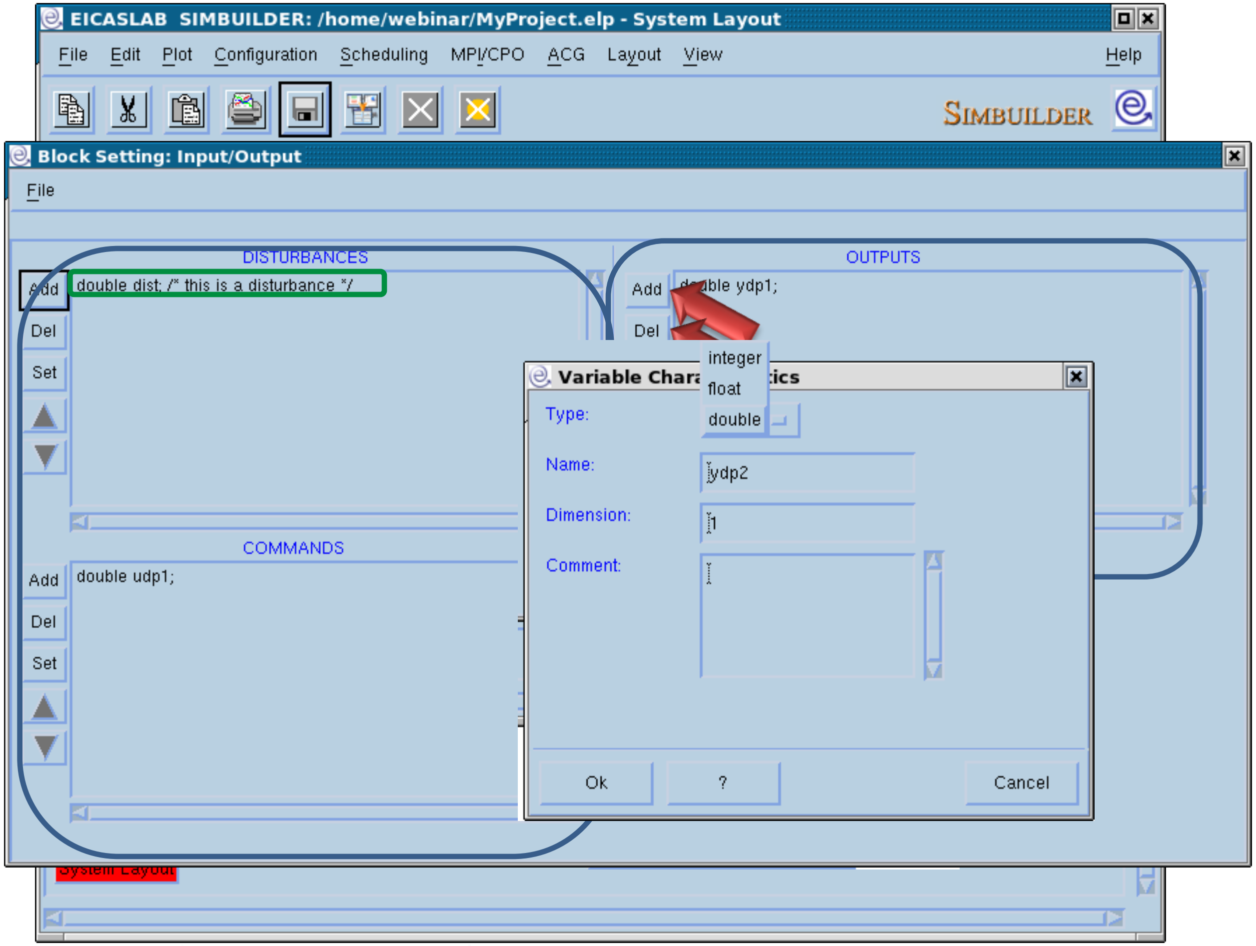
The Discrete Plant programmed with ANSI C language

The Library Read/Write Functions



The Discrete Plant programmed with ANSI C language

The Input/Output variables



The input/output variables of the block are defined by means of an appropriate window.

The input/output variables are ANSI C variables that can be used in any ANSI C function of the block.

The scheduling of the Discrete Plant functions

The Discrete Plant functions

The Discrete Plant may be programmed through a set of activities (functions):

- **Graphical** Discrete Plant:
  all the functions are entirely created and managed by EICASLAB and depend on the graphical scheme of the Discrete Plant Layout and on the data (e.g. parameters, states) directly inserted by the user.

- Discrete Plant programmed in **ANSI C**:
  all the functions have a template provided by EICASLAB and are managed by the user.

The scheduling of the Discrete Plant functions

Functions categories

The functions belong to three main categories:

Discrete Plant Activities

- Initialization functions → called just once at the beginning of the simulation
- Execution functions → periodically scheduled
- Final functions → called just once at the end of the simulation

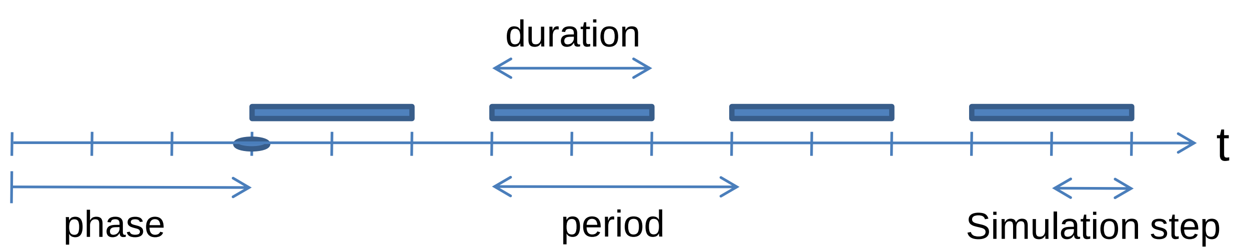
The scheduling of the Discrete Plant functions

Scheduling parameters

The user has to fix a **simulation step**,
which represents the time resolution applied in the simulation of the overall project.

The execution functions implement periodic activities characterized by the following scheduling parameters (expressed as a multiple of the simulation step):

- **Phase** time at which they are called for the first time,
- **Period** their sample time interval,
- **Duration** their execution time.



The scheduling of the Discrete Plant functions

Initialization functions

- **Graphical** Discrete Plant:
  → functions entirely created and managed by EICASLAB,
- Discrete Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

The initial functions are called just once at the beginning of the simulation, in the following order:

1) Parameter file reading,

2) Initial state file reading,

3) User initialisation function (Only when programmed in ANSI C language).

The scheduling of the Discrete Plant

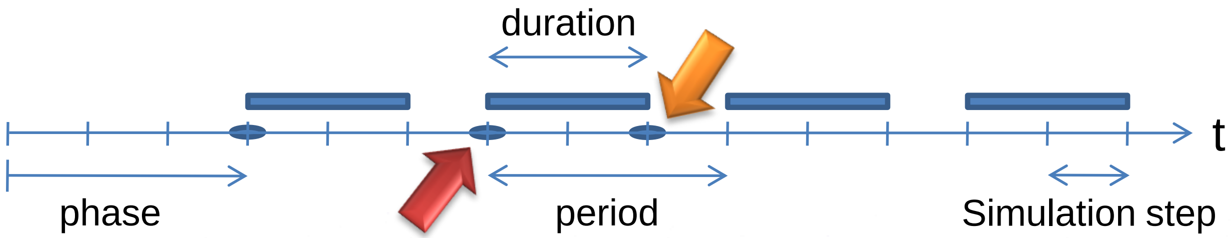
The execution functions

- **Graphical** Discrete Plant:
  → functions entirely created and managed by EICASLAB,
- Discrete Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

State equation function	Updating of the state of the Discrete Plant
Output function	Computation of the outputs of the Discrete Plant (as a function of its current state)

To guarantee the correct scheduling of the Discrete Plant it is necessary to take into account its **duration**:

State equation function	called when the Discrete Plant is scheduled (considering its **phase** and **period**),
Output function	called with the same period of the *state equation function* but with a delay equal to the **duration** of the Discrete Plant in order to provide the outputs when they are expected



The scheduling of the Discrete Plant functions

Final functions

- **Graphical** Discrete Plant:
  → functions entirely created and managed by EICASLAB,
- Discrete Plant programmed in **ANSI C:**
  → functions created by EICASLAB (template) and managed by the user.

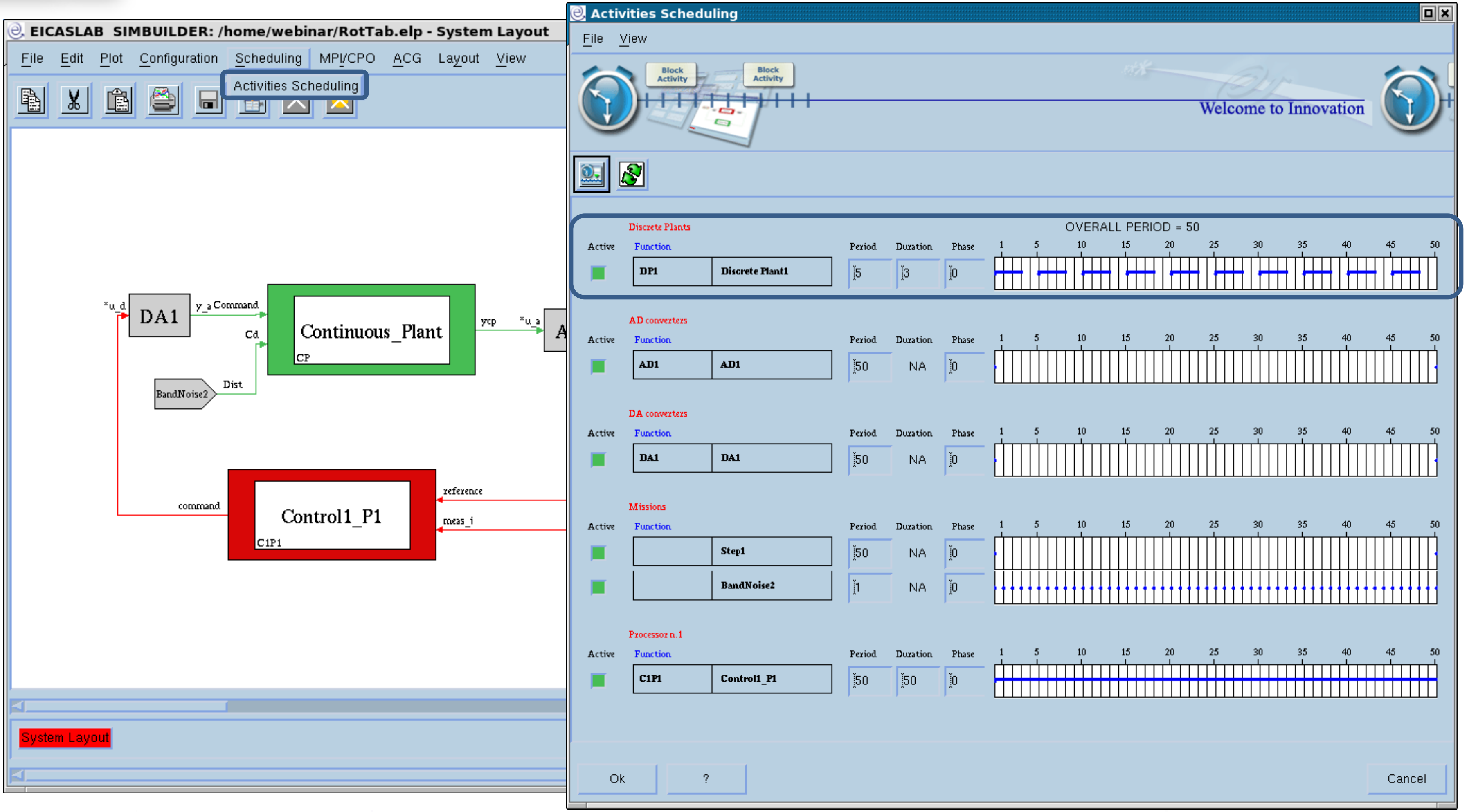
The final functions are called just once at the end of the simulation in the following order:

1) User final function (Only when programmed in ANSI C language),

2) Final state file writing.

The scheduling of the Discrete Plant

How to set the scheduling



The Experimental Data

Concept

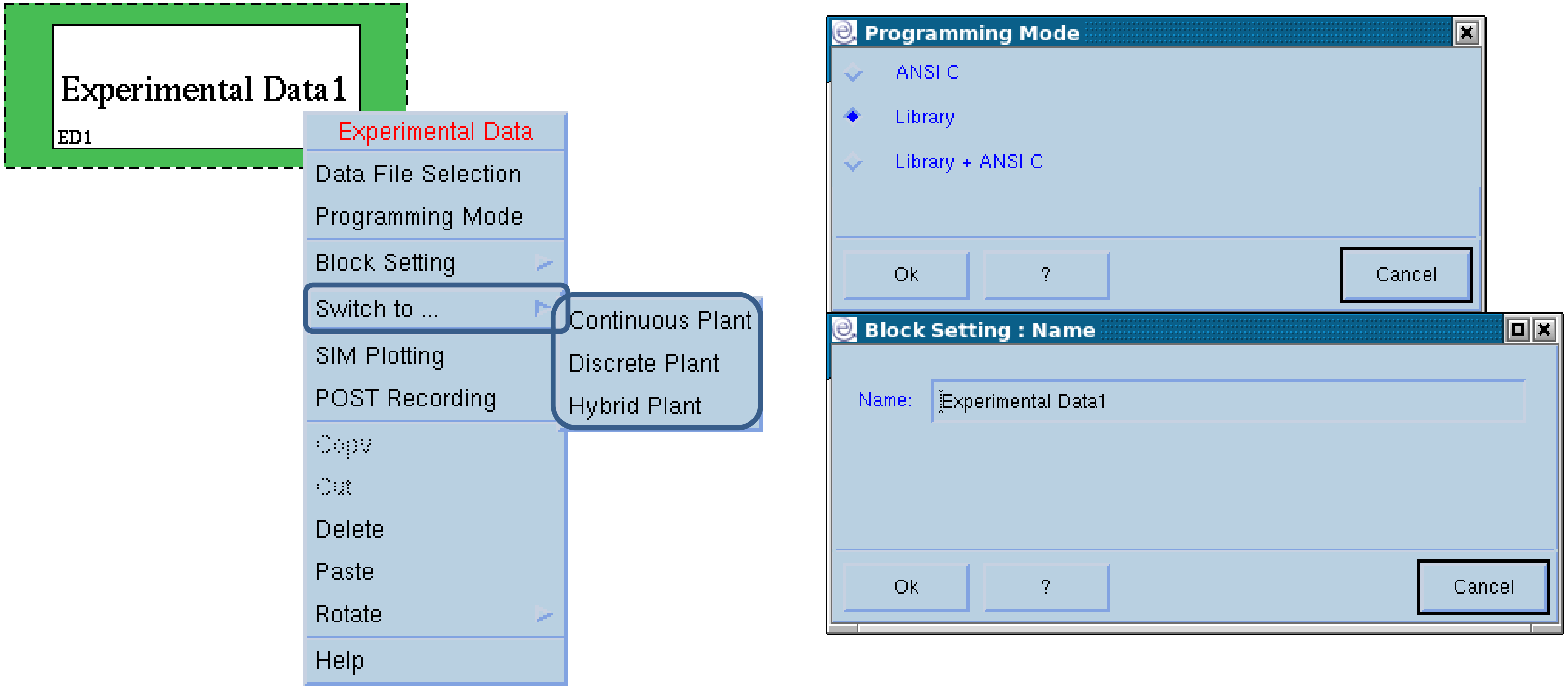
The Experimental Data allows to substitute the Plant model with a set of data collected on field during experimental trials.

It is then possible to perform simulations using directly the on field data instead of data computed by means of a Continuous or a Discrete Plant.

The Experimental Data

Associated popup menu

The Experimental Data is by default a library programmed block.



The Experimental Data: the library programming mode

The format of the Experimental Data File

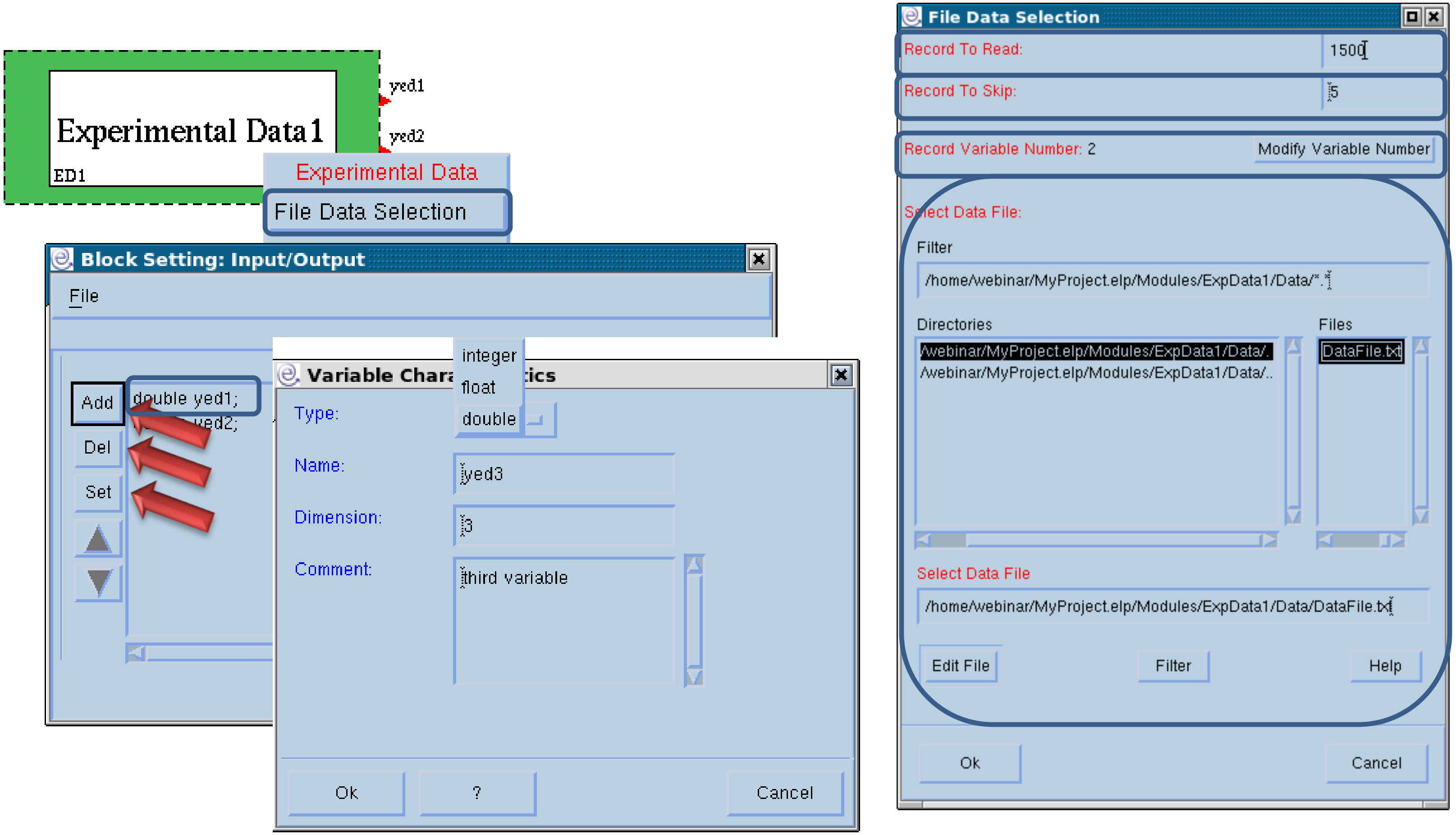
The **Experimental Data file** is a text file (formatted file) where each line contains data collected at the same time: It contains one sample for every variable to read.

In this way there are as many columns as the number of variables to read and as many lines as the total number of sample steps corresponding to the duration of the experimental trial.

Samples	Variables: V1	V2	...	Vm
S1	V1 S1	V2 S1	... S1	Vm S1
S2	V1 S2	V2 S2	... S2	Vm S2
S3	V1 S3	V2 S3	... S3	Vm S3
...	V1 ...	V2 ...	... ...	Vm ...
Sn	V1 Sn	V2 Sn	... Sn	Vm Sn

The Experimental Data: the library programming mode

The File Data Selection and Input/Output variables



The Experimental Data programmed with ANSI C language

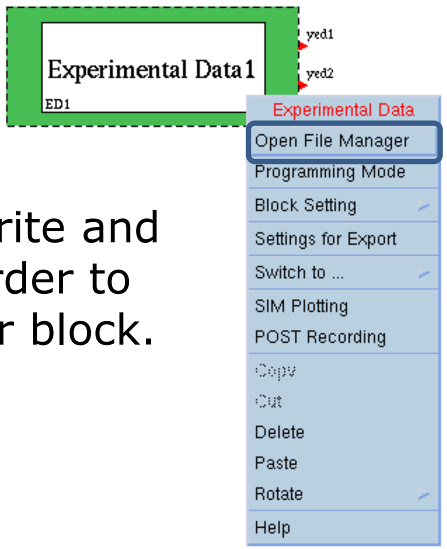
The file manager

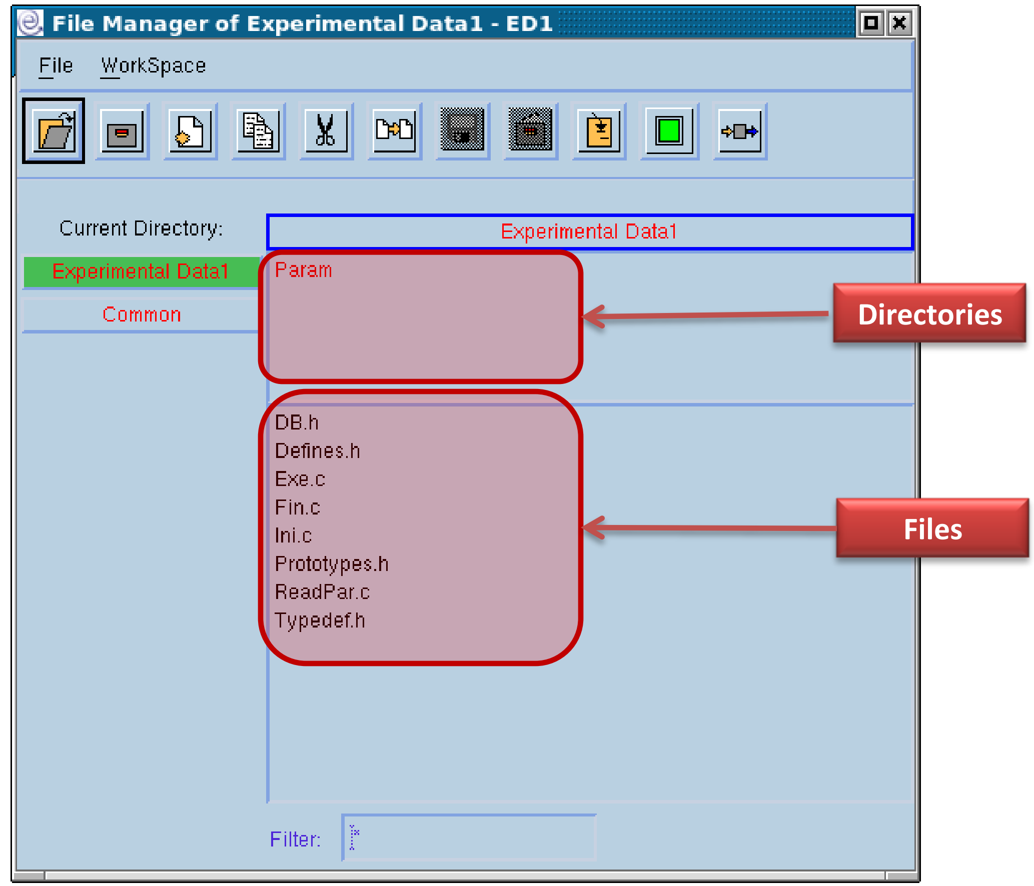
The Experimental Data programmed with ANSI C language has its own file manager through which it is possible to program the block.

EICASLAB provides a pre-organised structure: a set of template files subdivided in:

- data files,
- header files,
- C files,

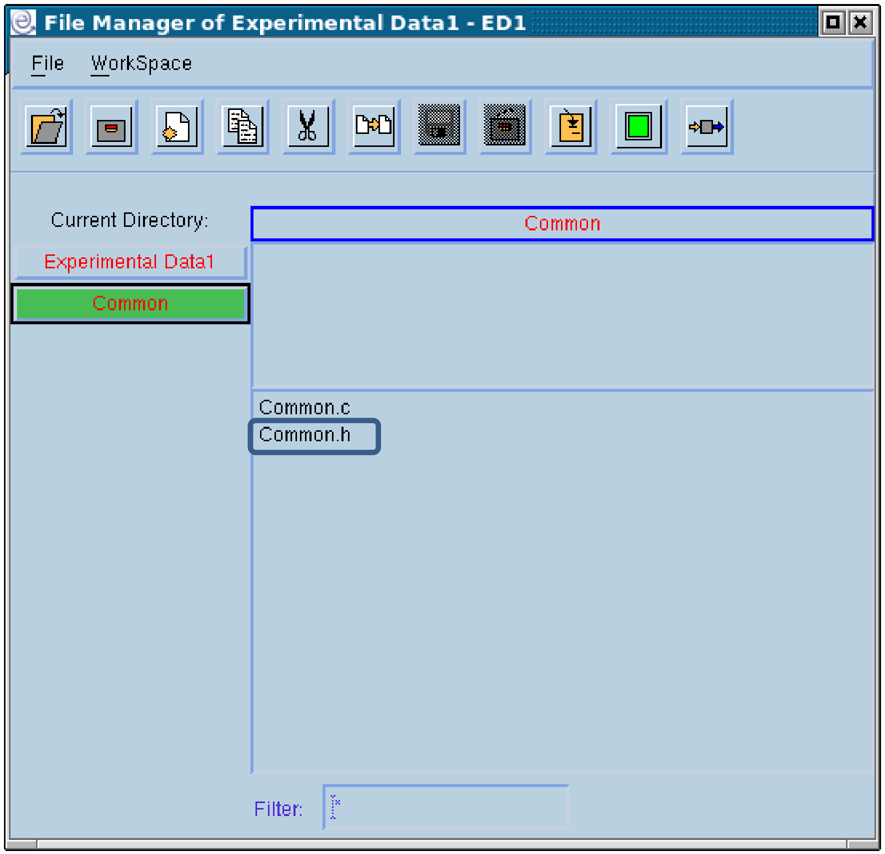
that you can write and customize in order to implement your block.





The Experimental Data: the ANSI C programming mode

The header files



Header files of the pre-organised structure that are written by the user.

Defines.h	Definition of user constants
Typedef.h	Definition of user structures
DB.h	Definition / declaration of user variables
Prototypes.h	Declaration of the function prototypes
Common.h	Available for all the blocks programmed in C

The Experimental Data programmed with ANSI C language

Initialisation functions

Name	Description	C File	Data File
ED#_ReadPar	Parameter file reading	ReadPar.c	ExpData.par
ED#_In	User initialisa on function	I .c	

The Experimental Data programmed with ANSI C language

Execution function

Name	Description	C File	Data File
ED#_Exe	Read one record of the Experimental Data file	Exe.c	---

The Experimental Data programmed with ANSI C language

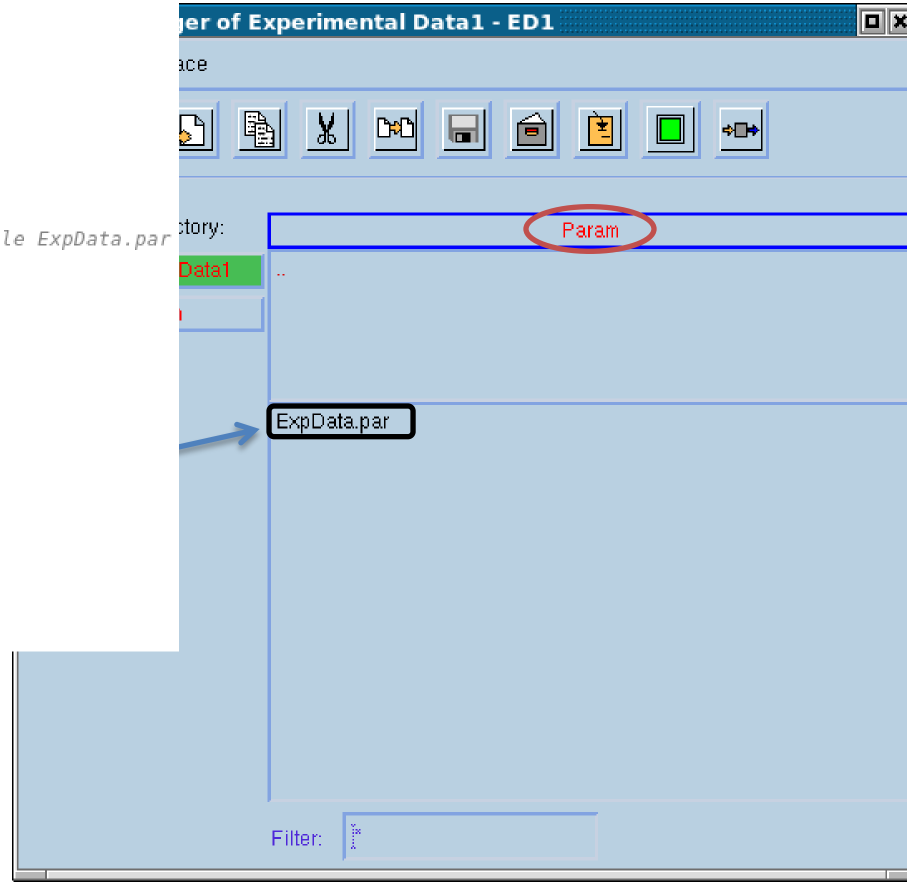
Final functions

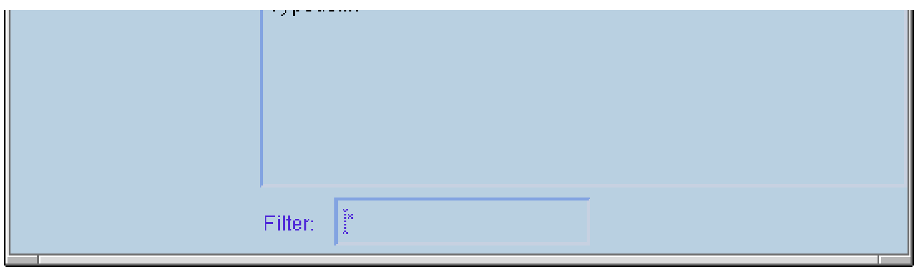
Name	Description	C File	Data File
ED#_Fin	User final function	Fin.c	---

The Experimental Data programmed with ANSI C language

Data file management

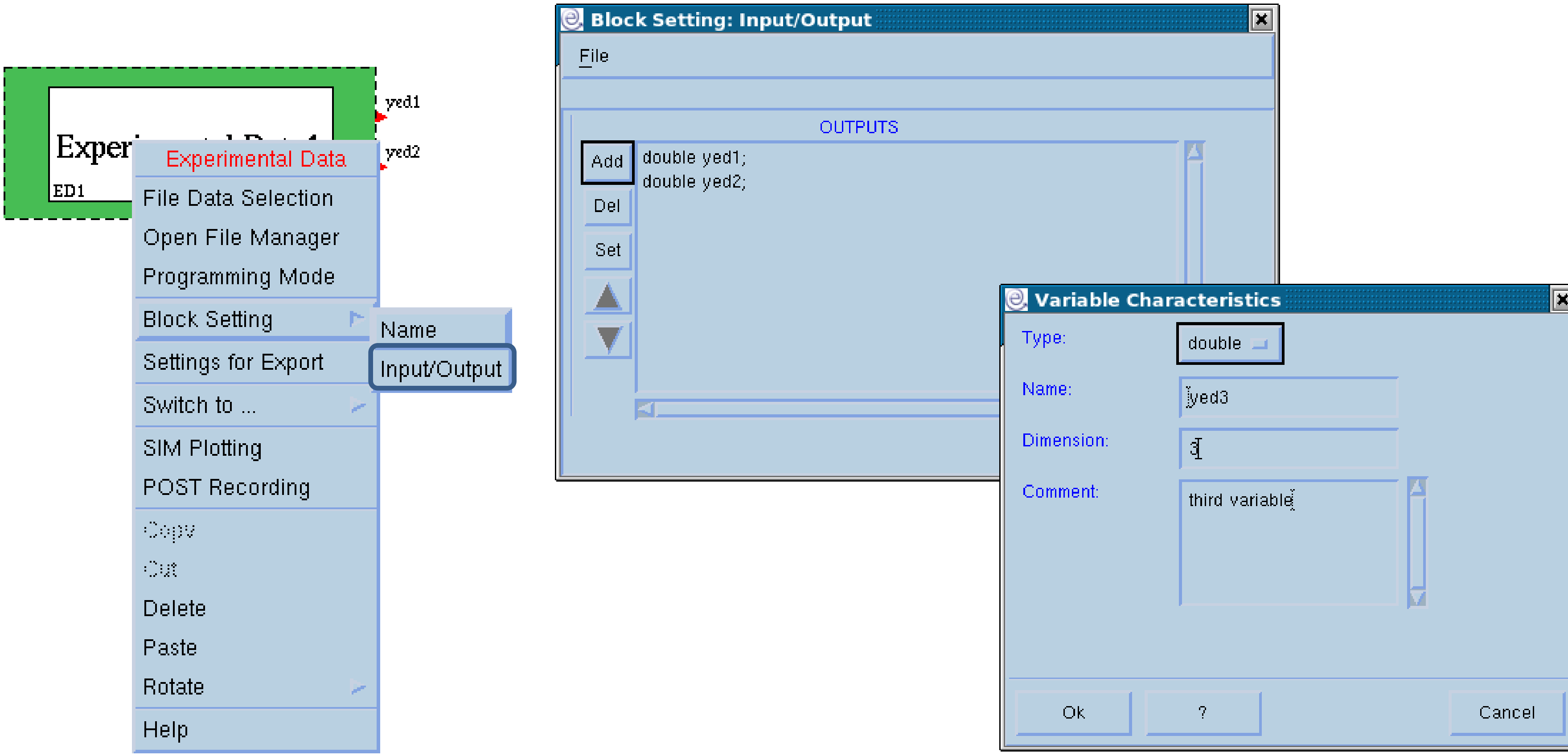
```
/**************************/
void     ED1_ReadPar(FILE *fp)
/*
INPUTS:
fp       file pointer to the file ExpData.par

OUTPUTS:
value of the Experimental Data1 parameters

OBJECTIVES:
The function can read the parameter set of the Experimental Data1, from the file ExpData.par

All the parameters should be defined in:
         DB.h              database of the Experimental Data1 Module

SCHEDULE:
The function is called by the EICASLAB simulator nucleus,
once at the beginning of the simulation,
before the function ED1_Ini.
*/
{


return;
}
/**************************/
```





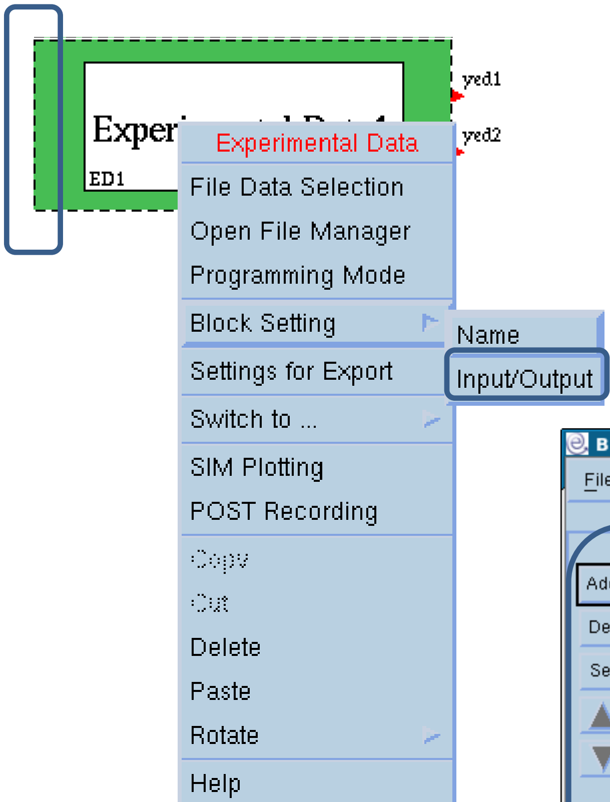
The Experimental Data programmed with ANSI C language

The Input/Output variables

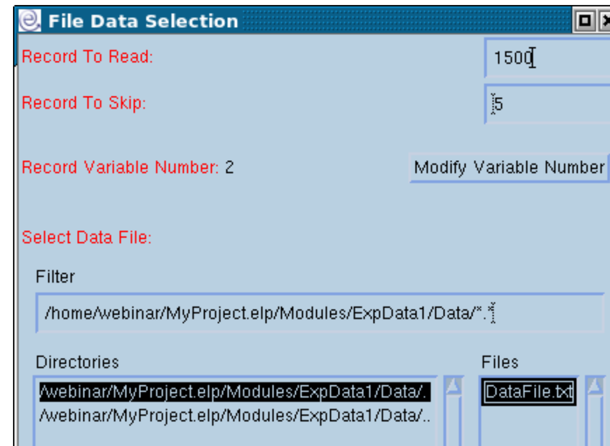


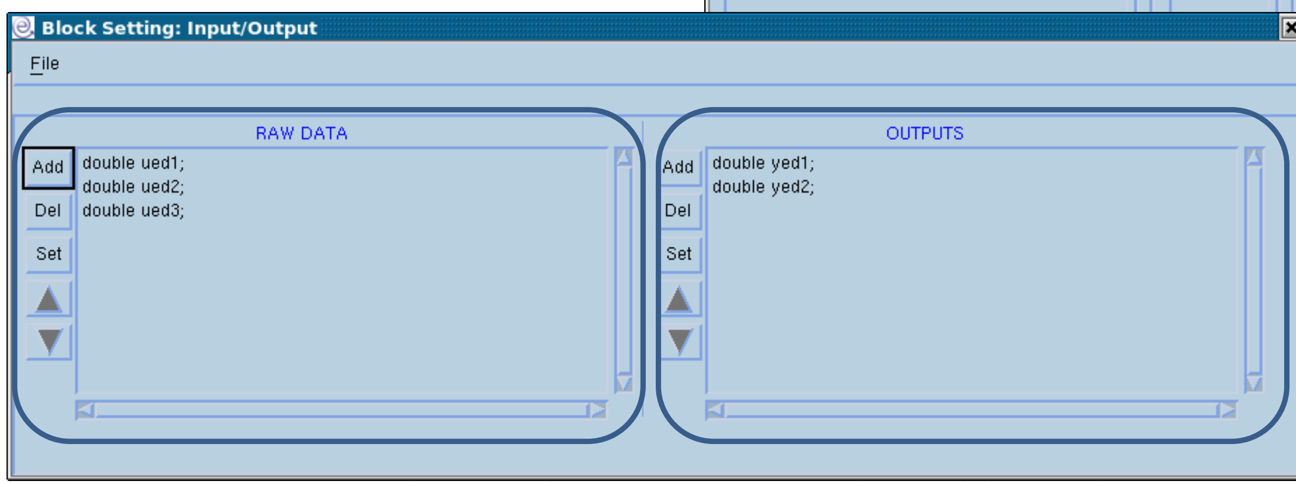
The Experimental Data: The Library + ANSI C programming mode

The library reading and the user post-processing



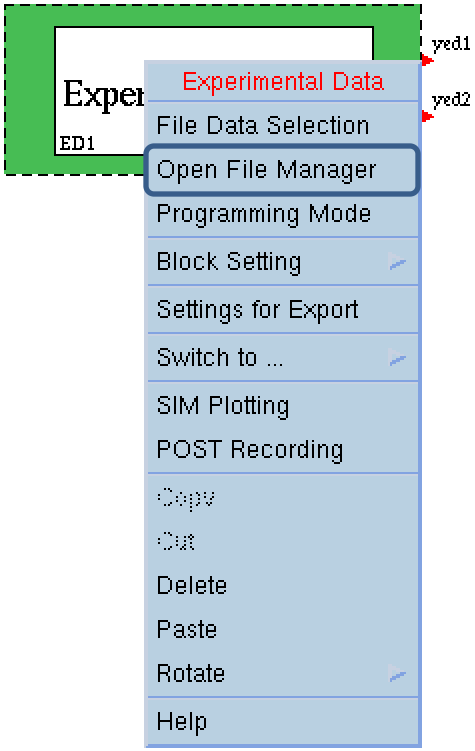
The Experimental Data file is automatically read by the library function and its outputs (the **raw data**) are post-processed by a user function which computes the **outputs** of the Experimental Data block.

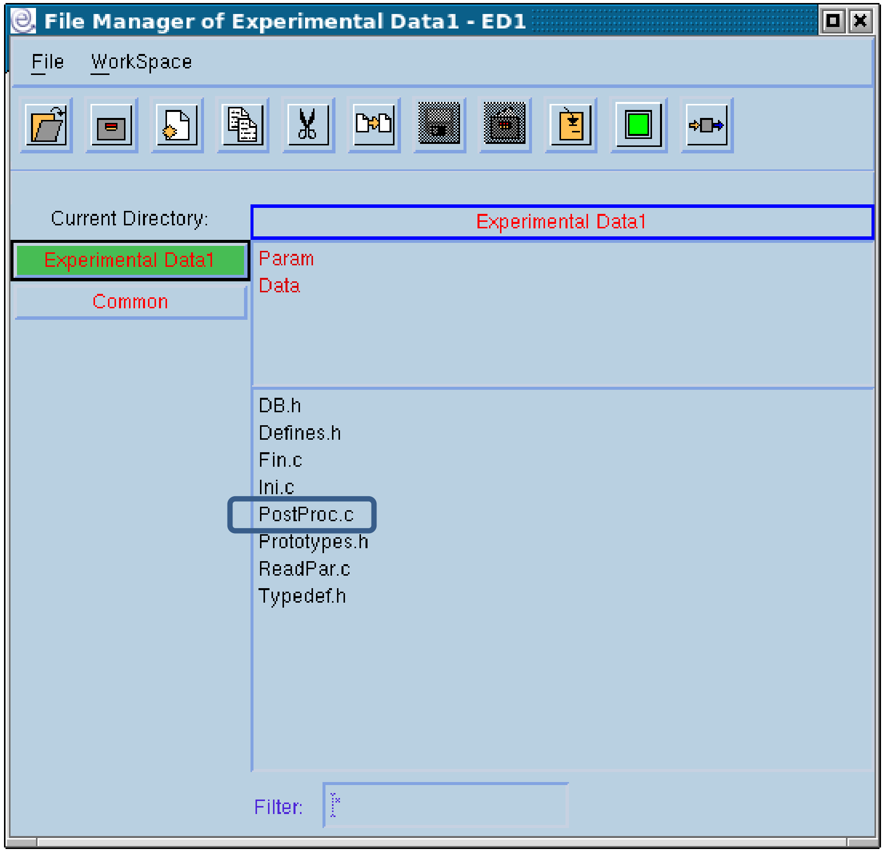




The Experimental Data: The Library + ANSI C programming mode

The file manager





The Experimental Data: The Library + ANSI C programming mode

ANSI C functions

Name	Description	C File	Data File
ED#_ReadPar	Parameter file reading	ReadPar.c	ExpData.par
ED#_Ini	User initialisation function	Ini.c	---
ED#_PostProc	Post-processing of the data read from the Experimental Data file	PostProc.c	---
ED#_Fin	User final function	Fin.c	---

The scheduling of the Experimental Data functions

The Experimental Data functions

The Experimental Data may be programmed through a set of activities (functions):

- **Library** Experimental Data:
  all the functions are entirely created and managed by EICASLAB.
- Experimental programmed in **ANSI C** language:
  all the functions have a template provided by EICASLAB and
  are managed by the user.
- Experimental programmed with a combination of **library** functions and **ANSI C** language:
  the functions are managed by the user
  except the library functions for reading the data file.

The scheduling of the Experimental Data

Functions categories

The functions belong to three main categories:

Experimental Data Activities

- Initialization functions → called just once at the beginning of the simulation
- Execution functions → periodically scheduled
- Final functions → called just once at the end of the simulation

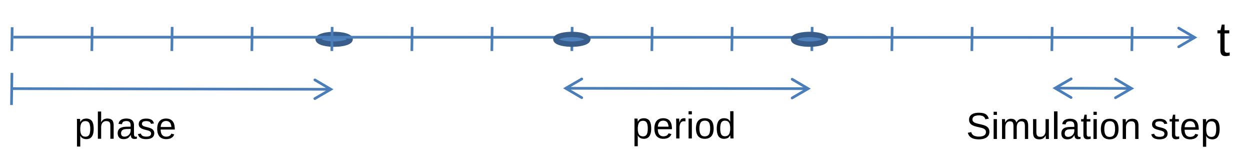
The scheduling of the Experimental Data functions

Scheduling parameters

The user has to fix a **simulation step**,
which represents the time resolution applied in the simulation of the overall project.

The execution function implements a periodic activity characterized by the following scheduling parameters (expressed as a multiple of the simulation step):

- **Phase** time at which it is called for the first time,
- **Period** its sample time interval.



The scheduling of the Experimental Data functions

Initialization and final functions

Function description	Note	Order of scheduling

The initial functions are called just once at the beginning of the simulation.

Parameter file reading	Only when programmed in ANSI C language	↓
User initialisation function	Only when programmed in ANSI C language	

The final function is called just once at the end of the simulation.

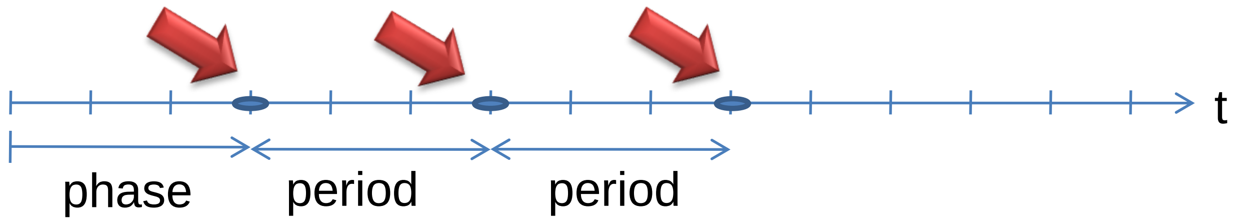
User final function	Only when programmed in ANSI C language	

The scheduling of the Experimental Data functions

The execution function

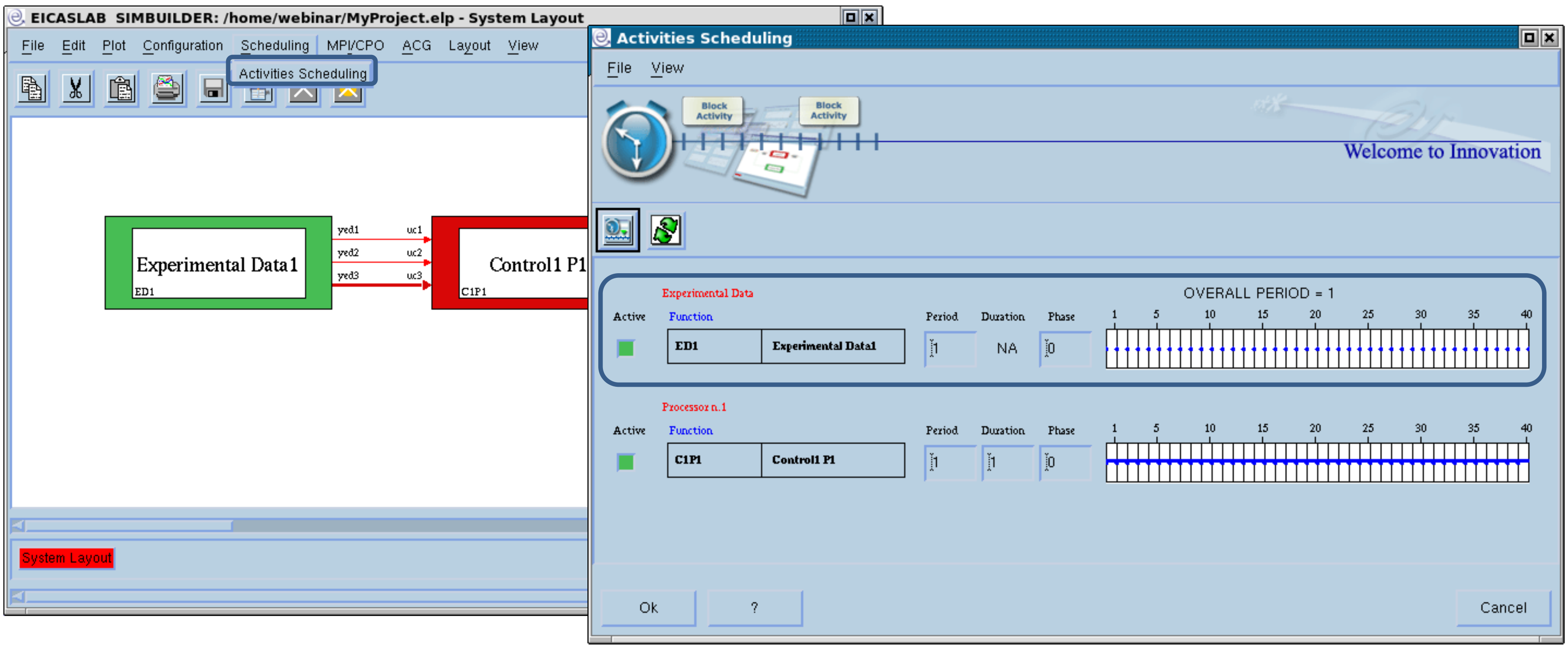
The Experimental Data has one execution function
which reads the data file and, if it is requested, a post-processing function:

they are instantaneous functions
called when the block is scheduled (considering its **phase** and its **period**).

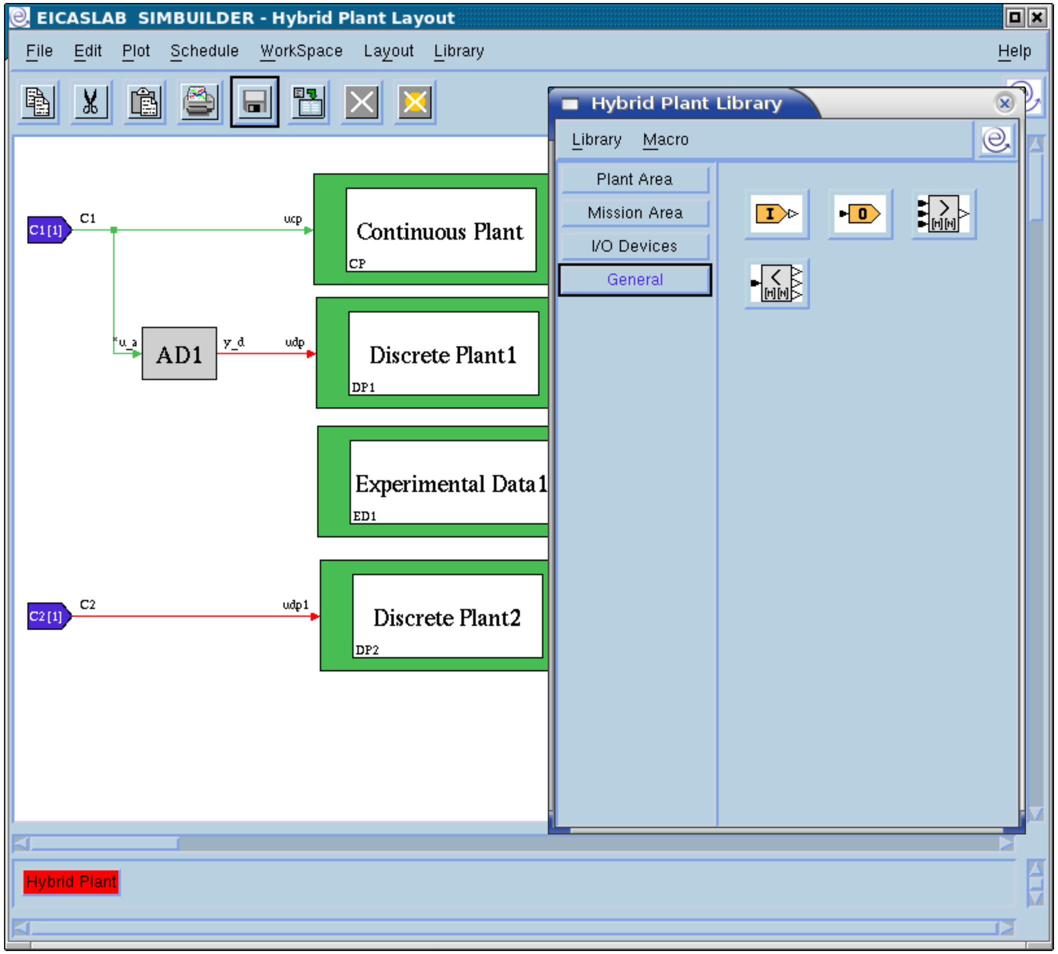


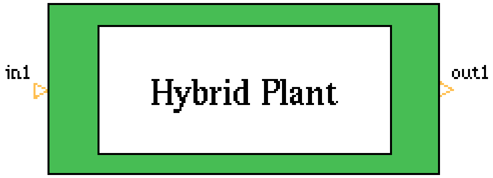
The scheduling of the Experimental Data functions

How to set the scheduling



The Hybrid Plant





The Hybrid Plant is a block graphically programmed that allows to group the Plant Area blocks (Continuous and Discrete Plants and Experimental Data), Plant Mission blocks (for modelling disturbances acting on the plant) and converters.

EICASLAB™

The Professional Software Suite
for Automatic Control Design
and Forecasting

for Linux & for Windows

www.eicaslab.com

Welcome to Innovation