



A/D and D/A Converters in EICASLAB™



Welcome to Innovation



TABLE OF CONTENT

- General description of the A/D and D/A converters
- The Library Converters
- The ANSI C Converters
- The scheduling of the A/D and D/A converters



Concept

The A/D and D/A **converters** are the interface between **analogical** and **digital** signals.

There are then 2 types of converters:

- the analogical → digital (A/D) converters (**sensors**),
- the digital → analogical (D/A) converters (**actuators**).

All the digital signals are characterized by a **quantization level**.

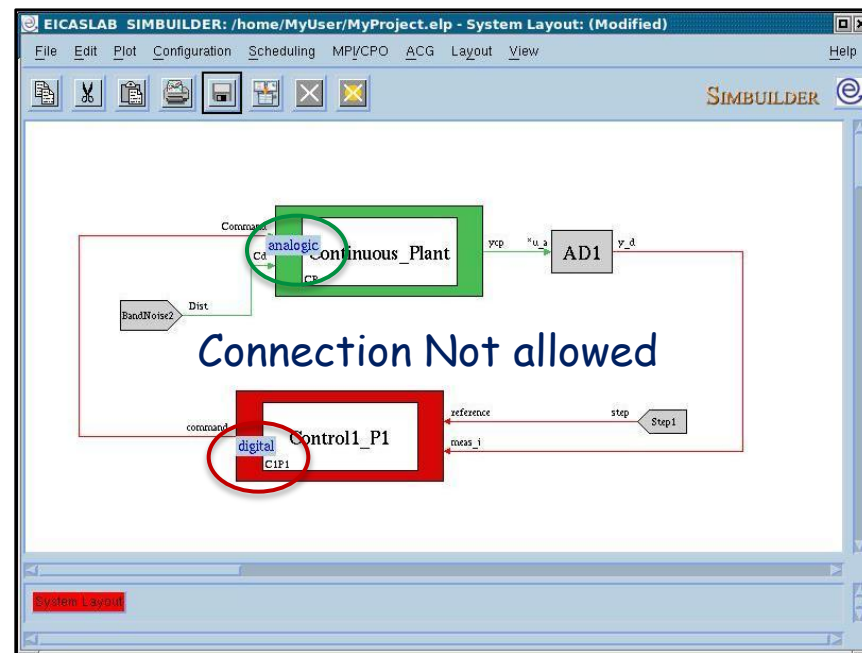
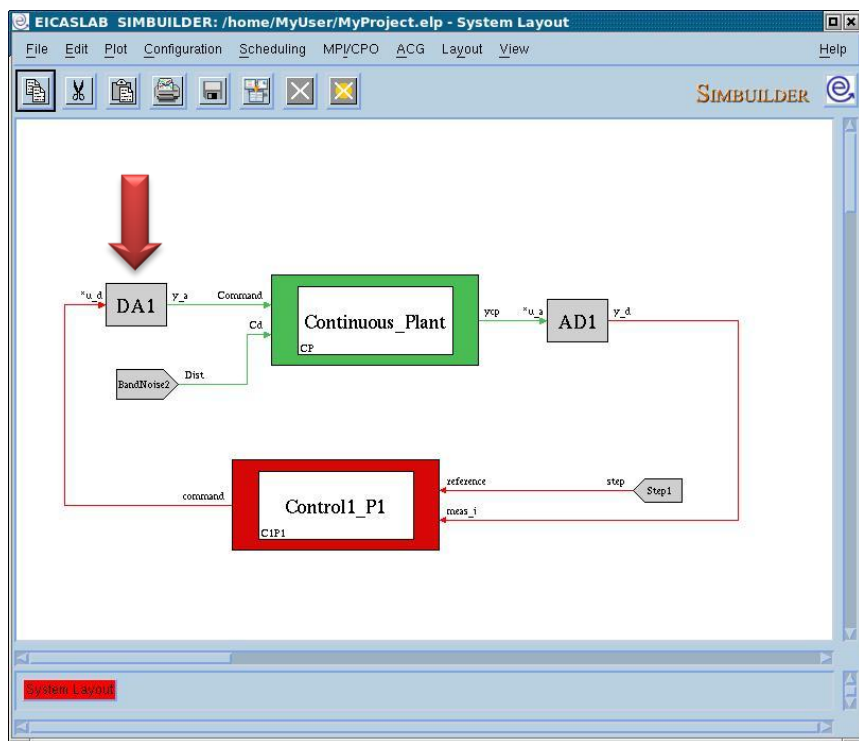
The converters need to know it for working: it is their fundamental parameter.



The A/D and D/A connections in EICASLAB

In EICASLAB the **Continuous Plant** block works on analogical data while all the other blocks use digital data.

To connect a Continuous Plant to any other block it is necessary to insert between them the A/D and D/A converter blocks that translate the analogical signals belonging to the Continuous Plant (analogical data) into discrete signals (discrete data) and vice versa.



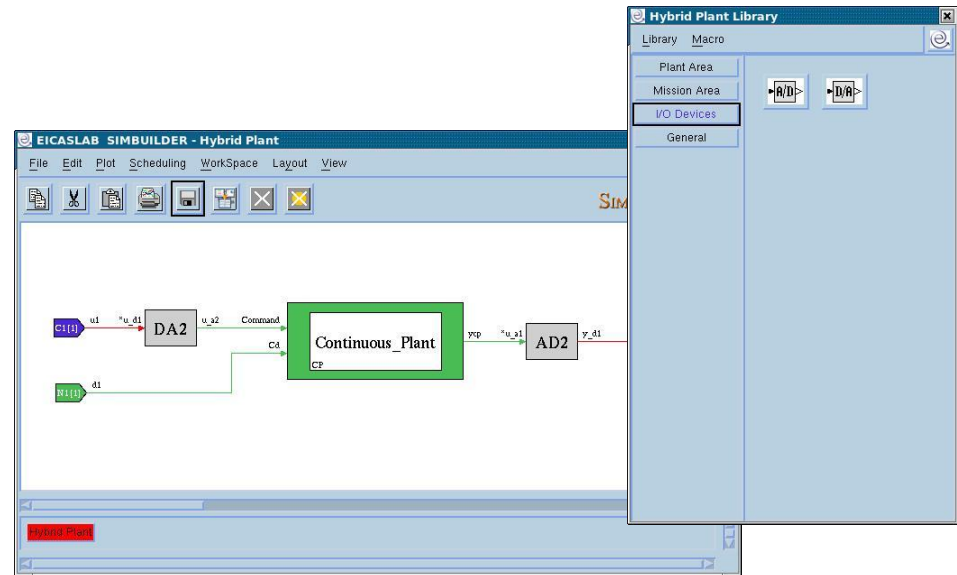
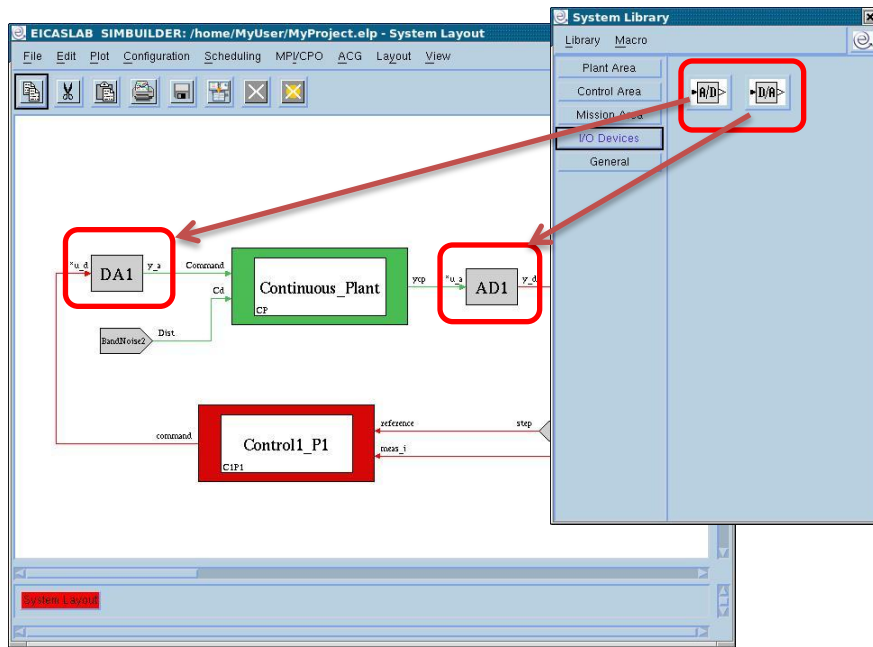
Welcome to Innovation



The A/D and D/A converter blocks in EICASLAB

The converters are blocks belonging to the ***I/O Devices*** library.
They can be inserted in:

- the System Layout
(they are available in the *I/O Devices* library of the System Library window)
- the Hybrid Plant Layout
(they are available in the *I/O Devices* library of the Hybrid Library window)





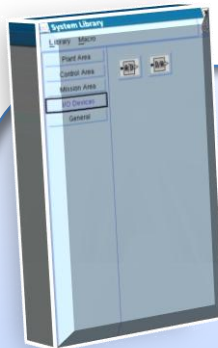
Welcome to Innovation



The Programming modes of the A/D and D/A converters

You can program the A/D and D/A converters in two ways:

-  using predefined models that you can customize setting suitable parameters (**Library Converters**),
-  using the ANSI C language (**ANSI C Converters**).



Library Converters in EICASLAB



Welcome to Innovation



The Library Converters Associated popup menu

When you insert a converter in your project it is, by default, a Library Converter.

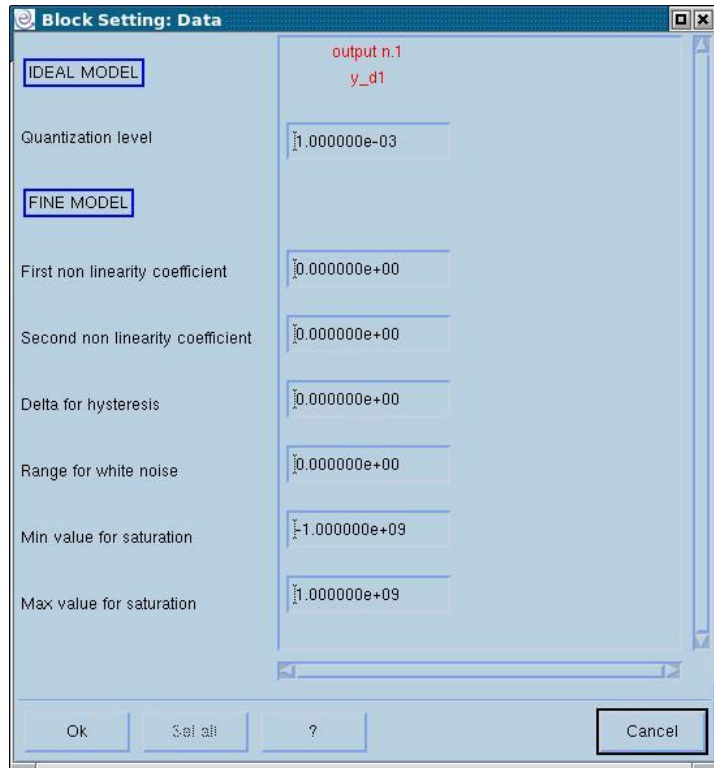
The screenshot illustrates the software interface for configuring a converter block. On the left, a block labeled 'AD' is shown with input 'u_d1' and output 'y_d1'. A context menu is open over the block, listing options: Programming Mode, Block Setting, SIM Plotting, POST Recording, Copy, Cut, Delete, Paste, Rotate, and Help. The 'Block Setting' option is selected, opening a 'Block Setting: Data' dialog box. This dialog has two tabs: 'IDEAL MODEL' and 'FINE MODEL'. The 'FINE MODEL' tab is active, showing parameters for quantization level, non-linearity coefficients, hysteresis, and saturation limits. The 'Quantization level' is set to 1.000000e-03. The 'Block Setting: Name' dialog is also open, showing the block name 'AD1'.

Welcome to Innovation



The Library Converters

The pre-defined model for A/D converters (1)



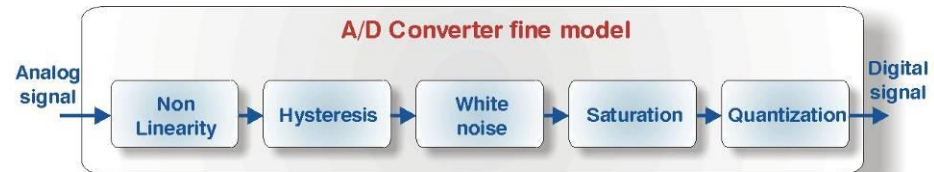
Ideal Model:

characterised only by the quantization level.

Fine Model:

models the measurement errors:

- non linearity,
- hysteresis,
- white noise,
- saturation.





The Library Converters

The pre-defined model for A/D converters (2)

Quantization:

divides the measure for the quantization value.

Non Linearity:

it adds to the measure 'y' the term ' $a \sin(y b)$ ' where 'a' and 'b' are respectively the *first and the second non linear coefficients*;

Hysteresis:

it models the behaviour of incremental encoders. It works by verifying the difference between the new measure and the old one, decreasing/increasing the new measure if the difference is greater/minor than a factor *delta*;

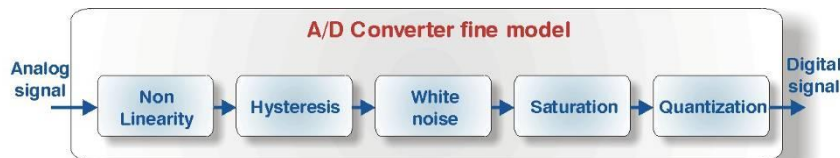
White noise:

it adds to the measure a signal described by an uniform distribution with a specific *range*;

Saturation:

applies an upper and a lower limit to the measure;

Welcome to Innovation





The Library Converters

The pre-defined model for A/D converters (3)



An A/D and D/A converter block can have more inputs and outputs.

For complying the pre-defined model the number of outputs must be equal to the number of inputs.

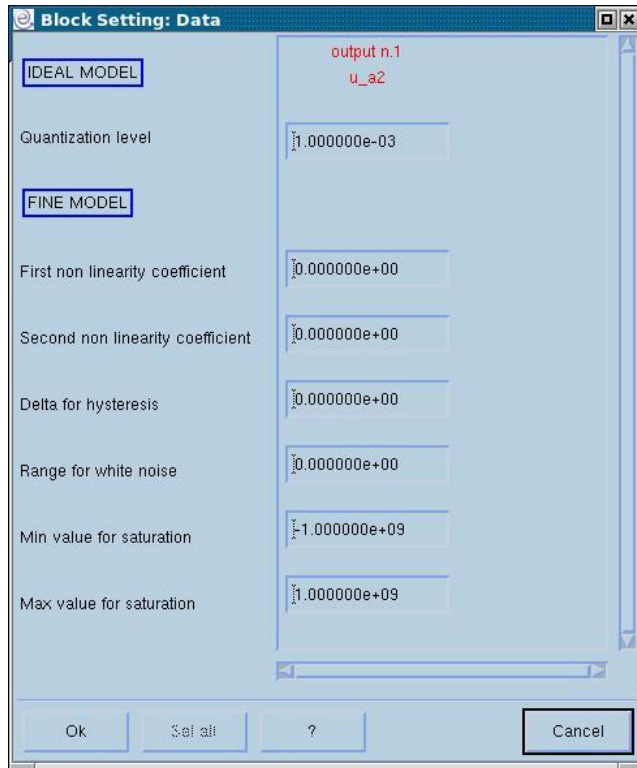
Every input/output can be customized having its own model parameters.

Parameter	output n.1	output n.2	output n.3
Quantization level	1.000000e-03	1.000000e-03	1.000000e-03
First non linearity coefficient			
Second non linearity coefficient			
Delta for hysteresis			
Range for white noise			
Min value for saturation	1.000000e+09	1.000000e+09	1.000000e+09
Max value for saturation	1.000000e+09	1.000000e+09	1.000000e+09



The Library Converters

The pre-defined model for D/A converters



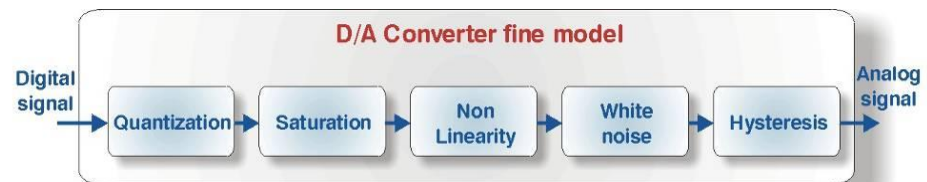
Ideal Model:

characterised only by a quantization level.

Fine Model:

considers also other phenomena:

- saturation.
- non linearity,
- white noise,
- hysteresis.





The Library Converters

The Input/Output variables (1)

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

For complying the pre-defined model the number of outputs must be equal to the number of outputs: if you add an input a corresponding output is automatically added.

The screenshot shows the SIMBUILDER interface with a block diagram. A block labeled 'ous_Plant' is highlighted in green. It has two input ports: 'u1' and '*u_d1'. The 'u1' port is connected to a signal 'u1' from a 'C1(t)' block. The '*u_d1' port is connected to a signal '*u_d1' from a 'DA2' block. The 'ous_Plant' block has two output ports: 'y_d1' and 'y_d2'. The 'y_d1' port is connected to a signal 'y_d1' from an 'AD2' block. The 'y_d2' port is connected to a signal 'y_d2' from an 'AD2' block. Below the block diagram, two windows are open: 'Block Setting: Input/Output' and 'Variable Characteristics'. The 'Block Setting: Input/Output' window shows a list of input/output variables with 'Add', 'Del', and 'Set' buttons. The 'Variable Characteristics' window shows the configuration for a variable named '*w2' with type 'EL_DOUBLE', dimension '1', and comment 'Angular rate'.

The input/output variables are ANSI C variables .



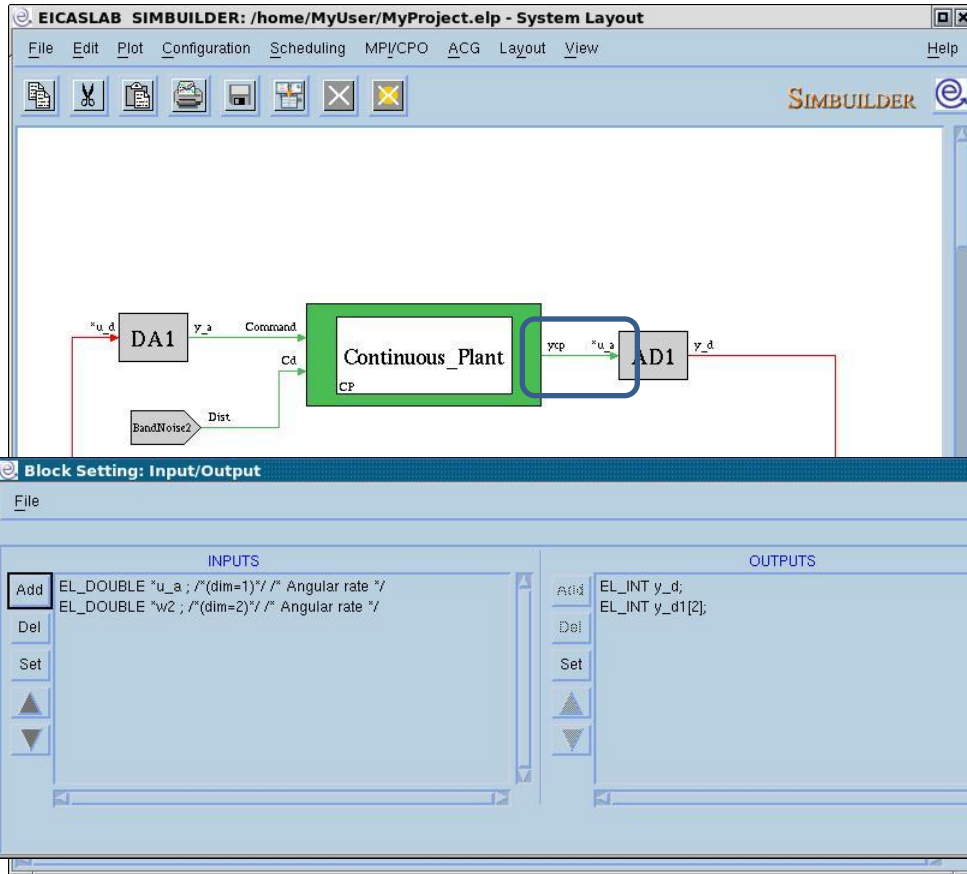
The Library Converters

The Input/Output variables (2)

The converters work directly on the output variables of the blocks to which they are connected.

Any quantity has to be measured when it is available (every measure corresponds to a precise measuring time).

The input variable is then a pointer to the output variable to which it is connected (e.g.: variable $*u_a$).

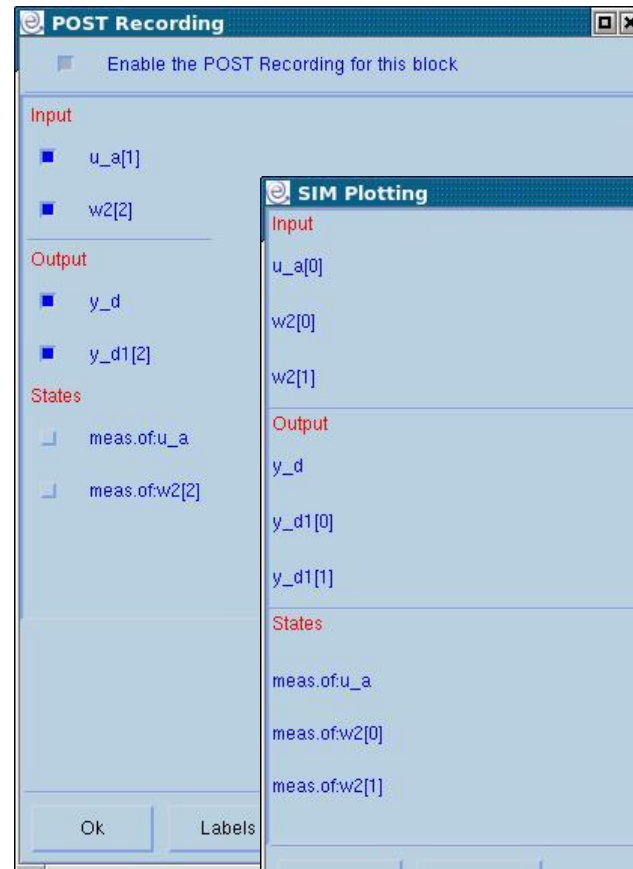
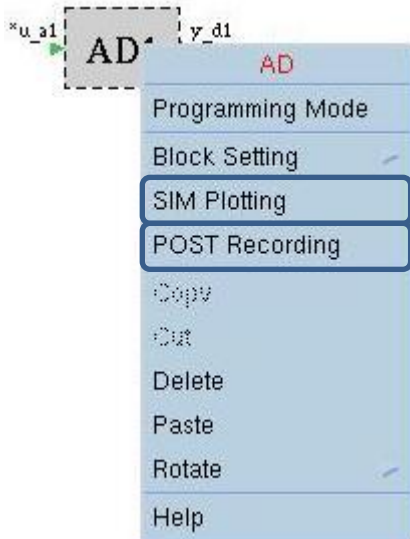




The Library Converters

SIM plotting and POST recording

You can select the input, output, and state variables for plotting and recording them during the simulations.



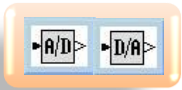
Welcome to Innovation



ANSI C Converters in EICASLAB



Welcome to Innovation



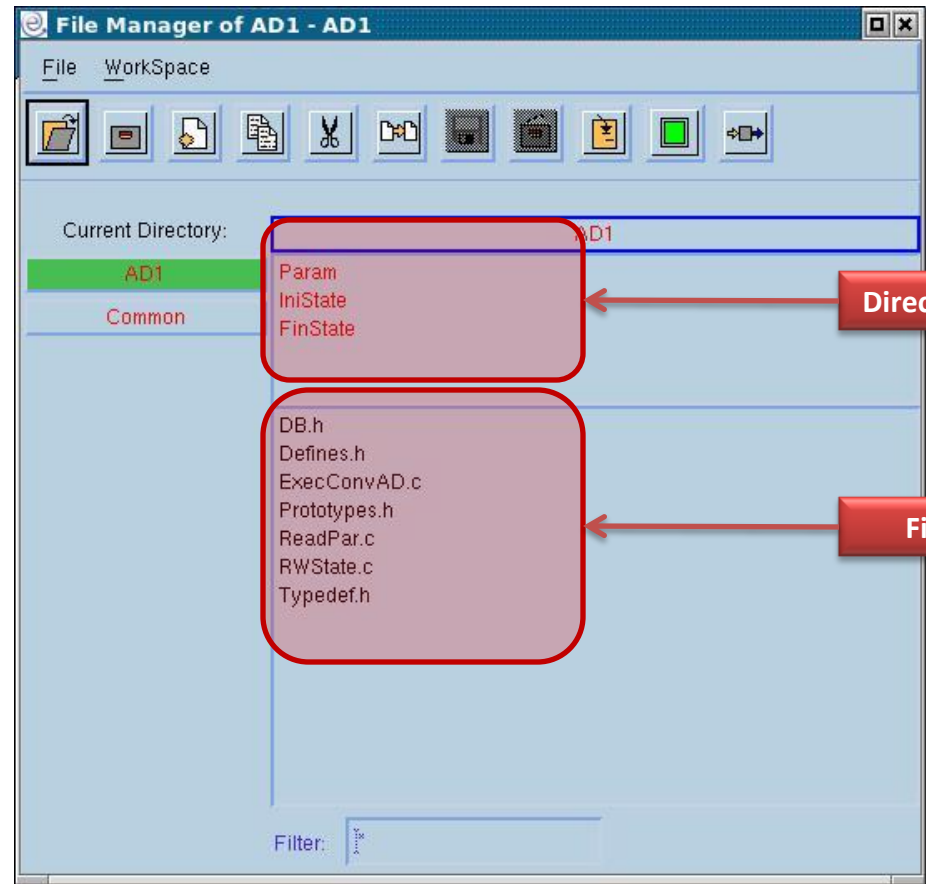
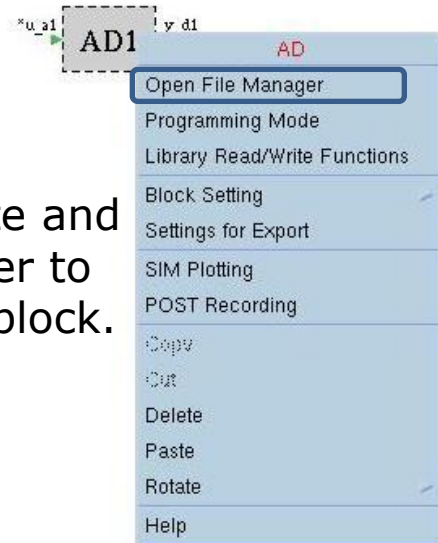
The Converter programmed with ANSI C language The Converter file manager

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

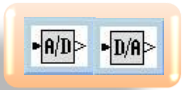
EICASLAB provides a pre-organised structure: a set of template files subdivided in:

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

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



Welcome to Innovation

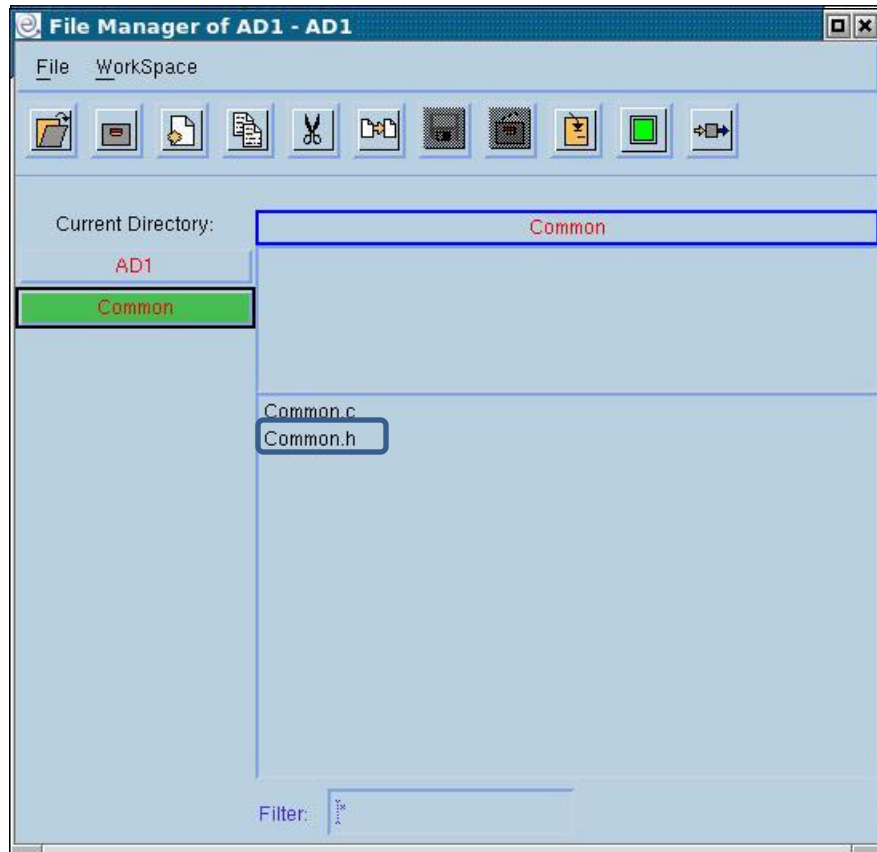


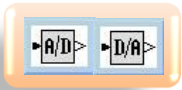
The Converter programmed with ANSI C language

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 ANSI C



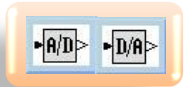


The Converter programmed with ANSI C language

Initialization functions

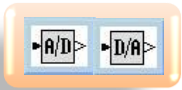
Name	Description	ANSI C File	Data File
AD#_ReadPar	Parameter file reading	ReadPar.c	ConvAD.par
AD#_ReadState	Initial state file reading	RWSate.c	ConvA_inistate
AD#_Ini	User initialisation function	ExecConvAD.c	---

The Converter programmed with ANSI C language



Execution functions

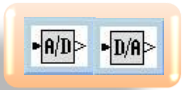
Name	Description	C File
AD#_Exe	Executes all the operations that the activity must perform each time it is scheduled and updates its outputs	ExecConvAD.c



The Converter programmed with ANSI C language

Final functions

Name	Description	C File	Data File
AD#_Fin	User final function	ExecConvAD.c	---
AD#_WriteState	Final state file writing	RWState.c	ConvAD.finstate



The Converter programmed with ANSI C language

Data file management

```

/*****/
void  AD@_ReadPar(FILE *fp)
/*
INPUTS:
fp.   file pointer to the file ConvAD.par

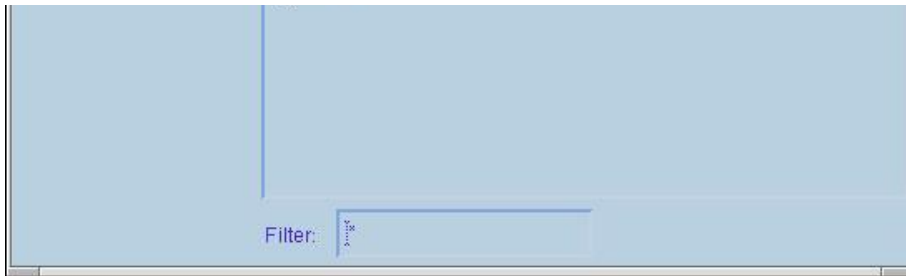
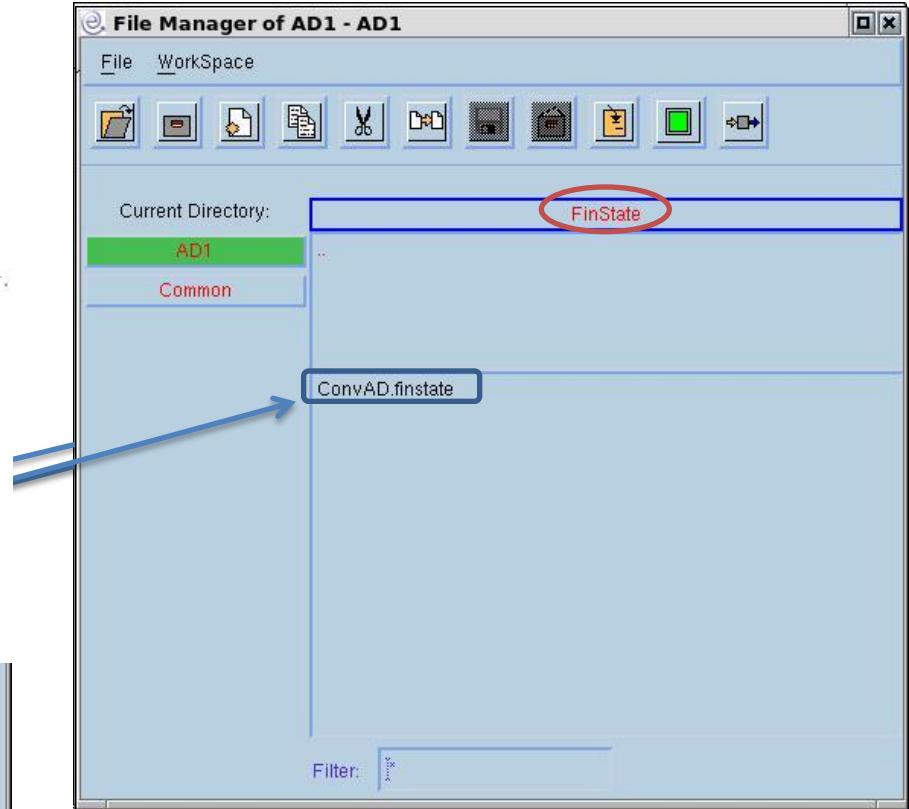
OUTPUTS:
value of the AD@ parameters

OBJECTIVES:
The function can read the parameters of the AD@ module from the file ConvAD.par.

All the parameters should be defined in:
.   DB.h.   .   database of the AD@ Module

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

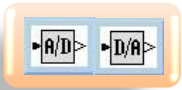
return;
}
/*****/
    
```





The Converter programmed with ANSI C language

The Library Read/Write Functions



File Structure dialog box:

- Add: scal1,scal2
- Structure: One or more scalar

Variables dialog box (top):

- Structure: One or more scalar
- Type: double

Variables dialog box (bottom):

- Structure: Array
- Type: double
- Dimensions: m[2][3]
- Show names in row: yes

Parameter list:

```
scalar parameters : scal1,scal2
1.000000e+00, 0.000000e+00,
array parameter : ar[2][3][4]
ar[0][0][0]: 0..
ar[0][0][1]: 0..
ar[0][0][2]: 0..
ar[0][0][3]: 0..
ar[0][1][0]: 0..
ar[0][1][1]: 0..
ar[0][1][2]: 0..
ar[0][1][3]: 0..
ar[0][2][0]: 0..
ar[0][2][1]: 0..
ar[0][2][2]: 0..
ar[0][2][3]: 0..
ar[1][0][0]: 0..
ar[1][0][1]: 0..
ar[1][0][2]: 0..
ar[1][0][3]: 0..
ar[1][1][0]: 0..
ar[1][1][1]: 0..
ar[1][1][2]: 0..
ar[1][1][3]: 0..
ar[1][2][0]: 0..
ar[1][2][1]: 0..
ar[1][2][2]: 0..
ar[1][2][3]: 0..
```

Welcome to Innovation



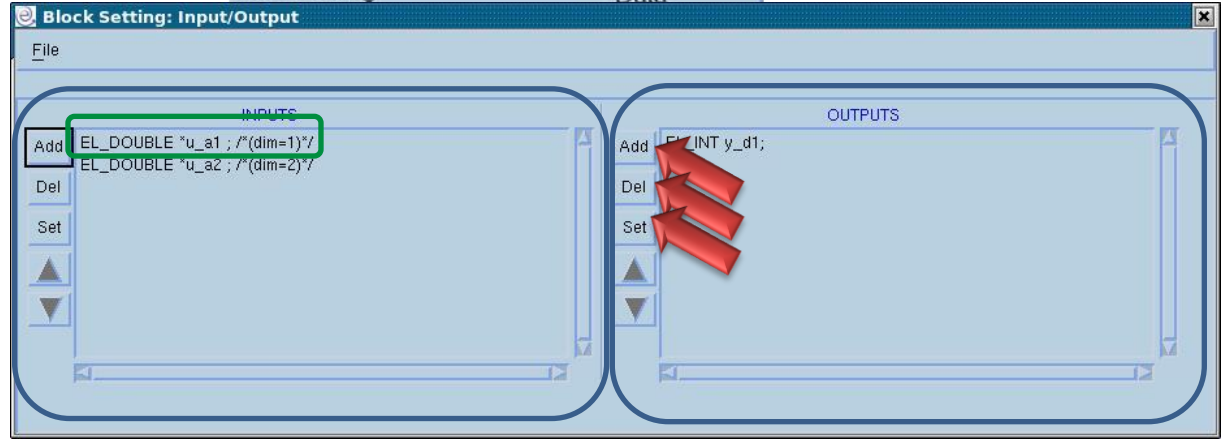
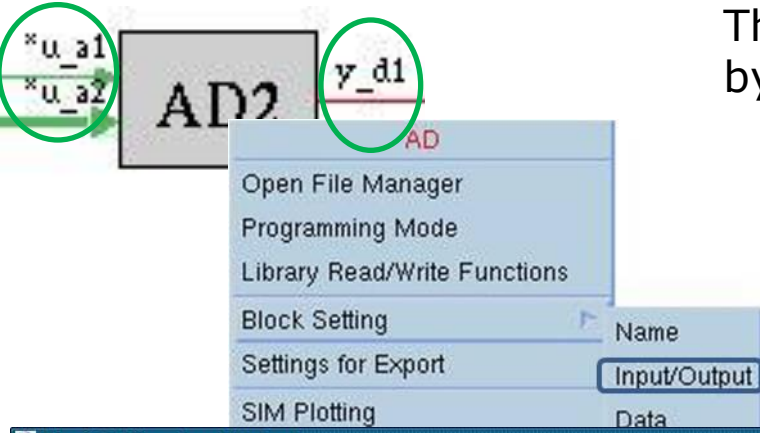
The Converter programmed with ANSI C language

The Input/Output variables

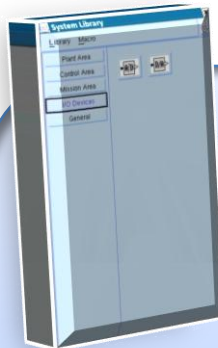
The input/output variables of the block are defined by means of an appropriate window.

The converters work directly on the output variables of the blocks to which they are connected.

The input variables are then pointers to the output variables to which they are connected .



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



The scheduling of the A/D and D/A converters in EICASLAB





The scheduling of the Converter functions

The Converter functions



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

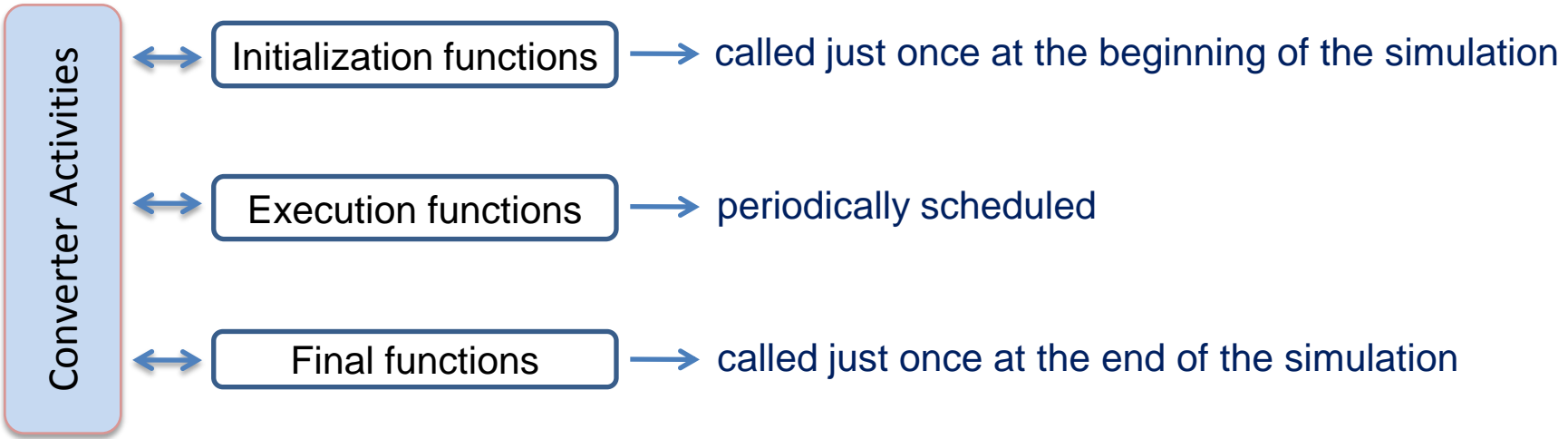
- Library Converter:**
all the functions are entirely created and managed by EICASLAB.
- Converter programmed in **ANSI C:**
all the functions have a template provided by EICASLAB and are managed by the user.

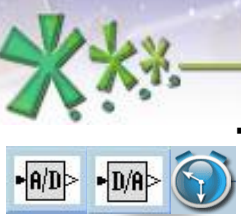


The scheduling of the Converter functions

Functions categories

The activities are subdivided in three main categories:





The scheduling of the Converter functions

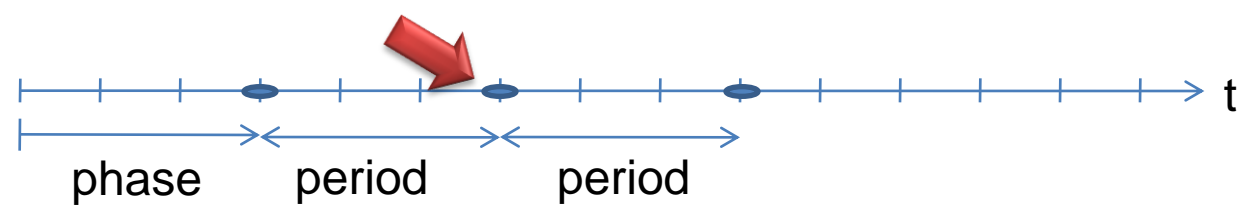
Scheduling parameters

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

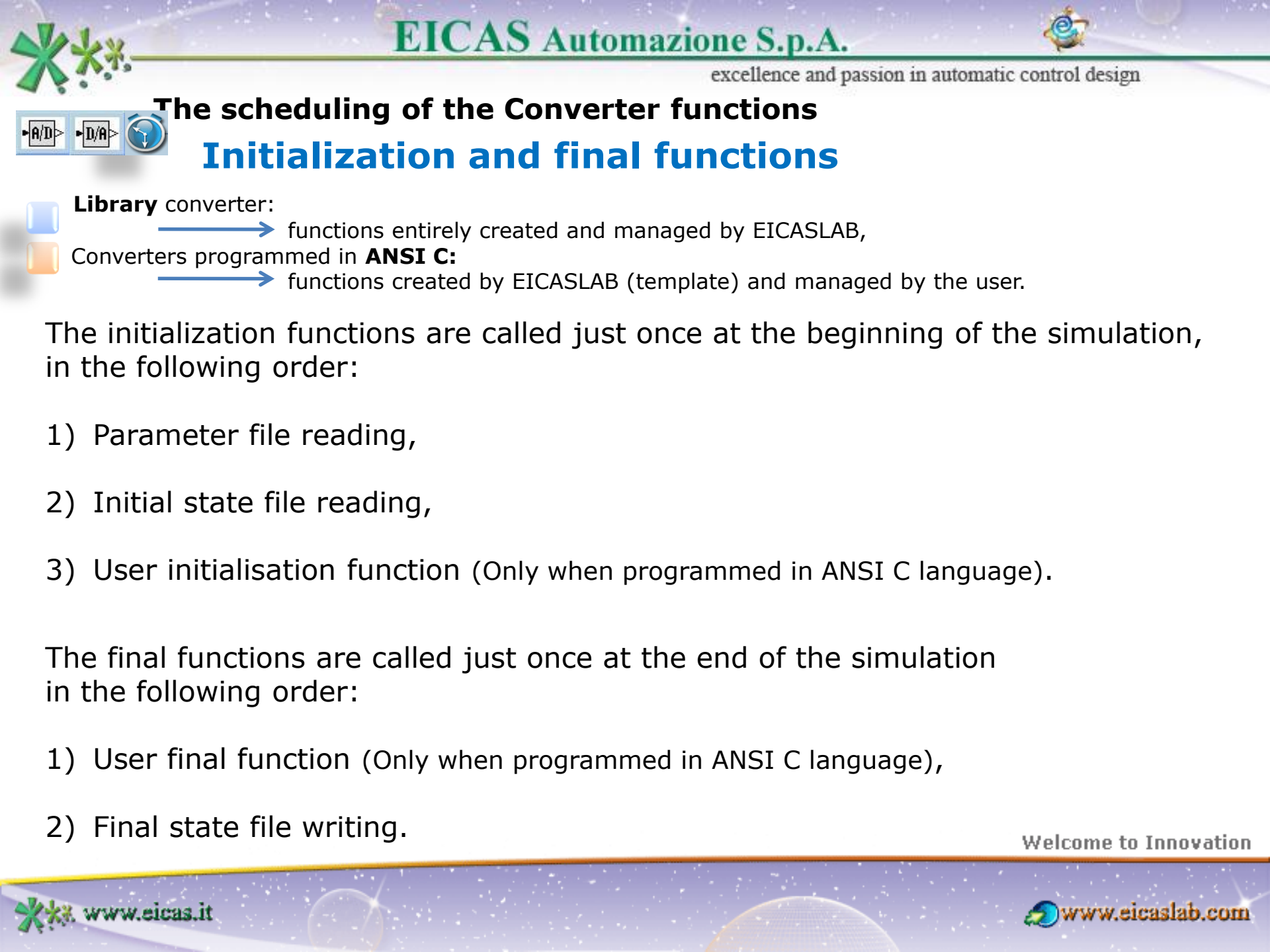
The converters are by default instantaneous activities (their duration is negligible with respect to the duration of the other activities).

Their 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.



execution function	executes all the operations that the activity must perform each time it is scheduled and updates its outputs	called when the block is scheduled (considering its phase and its period)
---------------------------	--	---



The scheduling of the Converter functions

Initialization and final functions



Library converter:

→ functions entirely created and managed by EICASLAB,



Converters programmed in **ANSI C**:

→ functions created by EICASLAB (template) and managed by the user.

The initialization 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 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 Converters

How to set the scheduling

The screenshot displays the EICASLAB SIMBUILDER interface. The main window shows a 'System Layout' with the following components and connections:

- Plant Mission (M1)**: An orange box representing the mission profile.
- Control1 P1 (CIP1)**: A red box representing the control system, receiving 'Cmd.' from the mission and providing 'Ref.' and 'Meas.' feedback.
- DA1**: A digital-to-analog converter block receiving 'u_d1' and outputting 'u_a2'.
- Continuous Plant (CP)**: A green box representing the plant, receiving 'ucp' and 'Dist' signals and outputting 'Distm'.

The 'Activities Scheduling' window is open, showing a timeline for the simulation. The overall period is 50 simulation steps. The scheduling table is as follows:

Category	Function	Execution flags	Period	Duration	Phase
Continuous Plant	CP	On	1	1	0
AD converters	AD1	On	1	NA	0
DA converters	DA1	On	50	NA	0
Missions	Step1	On	50	NA	0
Missions	BandNoise2	On	1	NA	0
Processor n.1	CIP1	On	50	50	0



EICASLAB™

*The Professional Software Suite
for Automatic Control Design
and Forecasting*



for Linux



& for Windows



www.eicaslab.com

Welcome to Innovation