Programming in ANSI C language in EICASLAB™

EICAS Automazione S.p.A.
excellence and passion in automatic control design
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

Introduction

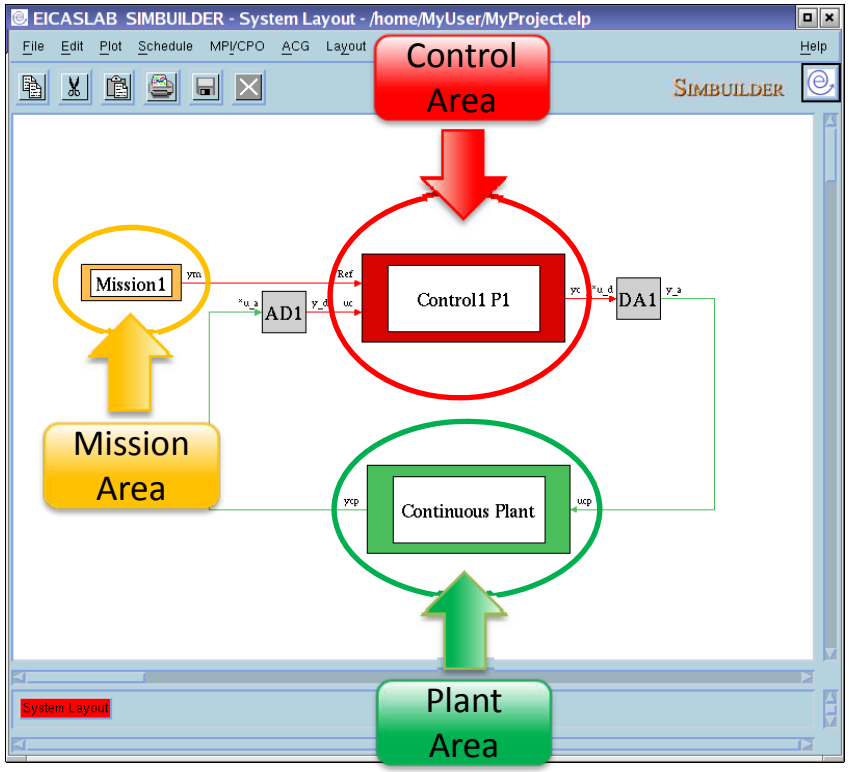
EICASLAB™ allows to develop embedded control system architectures at different hierarchical levels offering a pre-organized environment that supports the control designer in all the design steps.

During the programming phase, three main areas are available in the SIMBUILDER:

- the Plant Area
- the Control Area
- the Mission Area

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

Special attention is given to support the designer during the programming phase of the different areas, going from the possibility to use a graphical high level language to the possibility to directly program in standard ANSI C (or to combine both programming modes).



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

How to program in ANSI C in EICASLAB™

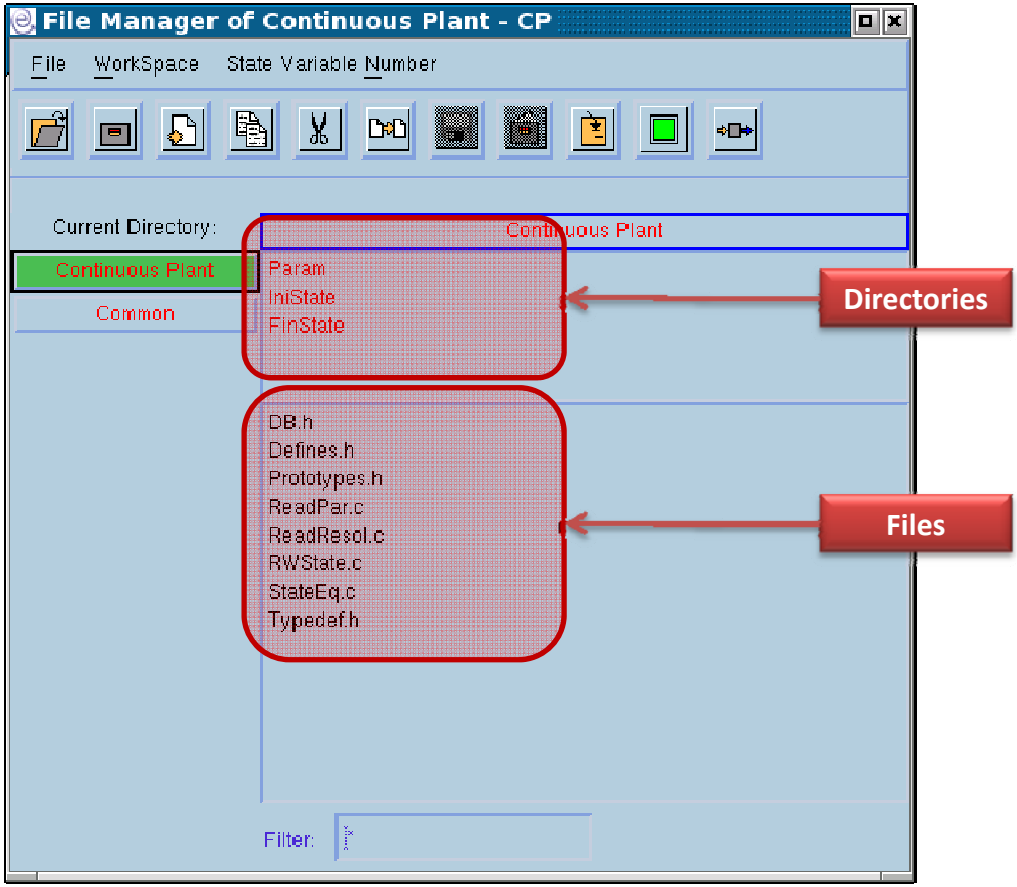
EICASLAB™ allows an easy programming in ANSI C by means of a pre-organized structure that allows you to focus just on specific and crucial aspects of the system to be programmed, being relieved from all the other aspects that are automatically managed by EICASLAB.

The pre-organised structure is **open** and **customizable**:

- you have all the potentialities of the standard ANSI C language,
- you can complete the pre-organised structure adding personal files, folder or libraries,
- you can exploit the pre-organised structure totally or partially.

Pre-organised structure for C programming

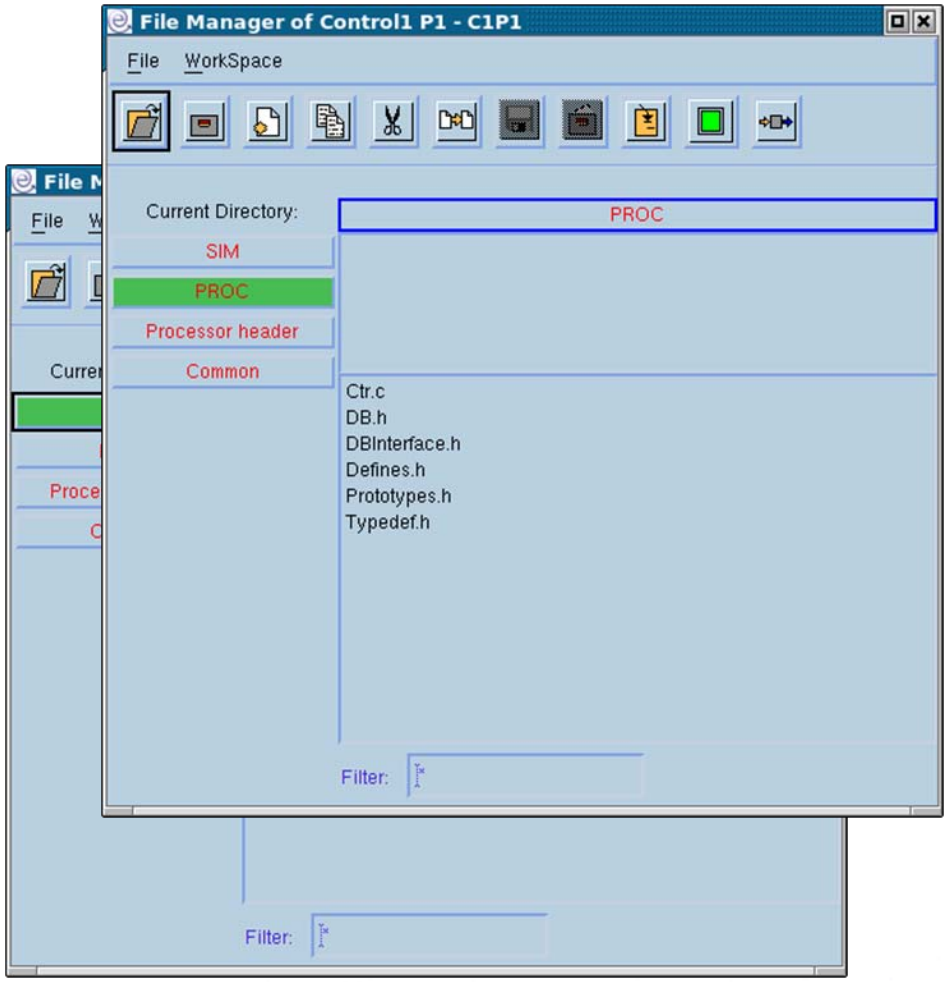
The EICASLAB™ file manager



Every block programmed in ANSI C has its own file manager through which it is possible to see all the pre-organised structure
and to program the block.

Pre-organised structure for C programming

Customization for Programming Areas



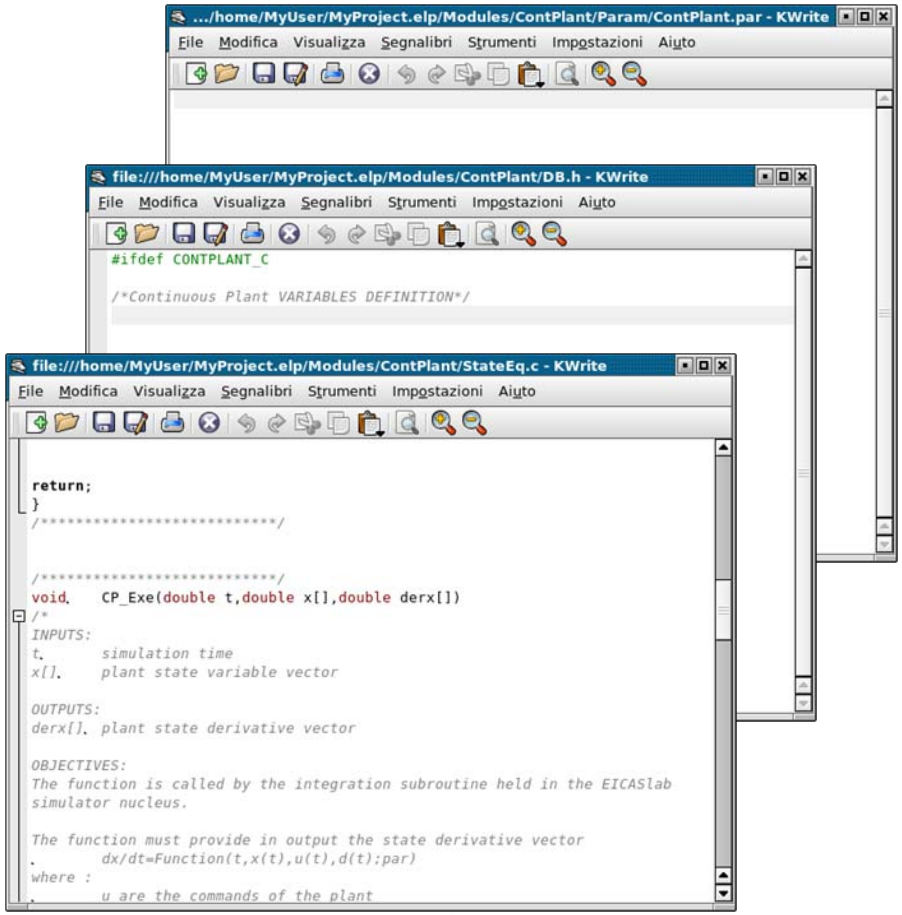
Every specific block of the Plant Area, Control Area and Mission Area has a particular and customized file organization.

For the control blocks the files are separated in order to clearly identify:

- the files only useful for simulation purposes,
- the files for control algorithm: destinated to the target (the Application Software to be transferred to the final target).

Pre-organised structure for C programming

Templates



For every block of your project programmed with standard ANSI C code EICASLAB provides a set of template files subdivided in:

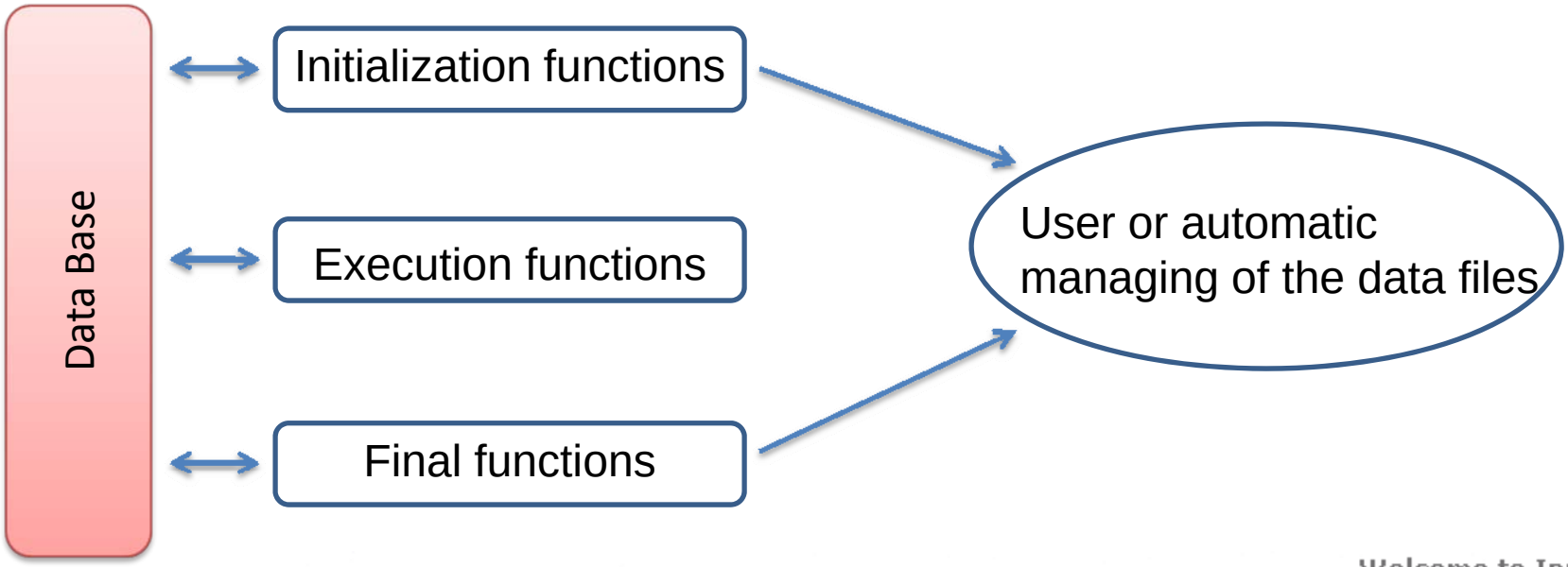
- data files,
- header files,
- C files.

You can write and customize these files in order to implement your block.

Pre-organised structure for C programming
Functional organization

The templates are organized in a functional way.
The C files contain functions devoted to a specific task, having at disposal:

- pre-configured data files,
- header files.



Pre-organised structure for C programming

Data Base: Header files

Every block has its own header files included by all its C files.

```
/*STANDARD LIBRARIES*/
#include <Standard.h>

/*COMMON INCLUDE*/
#include <GPDefines.h>
#include <GPTypedef.h>
#include <GPLibraries.h>
#include <PlantLibraries.h>
#include <Common.h>
#include <WSPLArea.h>

/*PLANT MODULE INCLUDE*/
#include "Defines.h"
#include "Typedef.h"
#include "DBInterface.h"
#include "DB.h"
#include "Prototypes.h"
```

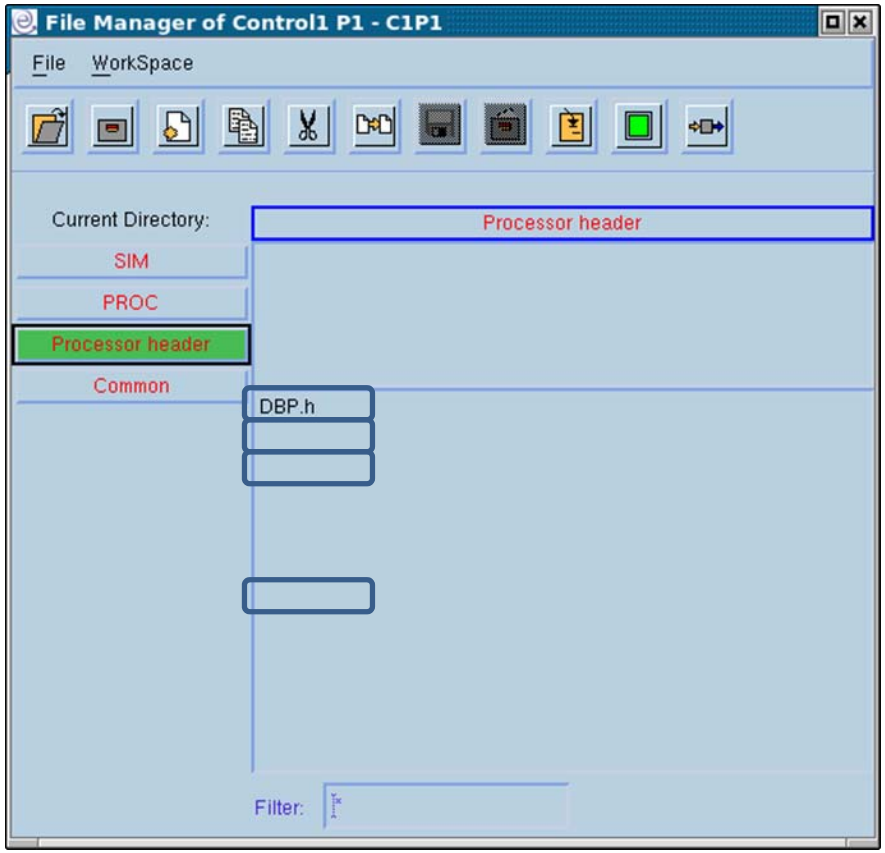
```
#include <stdio.h>
#ifdef Linux
#include <stdarg.h>
#endif
#include <math.h>
#include <signal.h>
#include <sys/time.h>
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <stdlib.h>
#include <ctype.h>
#include <float.h>
#include <sys/types.h>
#include <sys/stat.h>
```

All the header files are generated by EICASLAB, there are:

- header files available through the file manager,
- header files not directly available:
  - defined by the user through a window,
  - fixed, that cannot be modified

Pre-organised structure for C programming

Data Base: User Header files

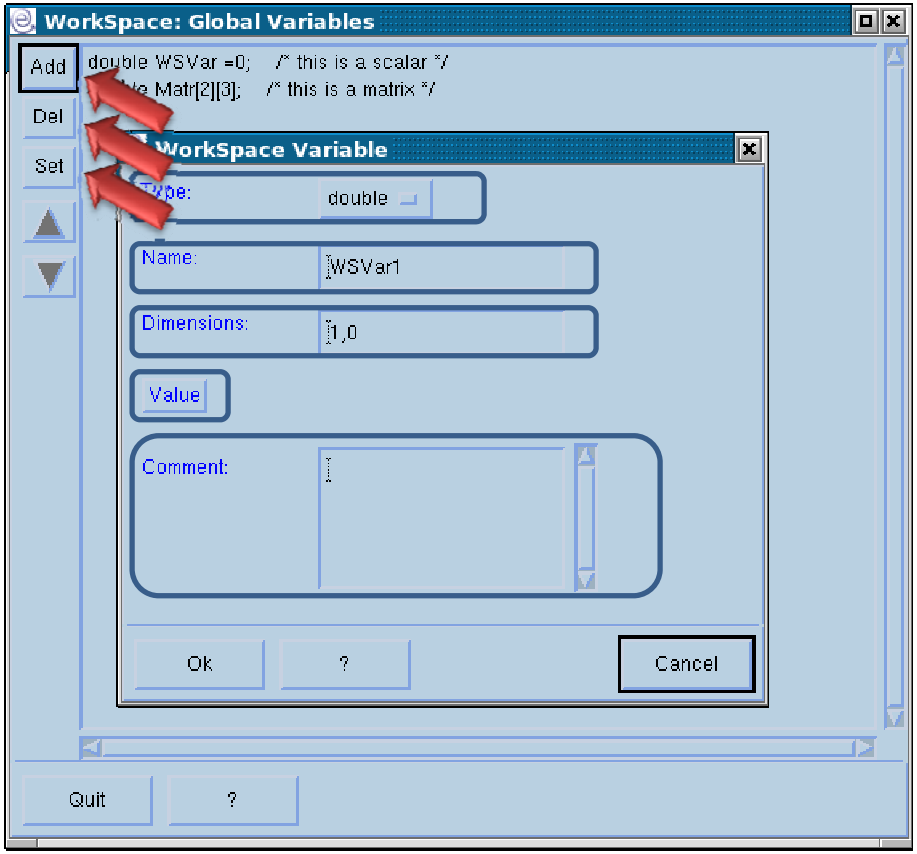


Header files of the pre-organised structure that are written by the user.

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
DBP.h	Available for all the controls of a processor

Pre-organised structure for C programming

Data Base: WorkSpace



The WorkSpace contains a set of global C variables, defined by the user

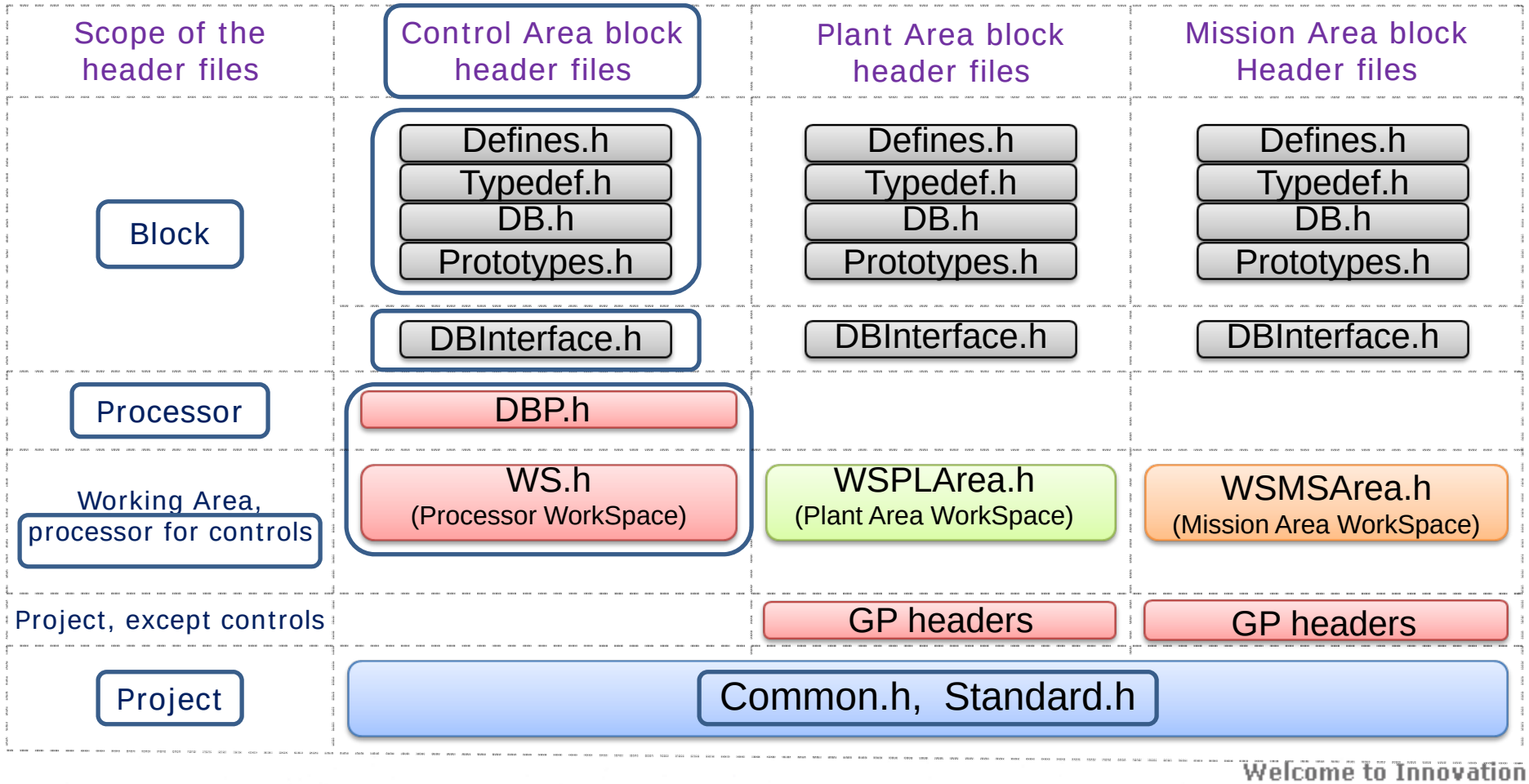
In EICASLAB there are the following WorkSpaces:

- the Plant Area WorkSpace
- the Mission Area WorkSpace
- One WorkSpace for every processor (of the Control Area)

WS.h
(Processor WorkSpace)

Pre-organised structure for C programming

Data Base: Header files scope



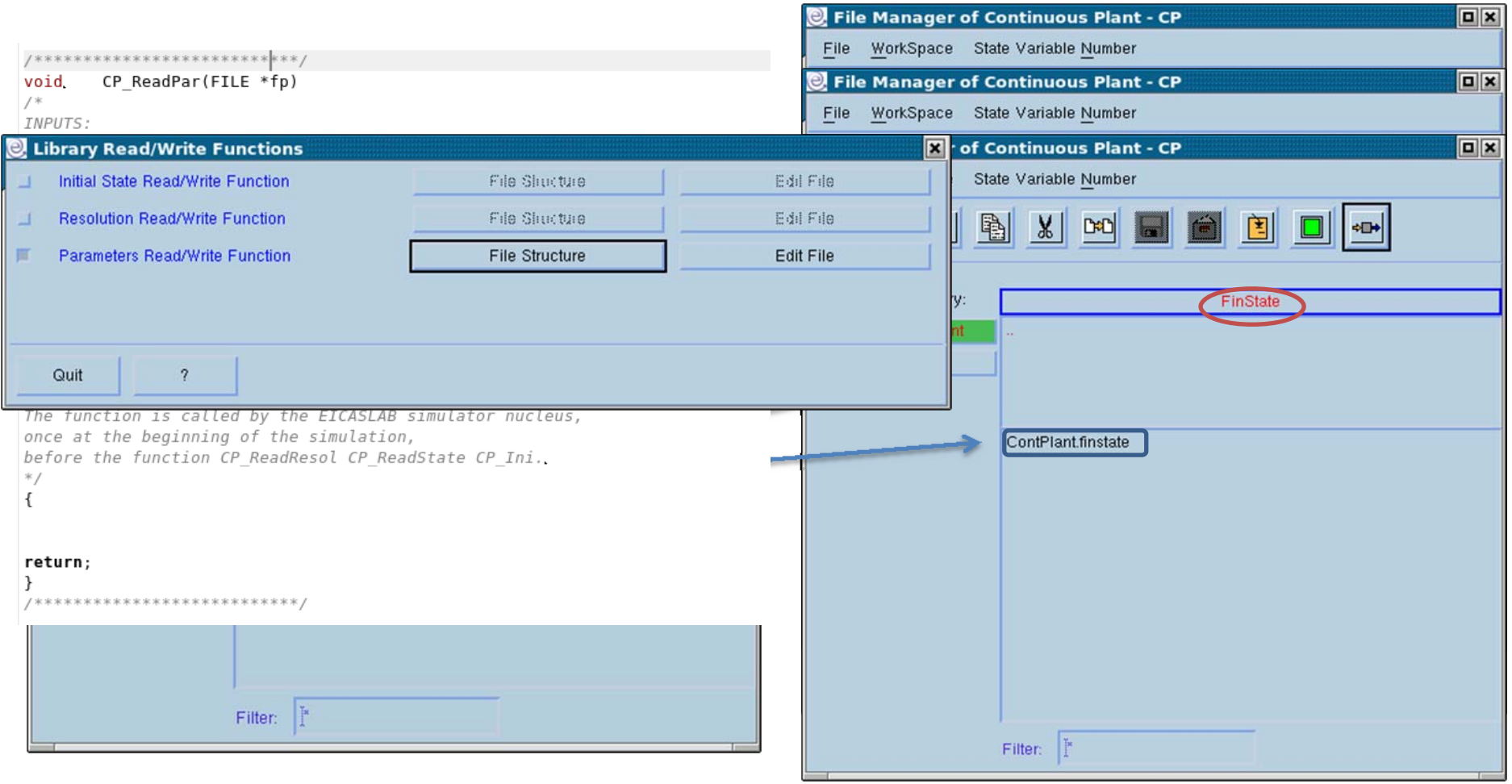
Pre-organised structure for C programming

Initialization functions

Function description	Function suffix	C File	Data File	Plant Area	Control Area	Mission Area
Parameter file reading	ReadPar	ReadPar.c	<block name>.par	✖	✖	✖
Initial state file reading	Read State	RW ate.c	<block na e>.inistate	✖	✖	✖
Resolution file reading	ReadResol	ReadResol.c	Resolution.par	Only Continuous Plant		
Control design	Des	CtrDes.c	---		✖	
User initialisation function	Ini	Specific C file of the block	---	✖	✖	✖

Pre-organised structure for C programming

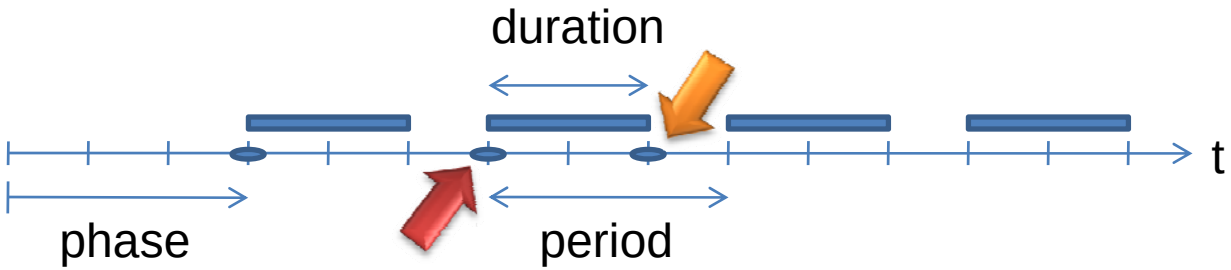
Data Base for the reading functions



Pre-organised structure for C programming
Execution functions

EICASLAB performs a like real time simulation of your project:
to perform such a simulation the user has to provide the following scheduling parameters for every block of its project:

- Phase — time at which the block begins to work,
- Period — all the blocks are periodic,
- Duration — duration of the periodic activity of the block.



To guarantee the correct scheduling of the block it is necessary to take into account its duration:
this is done by writing two functions for the periodic activity of the block:

execution function	executes all the operations that the block must perform each time it is scheduled	called when the block is scheduled (considering its phase and its period)
output function	computes the outputs of the block as a function of its current state	called after the fixed duration

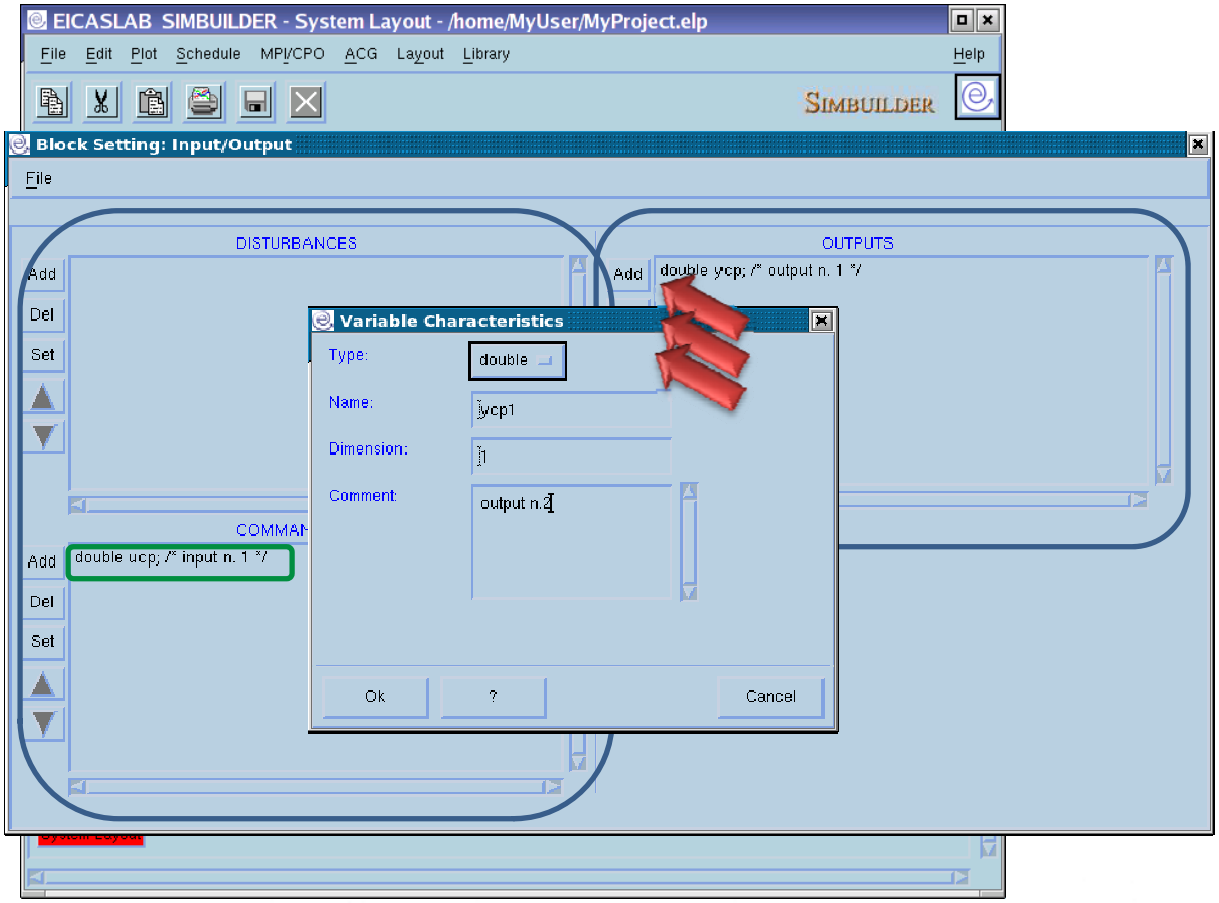
Pre-organised structure for C programming

Final functions

Function description	Function suffix	C File	Data File	Plant Area	Control Area	Mission Area
User final function	Fin	Specific C file of the block	---	✖	✖	✖
Final state file writing	WriteState	RWState.c	<block name>.finstate	✖	✖	✖

Pre-organised structure for C programming

Interface between the blocks programmed in C and the rest of the project



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

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

Pre-organised structure for C programming

Benefits of the pre-organised structure

No need to manually create all the base structure necessary for a good ANSI C programming.

Automatic standard managing of data files (opening, reading, writing, closing).

Automatic generation of the Makefile needed for compiling your ANSI C code.

Automatic link of your blocks programmed in ANSI C with the rest of your project.

You can modify the proposed structure adding:

- new user files,
  - data files,
  - header files,
  - C files,
- new directories.

You can link your code with external libraries.

You can take full advantage of the facilities offered by EICASLAB.
You have all the potentialities of the standard ANSI C language.

Blocks programmable in ANSI C language

Plant area

- Continuous Plant
- Discrete Plant
- Experimental Data
- A/D Converter
- D/A Converter

Control area

- Control
- Processor Input/Output

Mission area

- Mission

EICASLAB™ Tools

Function names

EICASLAB™

Function names: ***<prefix (type of block)>_<suffix (type of function)>***

	Function prefix	Example
Continuous Plant	CP	CP_Exe
Discrete Plant	DP<block index >	DP1_Exe
Experimental Data	ED<block index >	ED1_Exe
A/D Converter	AD<block >	AD1_Exe
D/A Converter	DA<block index>	DA1_Exe
Control	C<control index>P<processor index>	C1P1_Exe
Processor Input	ProcIn<control index>P<processor index>	ProcIn1P1_Exe
ProcessorOutput	ProcOut<control index>P<processor index>	ProcOut1P1_Exe
Mission	M<block index>	M_Exe

Function list

Function names ***<prefix (type of block)>_<suffix (type of function)>***

Function suffix	Function description		File
Ini	User initialisation function	for Continuous Plant / Discrete Plant	StateEq.c
		for Experimental Data	Ini.c
Des	Control Design	for Controls	CtrDes.c
ReadPar	Parameter file reading		ReadPar.c
ReadState	Initial State file reading		RWState.c
ReadResol	Resolution file reading (just for Continuous Plant)		ReadResol.c
Exe	Computation of the State derivative for Continuous Plant		StateEq.c
	Computation of the next state for the DiscrPlant		
	Reading of one cycle data for the Experimental Data		Exe.c
	Post processing of the read data for the Experimental Data		PostProc.c
	Computation of the outputs of the converters		ExeConvAD.c/ExeConvDA.c
	Mission execution		Mission.c
	Control execution		Ctr.c
	Processor Input/Output execution		ProcIn.c / ProcOut.c
Out	Computation of the outputs		(same as 'Exe' functions)
Fin	User final function for Continuous Plant / Discrete Plant		StateEq.c
		for Experimental Data	Fin.c
WriteState	Final State file writing		RWState.c

EICASLAB™

The Professional Software Suite
for Automatic Control Design
and Forecasting

for Linux & for Windows

www.eicaslab.com

Welcome to Innovation