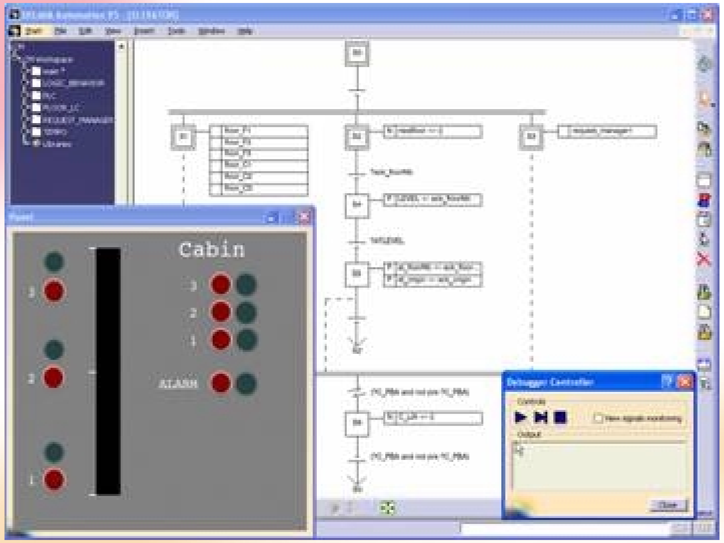
DELMIA Automation LCM Studio

POWER to PLAN is POWER to BUILD

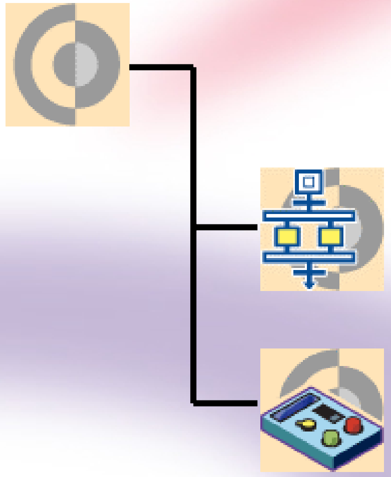
LCM Studio Workbenches

LCM Studio is composed of following workbenches.

The goal is to give the user the capability to create the Control program from scratch, to create the HMI and to validate them.



Logic Control Modeling



LCM Editor

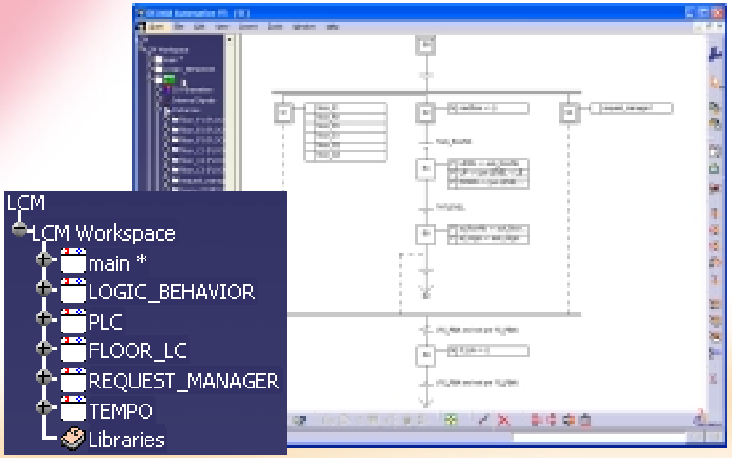
HMI Control Panel Design

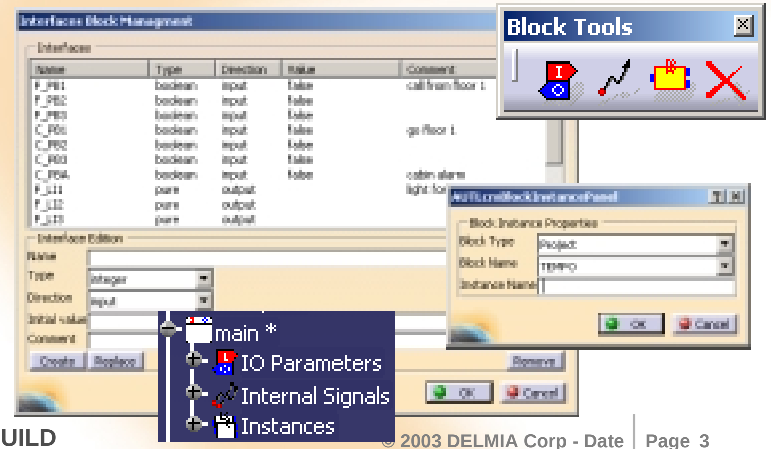
LCM Editor

Enables to structure the project into different independent blocks

- Creation of blocks to structure the project and to be able to re-use several time the same block (i.e. delay block)
  - ➔ The main block / program is represented with a star *
  - ➔ Subprograms without a star

- For each block, different signals have to be defined / declared
  - ➔ Input / Output
    - ➔ block interfaces
    - ➔ the mapping of blocks IOs enables the communication between them
    - ➔ main block: enables also the communication with an external system (i.e. the control panel)
  - ➔ Internal signal
    - ➔ signals that are only used inside the block, their value is not sent to another block
  - ➔ Instance
    - ➔ blocks that are used inside another one have to be defined


LCM
LCM Workspace
main *
LOGIC_BEHAVIOR
PLC
FLOOR_LC
REQUEST_MANAGER
TEMPO
Libraries


Interfaces Block Management

Interfaces

Name	Type	Direction	Value	Comment
F_PB1	boolean	input	false	call from floor 1
F_PB2	boolean	input	false	
F_PB3	boolean	input	false	
C_PB1	boolean	input	false	go floor 1
C_PB2	boolean	input	false	
C_PB3	boolean	input	false	
C_PB4	boolean	input	false	cabin alarm
F_LI1	pure	output		light fo
F_LI2	pure	output		
F_LI3	pure	output		

Interface Edition
Name:
Type: integer
Direction: input
Initial value:
Comment:

AUTLcmBlockInstancePanel

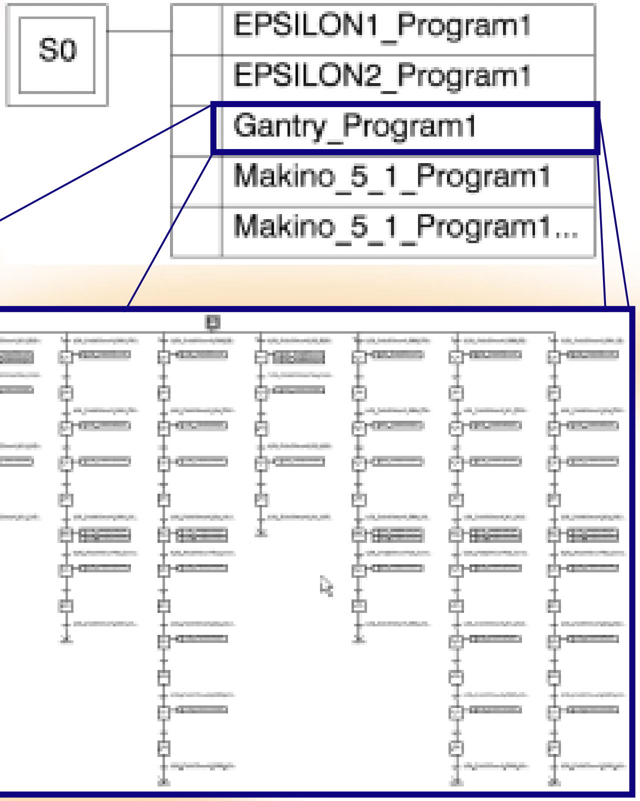
Block Instance Properties
Block Type: Project
Block Name: TEMPO
Instance Name:

main *
IO Parameters
Internal Signals
Instances

LCM Editor

Provides all of the tools required to perform PLC programming

- 4 toolbars necessary to create an entire PLC program
  - ➔ Normal steps
    - ➔ append step, add step after, add step before, add branch , add final step
  - ➔ Parallel steps
    - ➔ add parallel step after, add parallel step before, add parallel branch, restore convergence
  - ➔ Actions
    - ➔ add data flow action (i.e. modify a signal value)
    - ➔ add macro action: creates a sub-program directly inside the block, it can’t be re-used
    - ➔ add call action: runs the called instance
    - ➔ freeze action
  - ➔ Transitions


S0

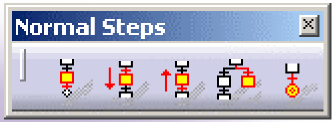
EPSILON1_Program1

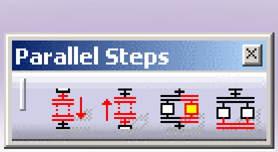
EPSILON2_Program1

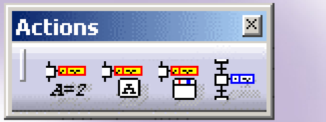
Gantry_Program1

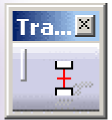
Makino_5_1_Program1

Makino_5_1_Program1...



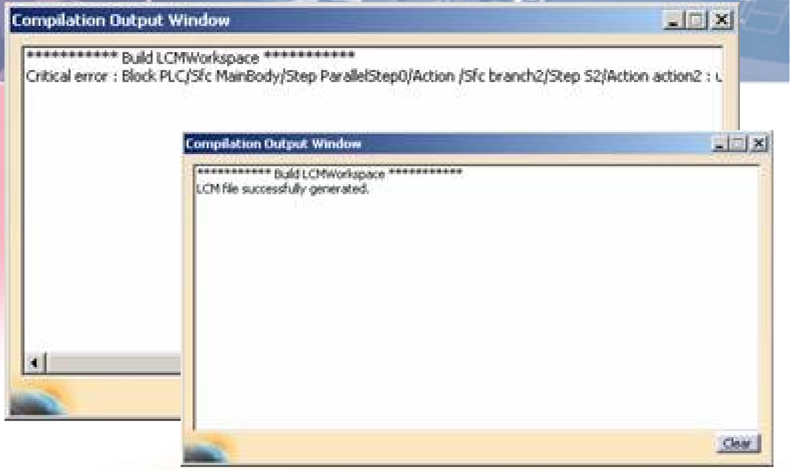






LCM Editor

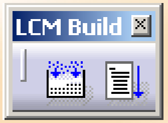
Enables the validation of the PLC program



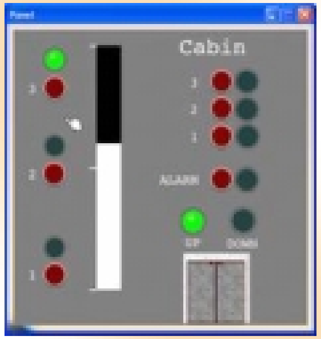
```
************ Build LCMWorkspace ************
Critical error : Block PLC/Sfc MainBody/Step ParallelStep0/Action /Sfc branch2/Step S2/Action action2 : u
```

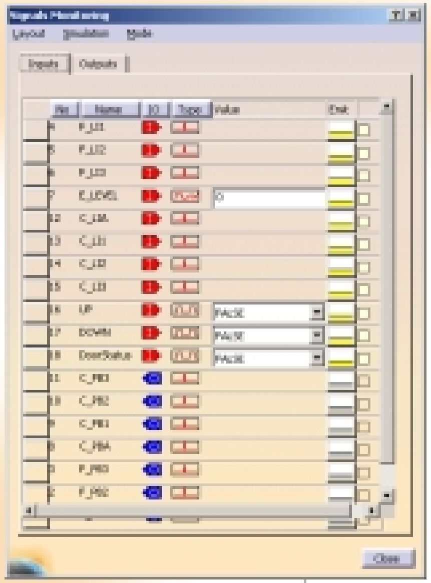
```
************ Build LCMWorkspace ************
LCM file successfully generated.
```

- LCM Compiler
  - ➔ Detects
    - ➔ syntax and semantic errors
    - ➔ instantaneous loops
    - ➔ causality errors (e.g. signal tested before its emission)
  - ➔ Schedule all parallel branches
  - ➔ Perform code optimization
  - ➔ Produce an intermediate code (LCMO)
  - ➔ Translate in C language



- 2 ways to test the PLC program
  - ➔ Ability to test the program by forcing the signals values
  - ➔ Ability to use the control panel after mapping the signals of the panel and the program
  - ➔ Behavior of the program visible in the SFC+ view (highlight of the running steps), in the signals monitoring window and if used in the control panel.



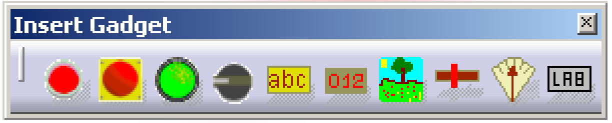


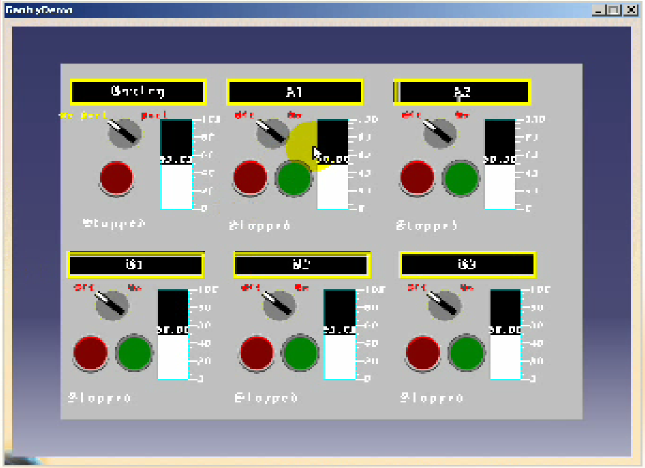
No	Name	IO	Type	Value	Dst
4	F_LI1				
5	F_LI2				
6	F_LI3				
7	E_LEVEL			0	
12	C_LIA				
13	C_LI1				
14	C_LI2				
15	C_LI3				
16	UP			FALSE	
17	DOWN			FALSE	
18	DoorStatus			FALSE	
11	C_PB3				
10	C_PB2				
9	C_PB1				
8	C_PBA				
3	F_PB3				
2	F_PB2				

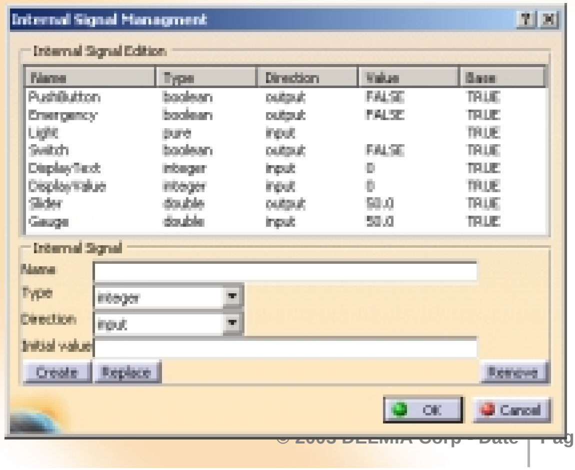
HMI Control Panel Design

Capability to create a realistic Control Panel with all real details

- Creation of the panel using pre-defined gadgets
  - ➔ Insertion of I/Os effecting gadgets
    - ➔ push button, emergency button, selector/switch, slider
  - ➔ Insertion of I/Os effected gadgets
    - ➔ light, text display, value display, gauge
  - ➔ Insertion of passive gadgets
    - ➔ image, label







Internal Signal Management

Internal Signal Edition

Name	Type	Direction	Value	Base
PushButton	boolean	output	FALSE	TRUE
Emergency	boolean	output	FALSE	TRUE
Light	pure	input		TRUE
Switch	boolean	output	FALSE	TRUE
DisplayText	integer	input	0	TRUE
DisplayValue	integer	input	0	TRUE
Slider	double	output	50.0	TRUE
Gauge	double	input	50.0	TRUE

Internal Signal

Name

Type: integer

Direction: input

Initial value

- Gadget signals automatically generated
  - ➔ Gadgets have default types and directions value

POWER to PLAN is POWER to BUILD

HMI Control Panel Design

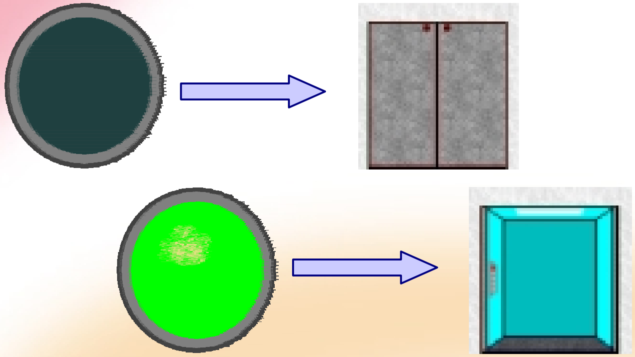
Capability to create a realistic Control Panel by formatting and customizing it

- Gadgets format and customization
  - ➔ General properties: name, position
  - ➔ Behavior properties: icons, fixed/not fixed
  - ➔ Value/graduation management
  - ➔ Background color, text properties

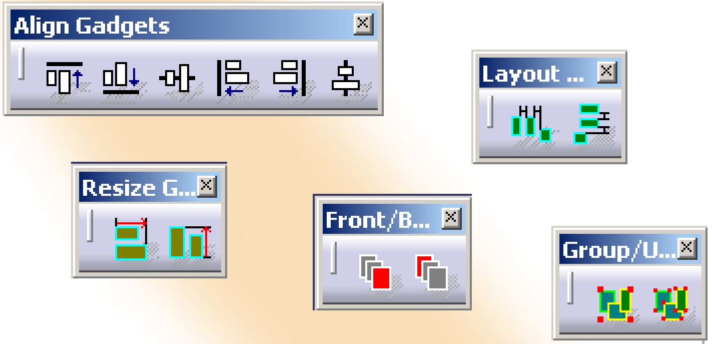
The signal needed to display the door status of an elevator is similar as for a light

- 2 values (closed / opened)
- display ➔ input signal

➔ The light gadgets is choosen, the icons light on / light off are changed



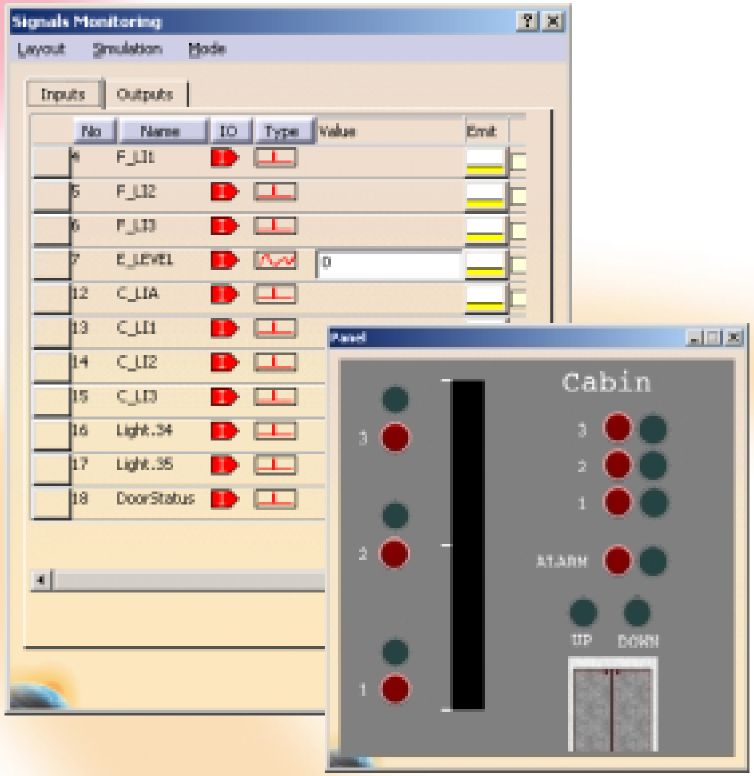
- Panel format
  - ➔ Similar tools as in PowerPoint
    - ➔ align
    - ➔ layout ⇔ distribute (PowerPoint)
    - ➔ resize tools
    - ➔ front / back ⇔ Order (PowerPoint)
    - ➔ group / ungroup



HMI Control Panel Design

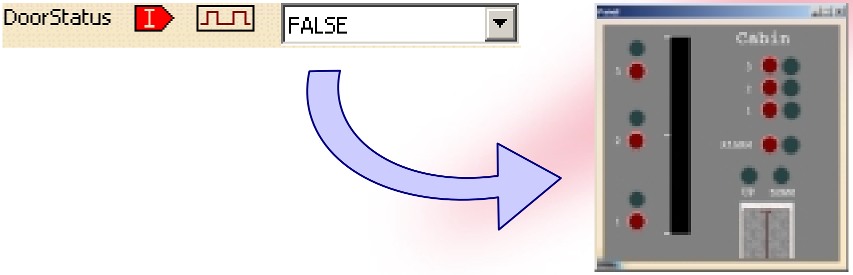
Capability to create a realistic Control Panel by testing its behavior

- Ability to test the control panel by forcing the signals values and visualizing the results in the graphical view

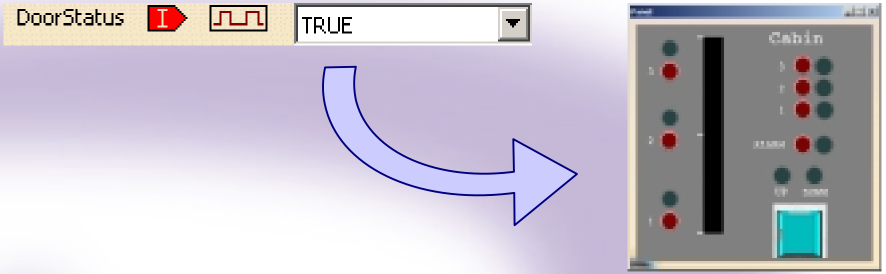


Signals Monitoring

No	Name	IO	Type	Value	Emit
4	F_LI1	I			
5	F_LI2	I			
6	F_LI3	I			
7	E_LEVEL	I		0	
12	C_LIA	I			
13	C_LI1	I			
14	C_LI2	I			
15	C_LI3	I			
16	Light.34	I			
17	Light.35	I			
18	DoorStatus	I			



DoorStatus I FALSE



DoorStatus I TRUE

Added-values

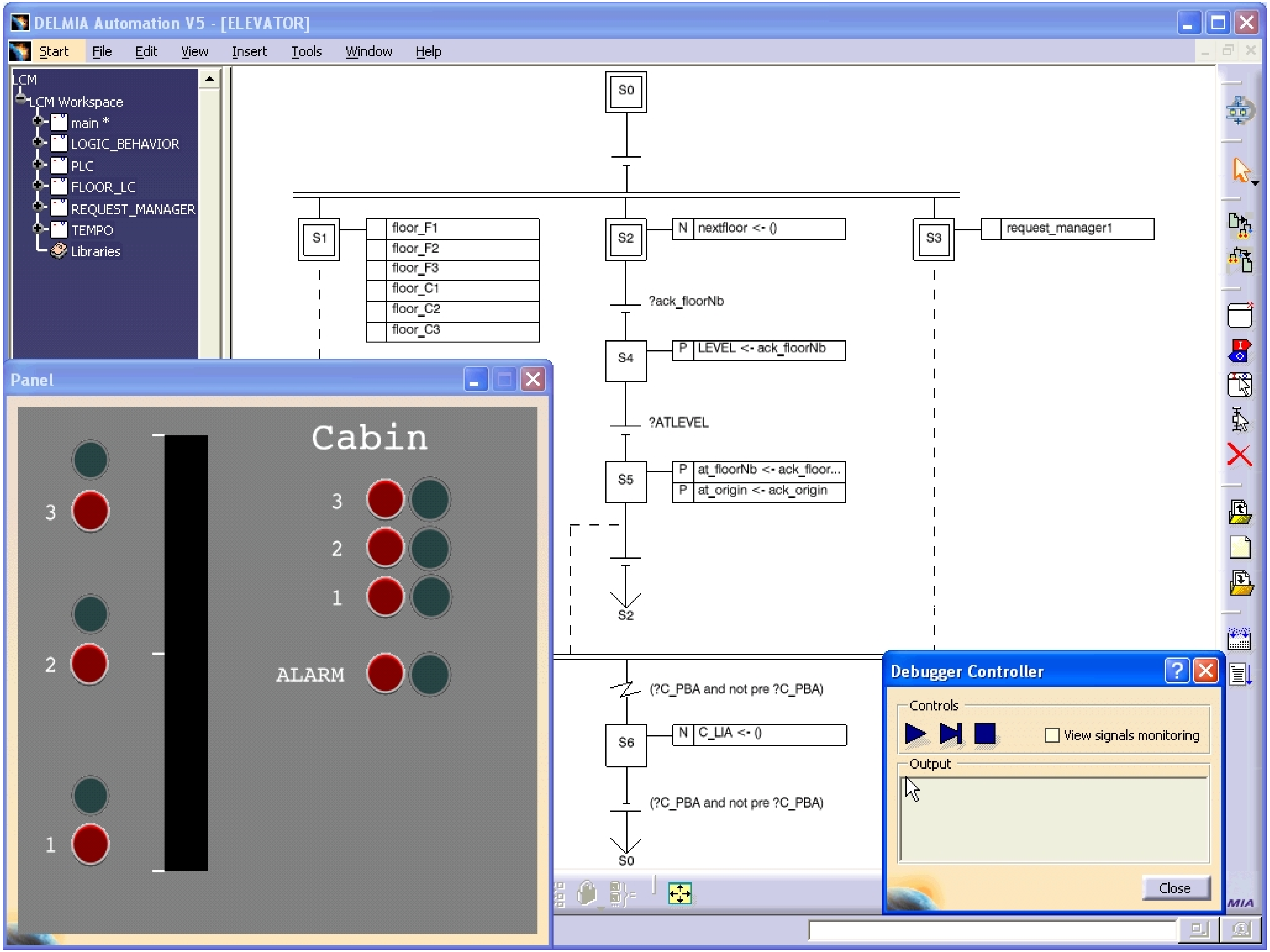
A Concise, Explicit and Rigorous Way to Design a Control Logic

- LCM Studio : high level editor to design the control logic of automated systems
  - ➔ New approach in real time programming:
    - ➔ Behavior of each component separately described
    - ➔ Then global finite state machine is produced using high level composition operators (parallel, preemption, suspend, goto…)
  - ➔ Write things once : thanks to this methodology, states and events combinations are held by the compiler
- LCM Is a High Level Formal Language
  - ➔ The behavior of a LCM program is independent of the implementation (opposite to IEC61131-3)
  - ➔ LCM allows writing safe and explicit programs
  - ➔ The compilation of an LCM program asserts its reactivity and determinism. That is : for each entry and each state combination, it exists one and only one reaction
  - ➔ In other words, there are no blocking situations and no ambiguous situations

Added-values

A Concise, Explicit and Rigorous Way to Design a Control Logic

- LCM is based on a generative graphical tool
  - ➔ Automatic generation of graphical layout
  - ➔ No time lost in graphical manipulations
- Complete identity between the simulated and the target executed program
- Performances (vs. low level tools)
  - ➔ Around 60% reduction in the program size between LCM and a Ladder program
  - ➔ 50% time reduction in specification and validation cycle (edition, debug,..)
  - ➔ Resulting code optimized in performances and memory size


DELMIA Automation V5 - [ELEVATOR]
Start
File
Edit
View
Insert
Tools
Window
Help
LCM
LCM Workspace
main *
LOGIC_BEHAVIOR
PLC
FLOOR_LC
REQUEST_MANAGER
TEMPO
Libraries
S0
S1
floor_F1
floor_F2
floor_F3
floor_C1
floor_C2
floor_C3
S2
N nextfloor <- ()
S3
request_manager1
?ack_floorNb
S4
P LEVEL <- ack_floorNb
?ATLEVEL
S5
P at_floorNb <- ack_floor...
P at_origin <- ack_origin
S2
(?C_PBA and not pre ?C_PBA)
S6
N C_LIA <- ()
(?C_PBA and not pre ?C_PBA)
S0
Panel
Cabin
3
2
1
3
2
1
ALARM
Debugger Controller
Controls
□ View signals monitoring
Output
Close