Working Session 2

Programming the CLEARSY Safety Platform

Paulo Bezerra , Thierry Lecomte

ROBOSTAR

robostar.cs.york.ac.uk

11th March 2025

Royal Academy of Engineering

UKRI Engineering and Physical Sciences Research Council

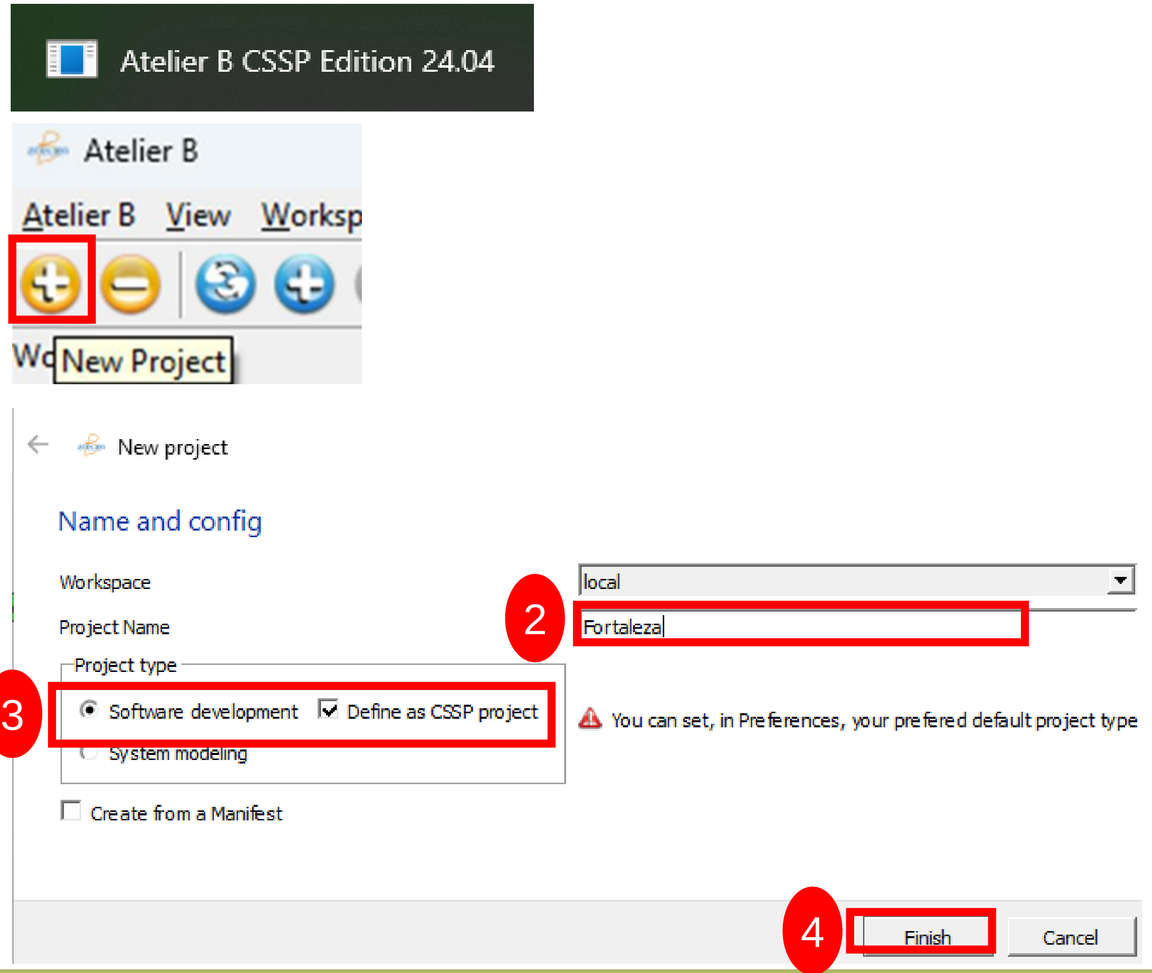
Agenda

- Setting / checking the modelling environment
- Developing a combinatorial function
- Developing a synchronous / timed function

Setting The Modelling Environment

Project Creation 1/3

- Start Atelier B
- Create a new project
- Give a name and a type

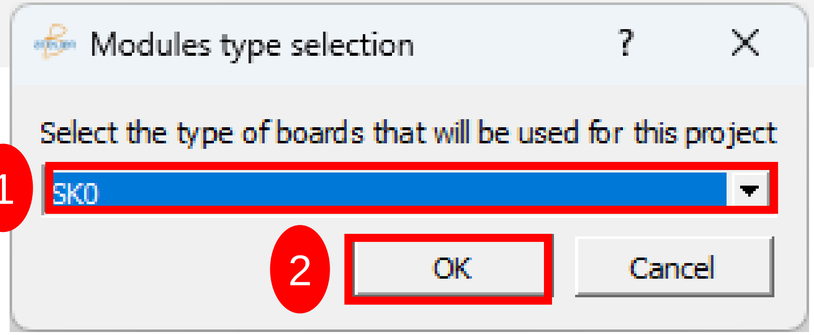


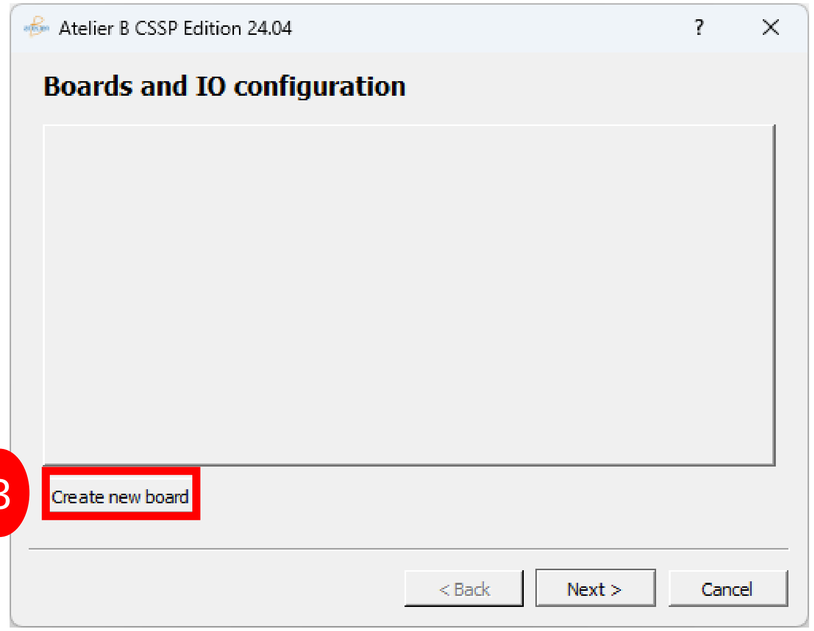
ROBOSTAR

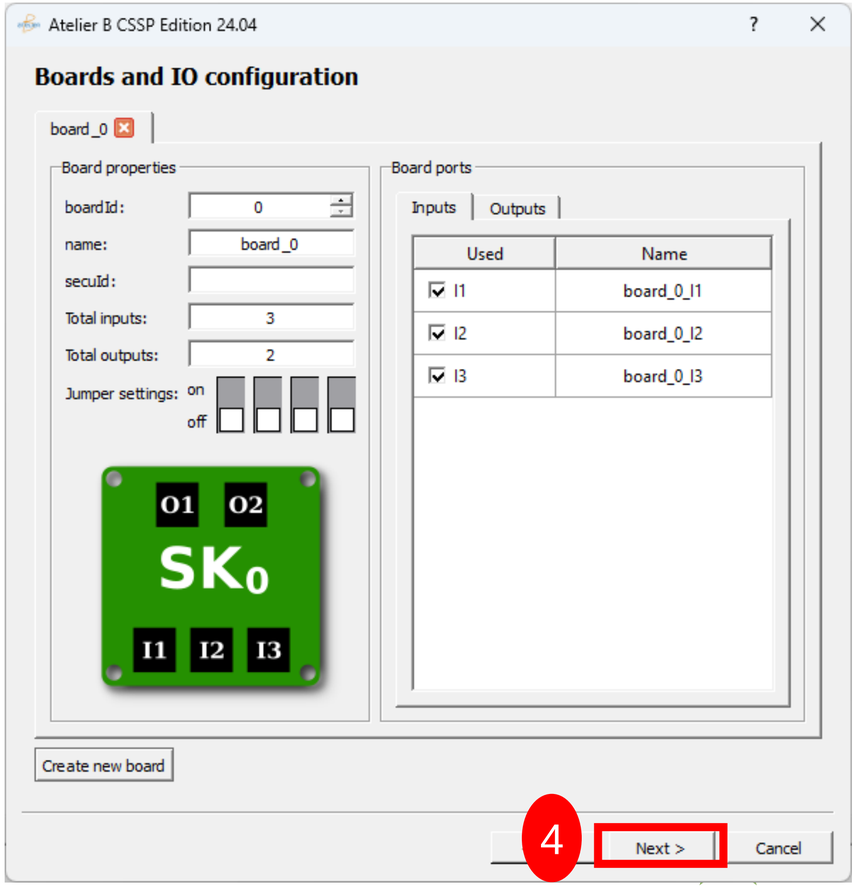
University of York, UK

Project Creation 2/3

- Select SK0
- Create a new board



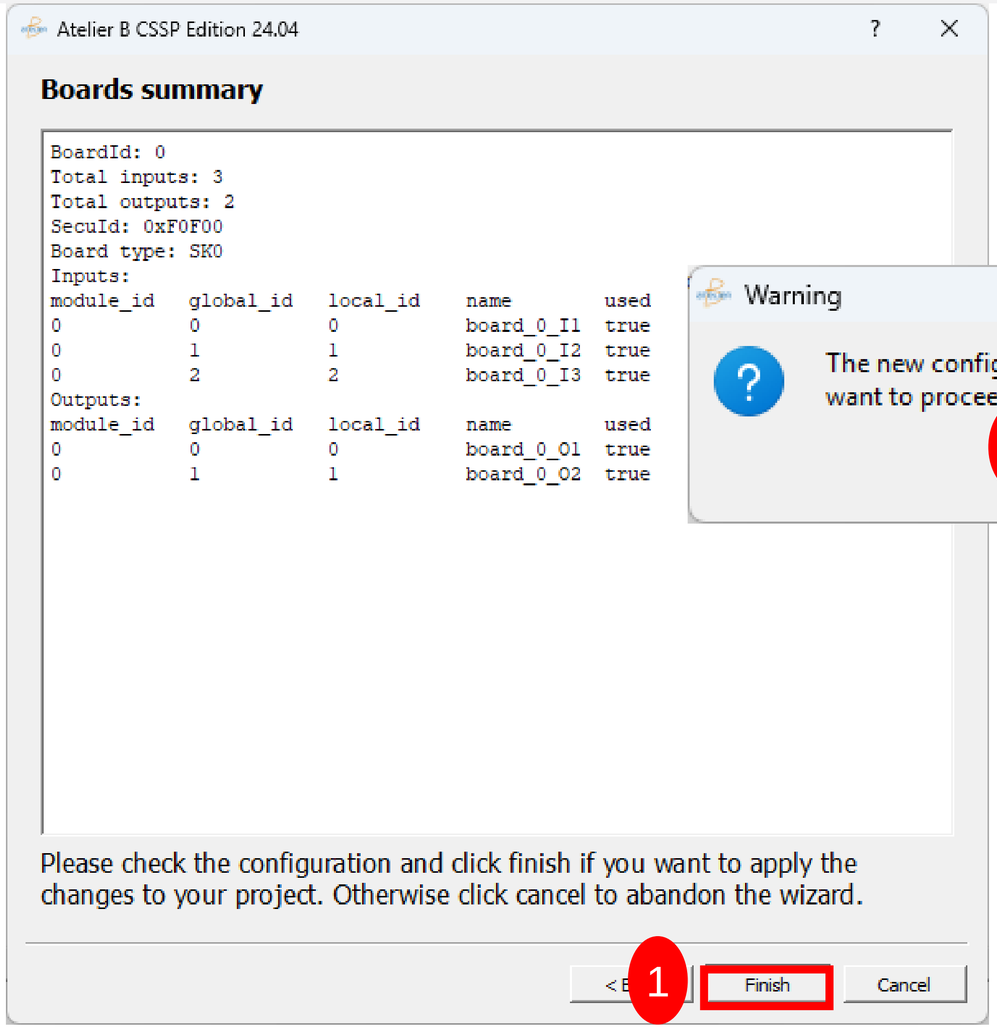




ROBOSTAR

Project Creation 3/3

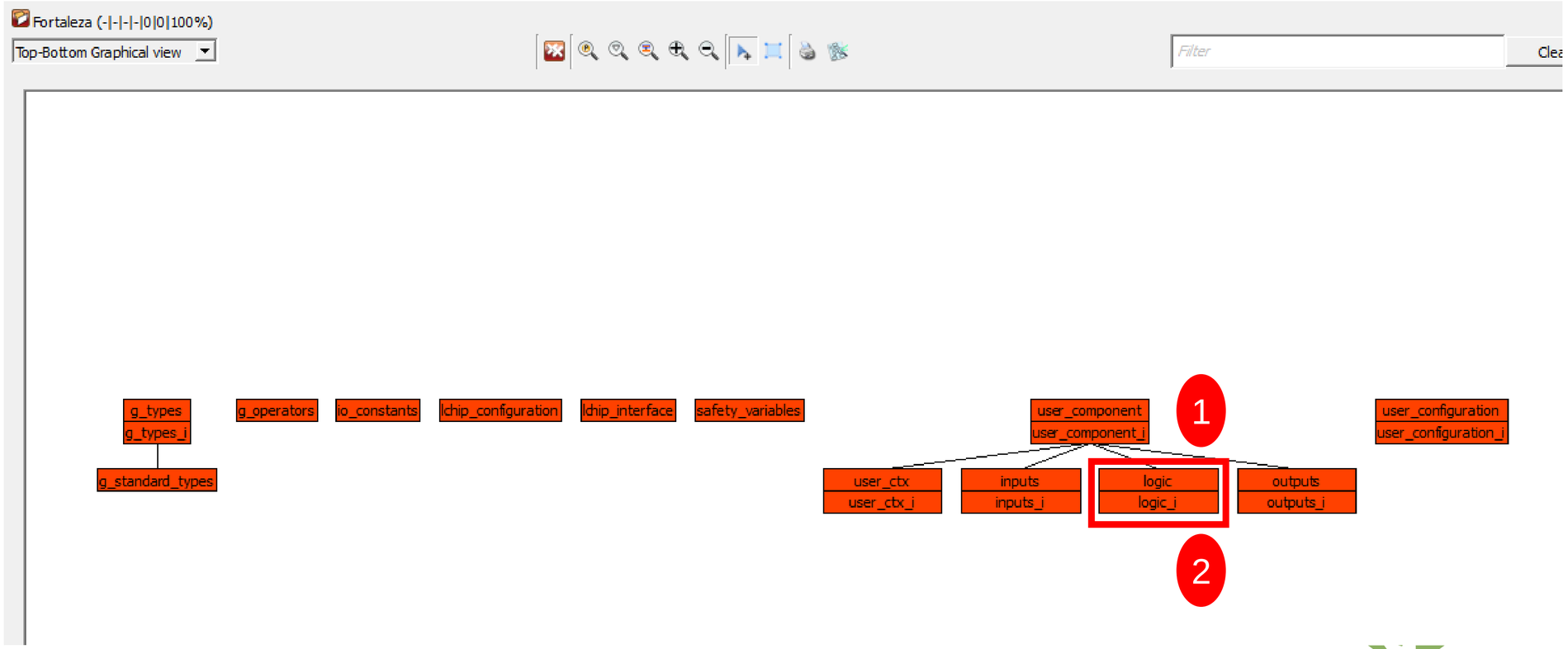
- Finish the creation



ROBOSTAR

Project Created

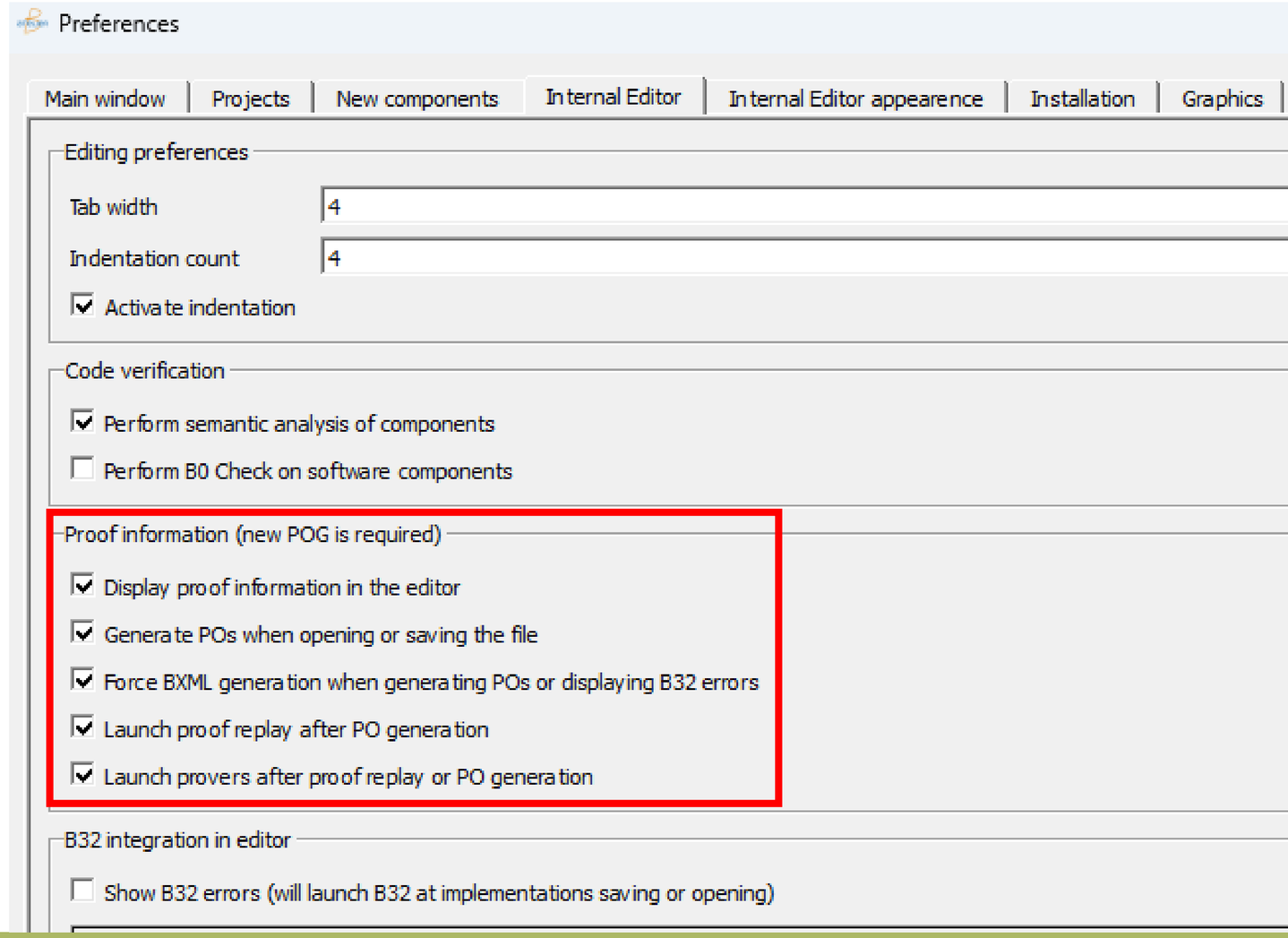
- The view
- 2 components to modify



ROBOSTAR

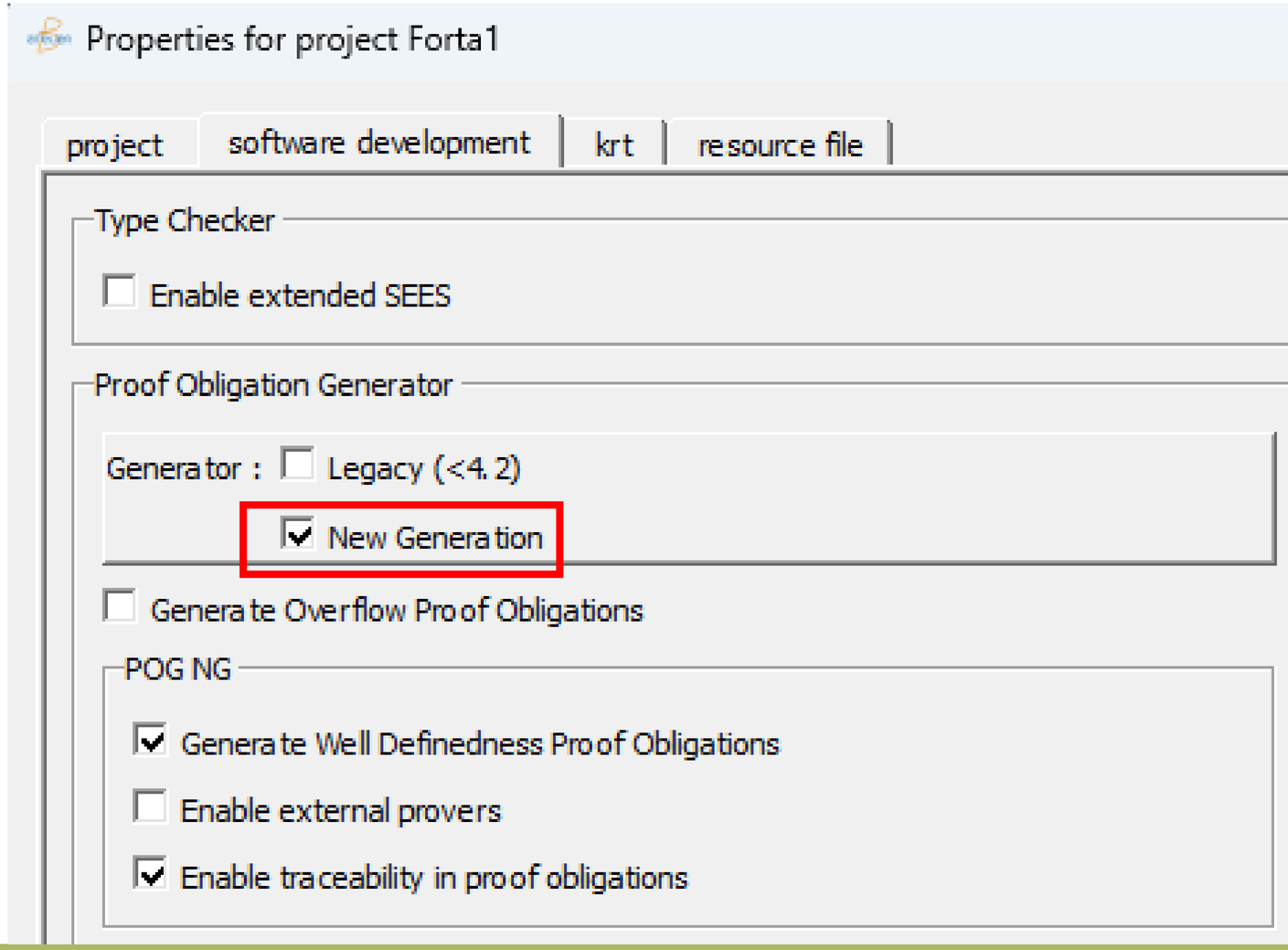
Checking Setup 1/2

- Open menu Atelier B / Preferences
- Select Internal Editor
- Ensure that Proof Information is fully checked



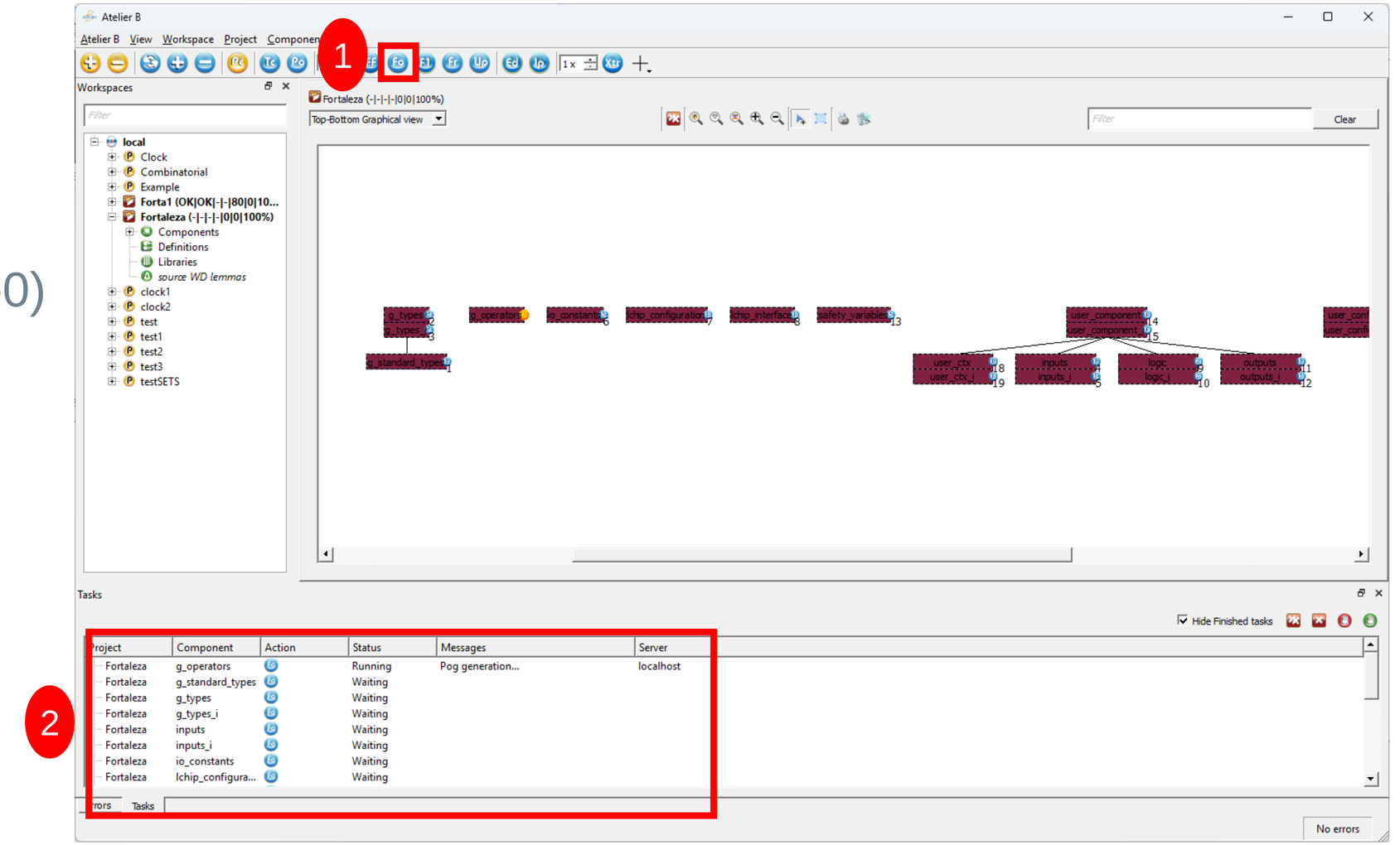
Checking Setup 2/2

- Open your project
- Open menu Project / Properties
- Select Software Development
- Ensure that New Generation is checked in Proof Obligation Generator



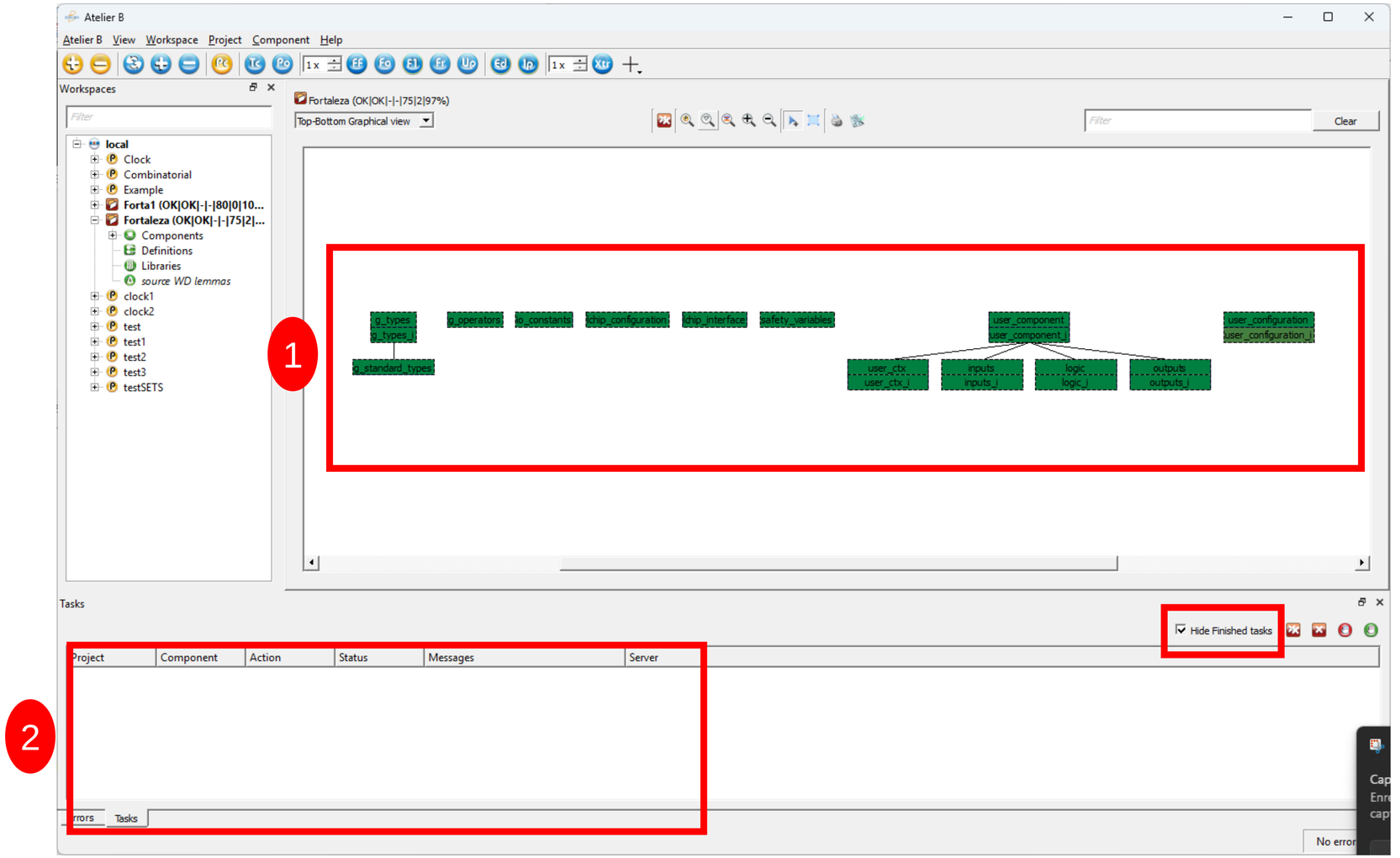
To be sure Your Environment is Operational …1/5

- Select all the components with Ctrl+A
- Start Proof Force 0 (or Ctrl-0)
- Wait for proof to complete



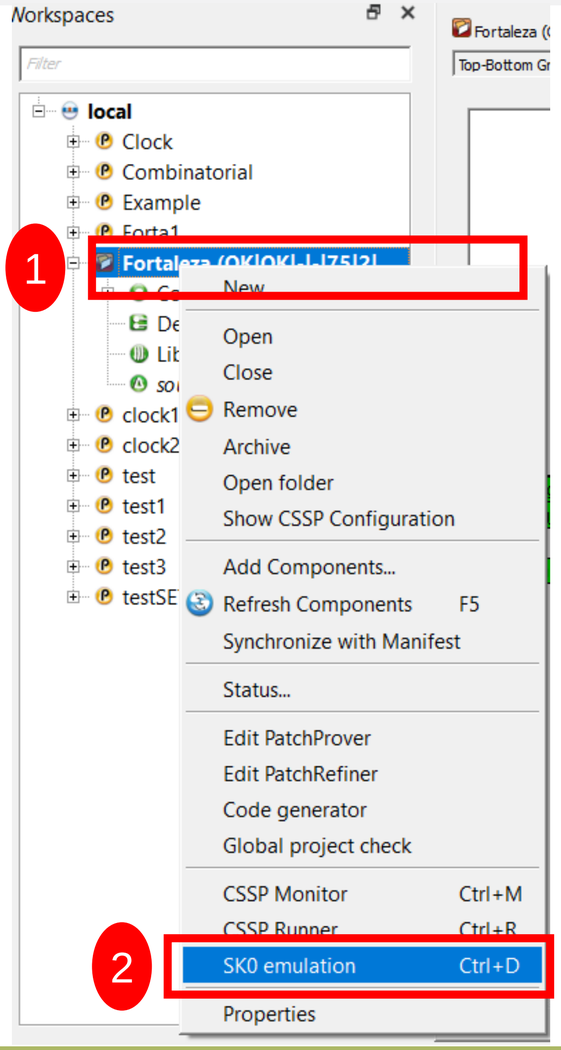
To be sure Your Environment is Operational … 2/5

- All components green
- All tasks completed (select “Hide finished tasks”)



To be sure Your Environment is Operational … 3/5

- Right click on the project
- Select “SK0 emulation” or Ctrl-D

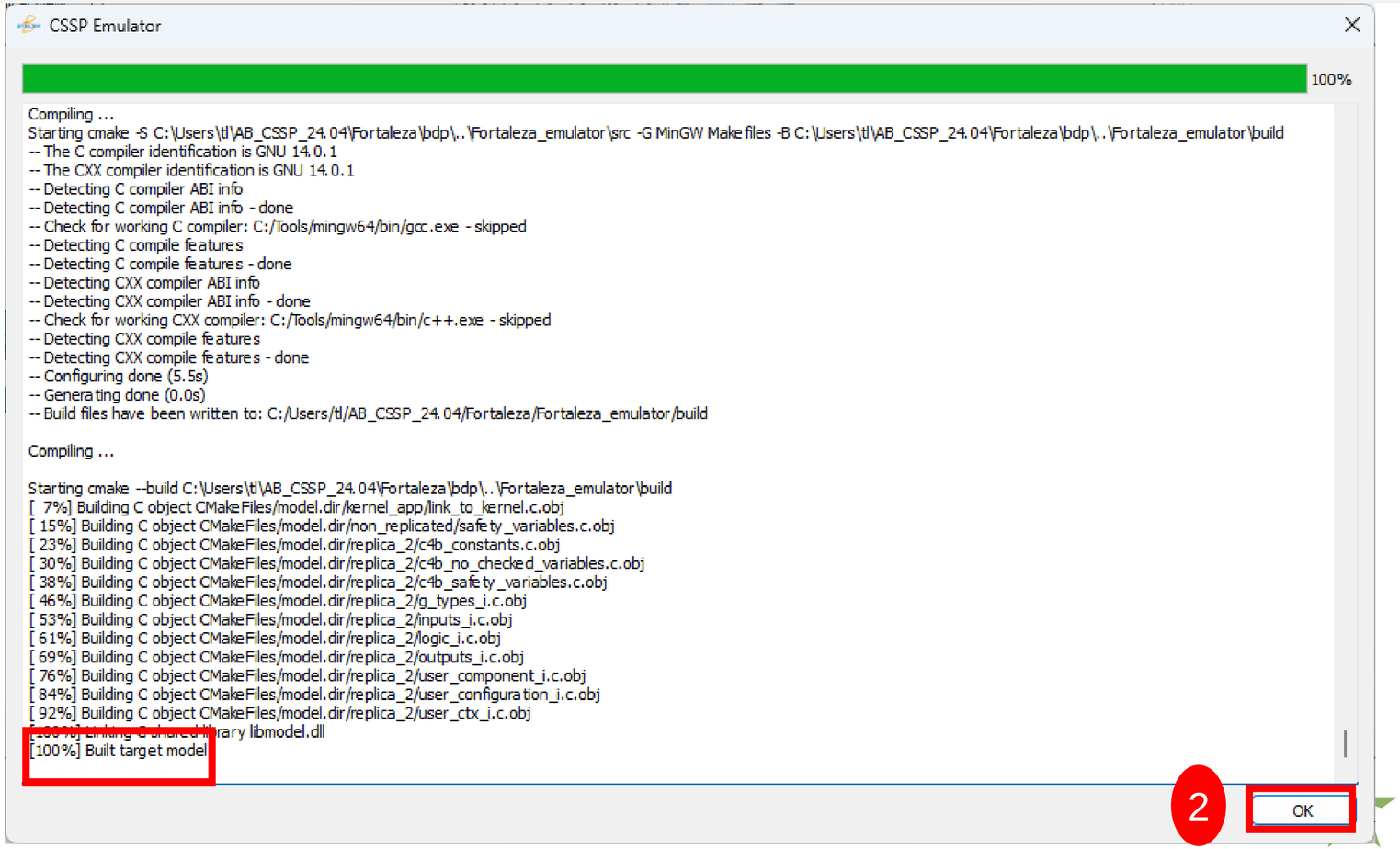


ROBOSTAR

University of York, UK

To be sure Your Environment is Operational … 4/5

- After several seconds and verbose messages …
- The process terminates with [100%] Built target model
- Click on OK

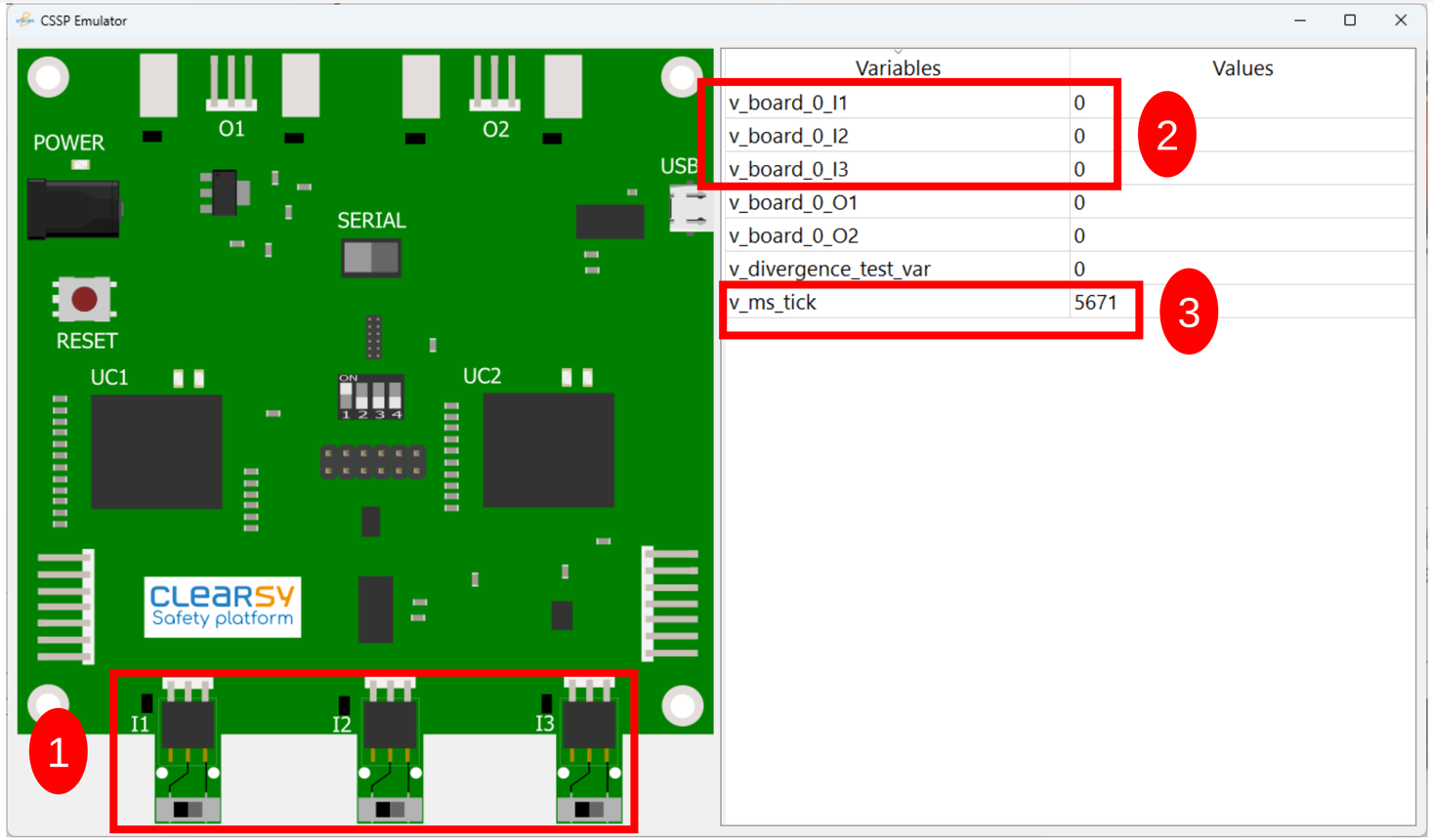


If stops before 100%, cmake / gcc installation is probably incomplete

ROBOSTAR

To be sure Your Environment is Operational … 5/5

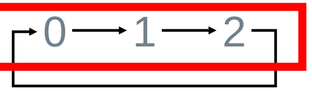
- The simulator starts
- If you click on I1, I2 or I3, the v_board_0_Ix variables change
- The time flies



ROBOSTAR

Developping a Combinatorial Function

Combinatorial Function

- *I1* belongs to {IO_OFF, IO_ON}
- *status* belongs to 0..2
- Every time *I1* changes from IO_OFF to IO_ON, *status* changes such as 

We need to read input I1

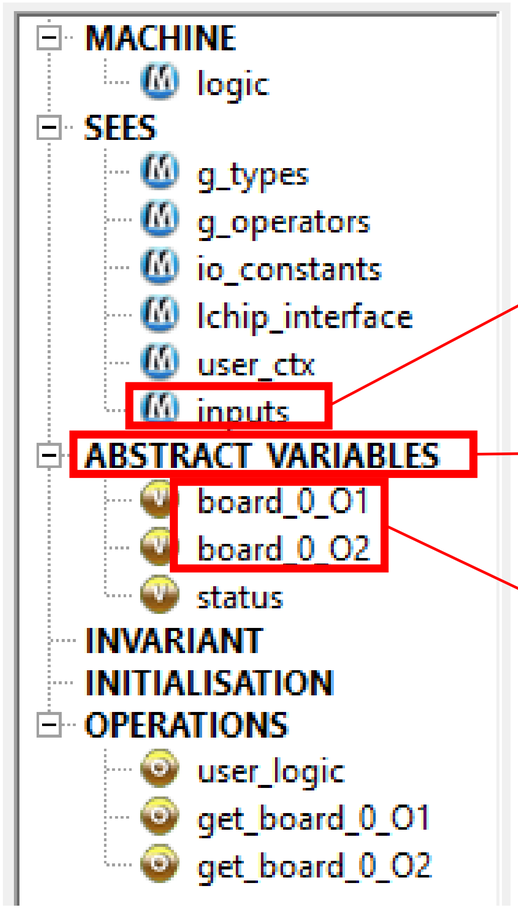
We need to program a behaviour

ROBOSTAR

Combinatorial Function

Specification component

double-click to open



Inputs are not variables of this model.
They are read and accessed from the *inputs* component

Abstract variables are not necessarily implemented.
They could appear only in specification for « reasoning »

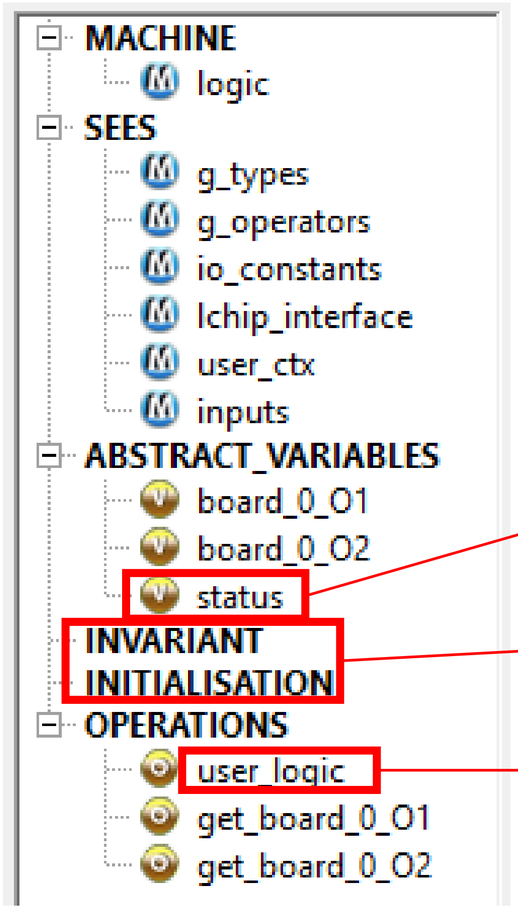
Outputs are variables of this component.
They are modified in this component.
In B, variables are modified in a single component

ROBOSTAR

Combinatorial Function

Specification component

double-click to open



addition

modification

modification

```
10-     ABSTRACT_VARIABLES
11          board_0_O1,
12          board_0_O2,
13          status
14-     INVARIANT
15          board_0_O1 : uint8_t &
16          board_0_O2 : uint8_t &
17  1/1     status : uint8_t
18-     INITIALISATION
19          board_0_O1 :: uint8_t ||
20          board_0_O2 :: uint8_t ||
21  1/1     status := 0
22-     OPERATIONS
23-     user_logic = BEGIN
24          board_0_O1,
25          board_0_O2,
26-         status: (
27              board_0_O1 : uint8_t &
28              board_0_O2 : uint8_t &
29              status : uint8_t
30          )
31      END;
```

list

conjunct

Parallel initialisation
No order

Non deterministic substitution

ROBOSTAR

Combinatorial Function

Specification component

double-click to open


```
10-     ABSTRACT_VARIABLES
11          board_0_O1,
12          board_0_O2,
13          status
14-     INVARIANT
15          board_0_O1 : uint8_t &
16          board_0_O2 : uint8_t &
17  1/1     status : uint8_t
18-     INITIALISATION
19          board_0_O1 :: uint8_t ||
20          board_0_O2 :: uint8_t ||
21  1/1     status := 0
22-     OPERATIONS
23-         user_logic = BEGIN
24              board_0_O1,
25              board_0_O2,
26-             status: (
27                  board_0_O1 : uint8_t &
28                  board_0_O2 : uint8_t &
29                  status : uint8_t
30              )
31      END;
```

The 3 variables are going to evolve according to their type and could keep the same value

xx: (xx : T(xx))

means

« xx becomes such that xx belongs to its type »

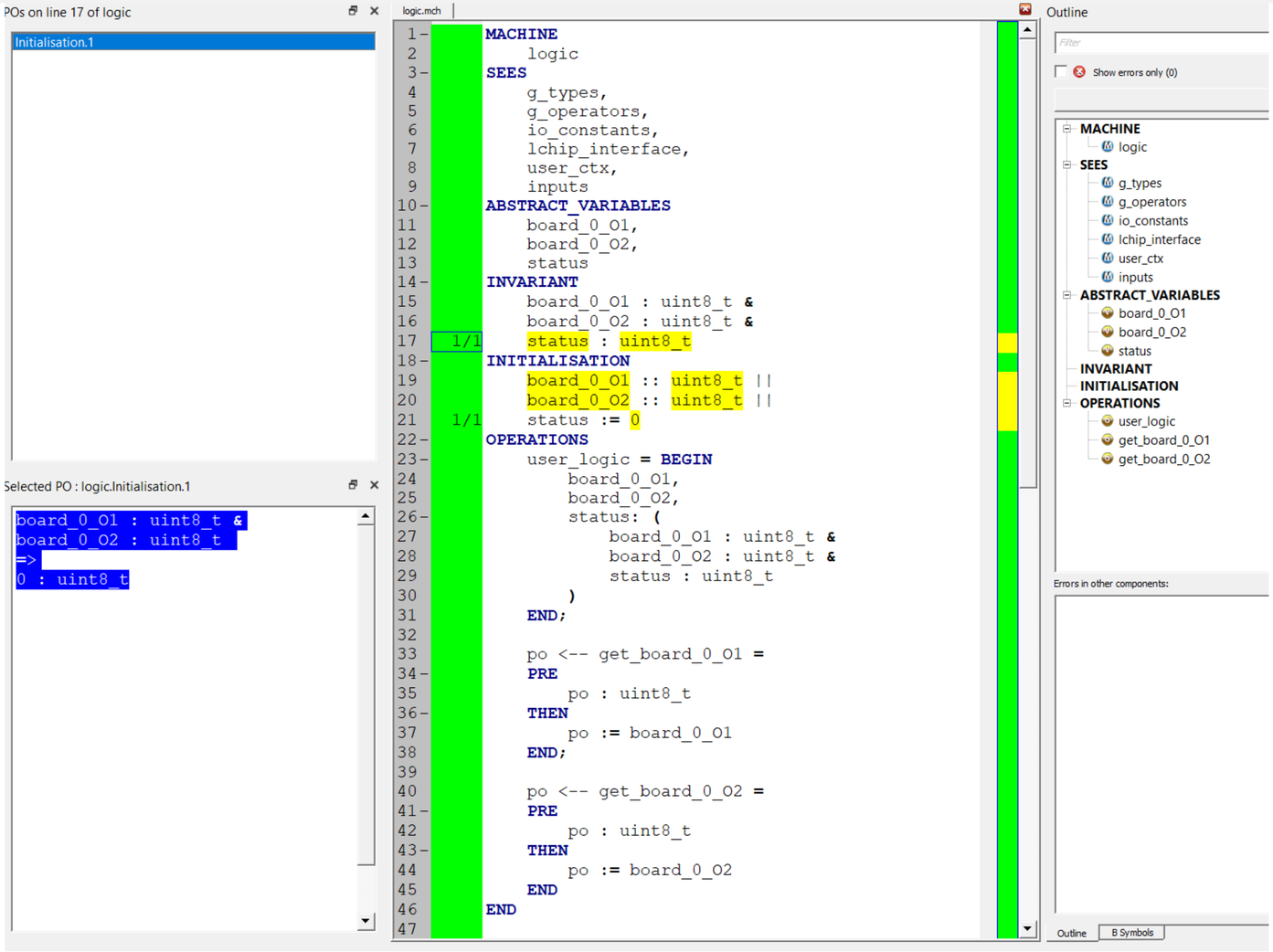
ROBOSTAR

Combinatorial Function

Specification component

double-click to open

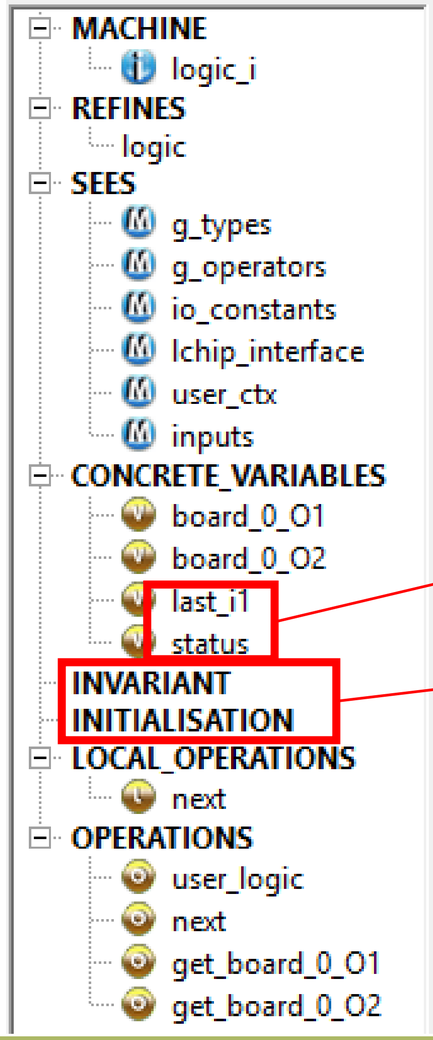
- Once the editing is completed
- Type Ctrl-S to save
- Proof is automatically initiated
- You should get a fully green (proved) component



Combinatorial Function

Implementation component

double-click to open



addition

modification

```
14 -        CONCRETE_VARIABLES
15              board_0_O1,
16              board_0_O2,
17              last_i1,
18      2/2     status
19 -        INVARIANT
20              board_0_O1 : uint8_t &
21              board_0_O2 : uint8_t &
22              last_i1 : uint8_t &
23      2/2     status : uint8_t
24 -        INITIALISATION
25              board_0_O1 := IO_OFF;
26              board_0_O2 := IO_OFF;
27              last_i1 := IO_OFF;
28      1/1     status := 0
```

means « to appear in code »

We need to store the « previous value of *I1* »

Sequential initialisation
Is separator, not end-of-line

The previous value of *I1* is considered not pressed (or down) at initialisation

ROBOSTAR

Combinatorial Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

addition

```
29-         LOCAL OPERATIONS
30              n  <-- next(state) =
31-             PRE state: uint8_t & n_ : uint8_t THEN
32    4/4           n_ :: uint8_t - {state}
33              END
34-         OPERATIONS
35              user_logic =
36+   1/1       BEGIN
50              END;
51
52    3/3       n_ <-- next(state) =
53-             BEGIN
54    1/1           IF state = 0 THEN n_ := 1
55    1/1           ELSIF state = 1 THEN n_ := 2
56    1/1           ELSE n_ := 0
57                  END
58              END;
```

OPERATIONS declared locally

user_logic can call *next*

Operation *next* with one parameter *state* returning *n_*

Precondition used to indicate types for parameter and return value

Non-deterministic substitution: *n_* becomes equal to any uint8_t except the value of *state*

ROBOSTAR

Combinatorial Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

addition

```
29-         LOCAL_OPERATIONS
30              n_ <-- next(state) =
31-             PRE state: uint8_t & n_ : uint8_t THEN
32    4/4           n_ :: uint8_t - {state}
33              END
34-         OPERATIONS
35              user_logic =
36+   1/1       BEGIN
50              END;
51
52    3/3       n_ <-- next(state) =
53-             BEGIN
54    1/1           IF state = 0 THEN n_ := 1
55    1/1           ELSIF state = 1 THEN n_ := 2
56    1/1           ELSE n_ := 0
57                  END
58              END;
```

Same signature
Types are kept from
LOCAL_OPERATIONS precondition

Algorithm implemented with
IF THEN ELSE

Assignment

ROBOSTAR

Combinatorial Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

modification

```
35
36-
37-
38
39
40
41
42-
43-
44      1/1
45
46
47
48
49
```

*i1_* local variable used in this block

```
user_logic =
BEGIN
    VAR i1_ IN
        i1_ : (i1_ : uint8_t);
        TO COMPLETE
    END
END;
```

Variables list

Typing predicates used by code generator.
No impact on behaviour

ROBOSTAR

Combinatorial Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

```
35
36-
37-
38
39
40
41
42-
43-
44      1/1
45
46
47
48
49
```

```
user_logic =
BEGIN
    VAR i1_ IN
        i1_ : (i1_ : uint8_t);

    END
END;
```

If I1 has a raising edge then change status to its next value.

Update the last value of I1

ROBOSTAR

Combinatorial Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

```
35
36-
37-
38
39
40
41
42-
43-
44      1/1
45
46
47
48
49
```

```
user_logic =
BEGIN
    VAR i1_ IN
        i1_ : (i1_ : uint8_t);

    END
END;
```

If I1 has a raising edge then change status to its next value.

Update the last value of I1

HINT1
Operation call syntax:
vv <-- ff(pp)

HINT2
The operation to read *I1* is in the *inputs* component

HINT3
To assign a variable *vv*:
vv := value

CONSTRAINT
Only one condition per IF

ROBOSTAR

Combinatorial Function: your turn

Implementation component

double-click to open

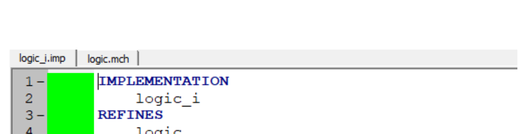
1 **Edit logic and logic_i components**
Ctrl-S to save and prove
Models should be « all green »

2 **Ctrl-D to start the emulator**
Reach « 100% built target model »
Press OK

3 **Click on I1 several times and observe the behaviour of *status***

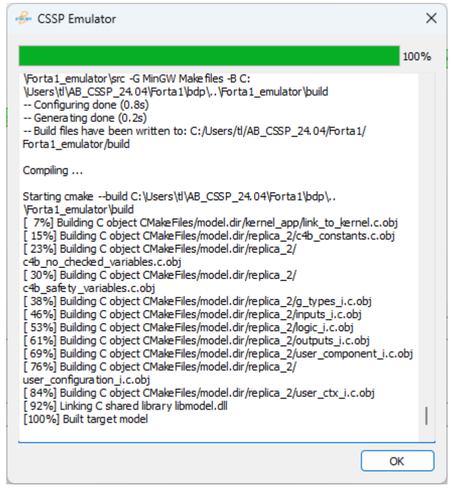
logic_i.imp | logic.mch

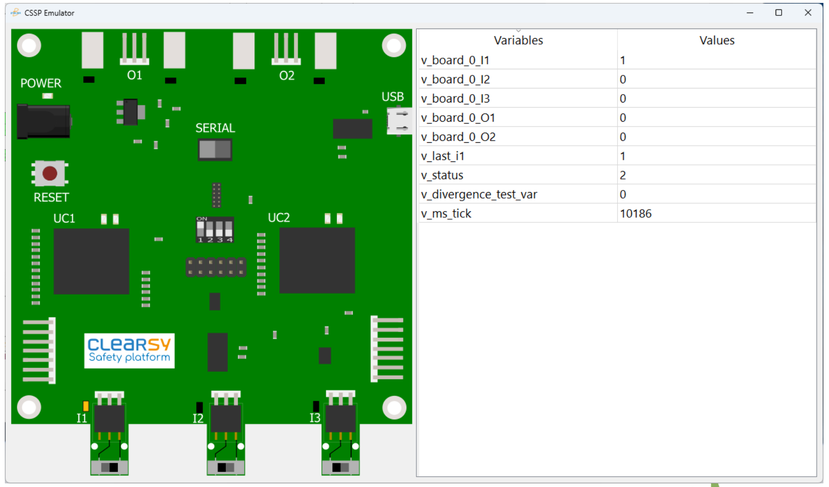
```
1-  MACHINE
2       logic
3-  SEES
4       g_types,
5       g_operators,
6       io_constants,
7       lchip_interface,
8       user_ctx,
9       inputs
10- ABSTRACT_VARIABLES
11      board_0_O1,
12      board_0_O2,
13      status
14- INVARIANT
15      board_0_O1 : uint8_t &
16      board_0_O2 : uint8_t &
17  1/1
18- IN
19
20
21  1/1
22- OPE
23-
24
25
26-         status: (
27              board_0_O1 : uint8_t &
28              board_0_O2 : uint8_t &
29              status : uint8_t
30          )
31      END;
32
33      po <-- get_board_0_O1 =
34-     PRE
35          po : uint8_t
36-     THEN
37          po := board_0_O1
38      END;
39
40      po <-- get_board_0_O2 =
41-     PRE
```



logic_i.imp | logic.mch

```
1-  IMPLEMENTATION
2       logic_i
3-  REFINES
4       logic
5-  SEES
6       g_types,
7       g_operators,
8       io_constants,
9       lchip_interface,
10      user_ctx,
11      inputs
12
13      // pragma SAFETY_VARS
14- CONCRETE_VARIABLES
15      board_0_O1,
16      board_0_O2
17
18  2/2
19- IN
20
21
22
23  2/2
24- IN
25
26      board_0_O2 := IO_OFF;
27      last_i1 := IO_OFF;
28  1/1 status := 0
29- LOCAL_OPERATIONS
30      n_ <-- next(state) =
31-     PRE state: uint8_t & n_ : uint8_t THEN
32  4/4     n_ :: uint8_t - {state}
33      END
34- OPERATIONS
35      user_logic =
36-     BEGIN
37-         VAR i1_, n_ IN
38              i1_ : (i1_ : uint8_t);
39              n_ : (n_ : uint8_t);
40
41              i1_ <-- get_board_0_I1;
42
```





ROBOSTAR

Combinatorial Function: your turn

Hard Work On Going !

Spoiler Alert: Solution on the next slide !

Combinatorial Function : the solution

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

modification

```
35
36 -
37 -
38
39
40
41
42 -
43 -
44      1/1
45
46
47
48
49
50
51      3/3
52 -
53      1/1
54      1/1
55      1/1
56
57
```

```
user_logic =
BEGIN
    VAR i1_ IN
        i1_ : (i1_ : uint8_t);

        i1_ <-- get_board_0_I1;

        IF last_i1 = IO_OFF THEN
            IF i1_ = IO_ON THEN
                status <-- next(status)
            END
        END;
        last_i1 := i1_
    END
END;

n_ <-- next(state) =
BEGIN
    IF state = 0 THEN n_ := 1
    ELSIF state = 1 THEN n_ := 2
    ELSE n_ := 0
    END
END;
```

We call the predefined operation *get_board_0_I1* to get the value of I1 and store it in *i1_*

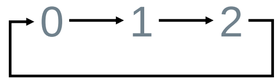
We call *next* to get the new value of *status*

ROBOSTAR

Developping a Synchronous Function

Synchronous / Timed Function

- I1 belongs to {IO_OFF, IO_ON}
- status belongs to 0..2
- Every time I1 changes from IO_OFF to IO_ON, status changes such as
- If status = 2 during 2 s or more then sets O1 to IO_ON
- Otherwise sets O1 to IO_OFF

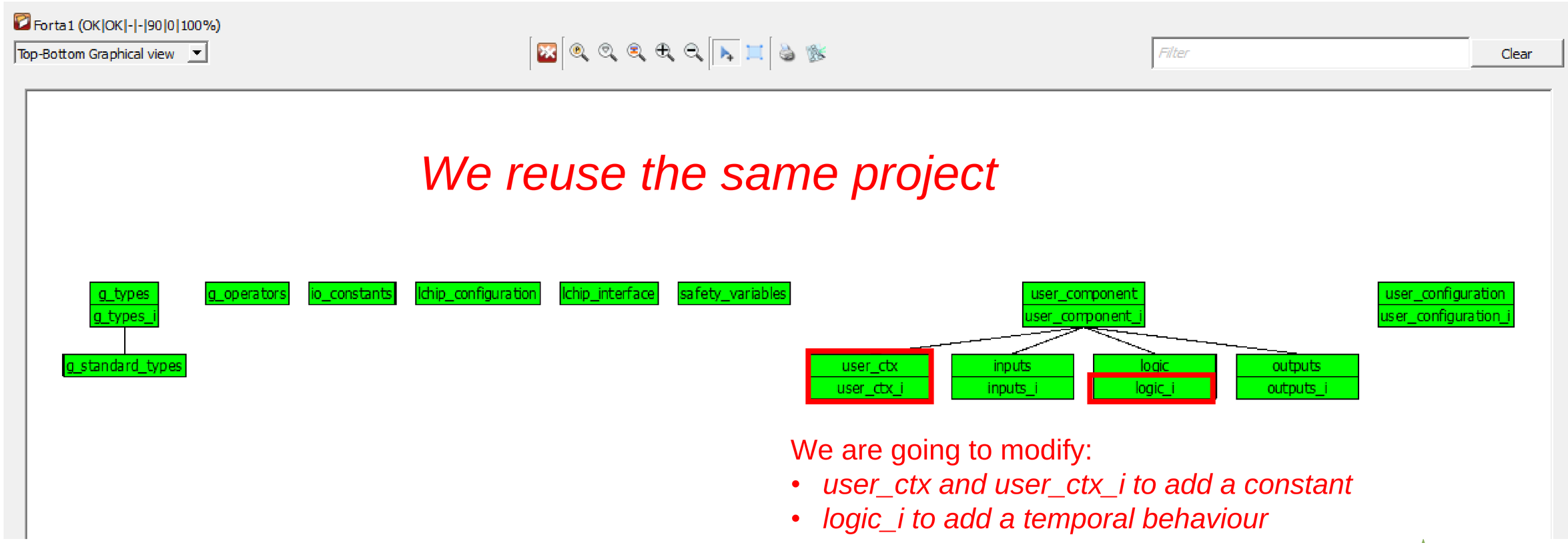


We need to specify a timer

We need to command output O1

ROBOSTAR

Synchronous / Timed Function

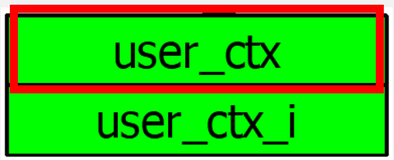


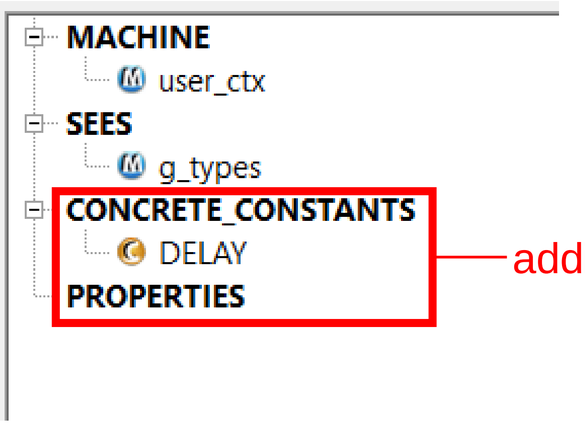
ROBOSTAR

Synchronous / Timed Function

Specification component

double-click to open





addition

user_ctx.mch

```
1  MACHINE
2      user_ctx
3  SEES
4      g_types
5  CONSTANTS
6      DELAY
7  PROPERTIES
8      DELAY : uint32_t
9  END
```

CONSTANTS are defined and used with their properties whatever their values

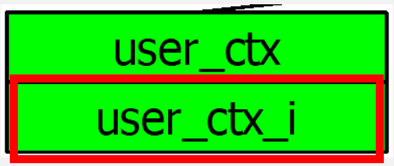
DELAY is Integer

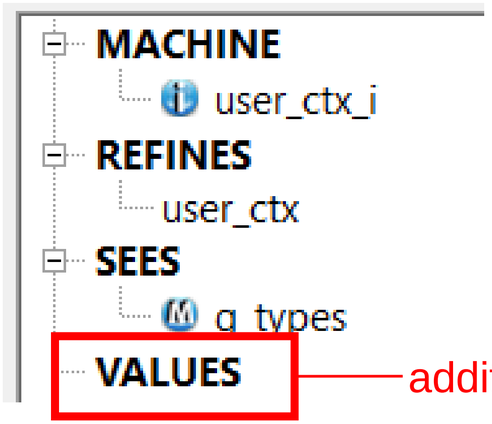
ROBOSTAR

Synchronous / Timed Function

Implementation component

double-click to open





addition

user_ctx.mch | user_ctx_i.imp

```
1-      IMPLEMENTATION
2           user_ctx_i
3-      REFINES
4           user_ctx
5
6           // pragma CONSTANTS
7-      SEES
8           g_types
9-      VALUES
10    1/1   DELAY = 2000
11      END
```

In

DELAY : uint32_t &
DELAY > 10 &
DELAY < 2

It is not possible to value DELAY: **it is a miracle**

Each CONSTANT should be valued to demonstrate it is not a miracle
i.e. *2000* belongs to uint32_t

Time returned by *get_ms_tick* OPERATION is in ms
Hence 2s == 2000 ms

ROBOSTAR

Synchronous / Timed Function

Specification component

double-click to open

- MACHINE
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- ABSTRACT_VARIABLES
  - board_0_O1
  - board_0_O2
  - status
- INVARIANT
- INITIALISATION
- OPERATIONS
  - user_logic
  - get_board_0_O1
  - get_board_0_O2

No change !

logic_i.imp | logic.mch

```
1-      MACHINE
2           logic
3-      SEES
4           g_types,
5           g_operators,
6           io_constants,
7           lchip_interface,
8           user_ctx,
9           inputs
10-     ABSTRACT_VARIABLES
11          board_0_O1,
12          board_0_O2,
13          status
14-     INVARIANT
15          board_0_O1 : uint8_t &
16          board_0_O2 : uint8_t &
17  1/1     status : uint8_t
18-     INITIALISATION
19          board_0_O1 :: uint8_t ||
20          board_0_O2 :: uint8_t ||
21  1/1     status := 0
22-     OPERATIONS
23-         user_logic = BEGIN
24              board_0_O1,
25              board_0_O2,
26-             status: (
27                  board_0_O1 : uint8_t &
28                  board_0_O2 : uint8_t &
29                  status : uint8_t
30              )
31          END;
```

ROBOSTAR

Synchronous / Timed Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
  - t0_s_2 — addition
- INVARIANT — modification
- INITIALISATION — modification
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

logic_i.imp

```
14-         CONCRETE_VARIABLES
15              board_0_O1,
16              board_0_O2,
17              last_i1,
18    4/4       status,
19    4/4       t0_s_2
20-         INVARIANT
21              board_0_O1 : uint8_t &
22              board_0_O2 : uint8_t &
23              last_i1 : uint8_t &
24    4/4       status : uint8_t  &
25    1/1       t0_s_2 : uint32_t
26-         INITIALISATION
27              board_0_O1 := IO_OFF;
28              board_0_O2 := IO_OFF;
29              last_i1 := IO_OFF;
30    1/1       status := 0;
31    1/1       t0_s_2 := 0
```

We need to store the last time *status* changed to the value 2

Timing information returned by the board is uint32_t

The initial value is 0, even if *status* is different from 2

ROBOSTAR

Synchronous / Timed Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
  - t0_s_2
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

get_ms_tick return the number of ms since last reboot

modification

```
38        user_logic =
39-       BEGIN
40-           VAR i1_, t_, d_ IN
41                i1_ : (i1_ : uint8_t);
42                t_ : (                );
43                d_ : (                );
44
45                i1_ <-- get_board_0_I1;
46    3/3         t_ <-- get_ms_tick;
47-               IF last_i1 = IO_OFF THEN
48-                   IF i1_ = IO_ON THEN
49    3/3               status <-- next(status);
50-                     IF                THEN
51    1/1                   t0_s_2 :=
52                      END
53                  END
54                END;
55                last_i1 := i1_;
56
57-
58    6/6
59-
60
61
62                END
63            END
64        END;
```

Type *t_* used to store the current time

Type *d_* used to store the difference between current time and last time *status* changed to 2

What is the condition if we are going to modify the last time *status* changed to 2 ?

What value is assigned to *t0_s_2* ?

Turn on *board_0_O1* if *status* has been 2 during at least 2s.
If not, turn it off

ROBOSTAR

Synchronous / Timed Function

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
  - t0_s_2
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

modification

```
38          user_logic =
39-         BEGIN
40-             VAR i1_, t_, d_ IN
41                  i1_ : (i1_ : uint8_t);
42                  t_ : (              );
43                  d_ : (              );
44
45                  i1_ <-- get_board_0_I1;
46      3/3         t_ <-- get_ms_tick;
47-                 IF last_i1 = IO_OFF THEN
48-                     IF i1_ = IO_ON THEN
49      3/3                 status <-- next(status);
50-                         IF              THEN
51      1/1                     t0_s_2 :=
52                          END
53                  END
54                  END;
55                  last_i1 := i1_;
56
57-                 Turn on board_0_O1 if status has
58      6/6         been 2 during at least 2s.
59-                 If not turn off 01
60
61
62                  END
63              END
64          END;
```

HINT1

To turn on an ouput OO

OO := IO_ON

To turn it off

OO := IO_OFF

HINT2

Substraction of two uint32_t with *sub_uint32*

Ex: vv := *sub_uint32*(aa,bb)

CONSTRAINT1

Condition only with values, no calculation

CONSTRAINT2

Only < or <= , no > or >= for condition

ROBOSTAR

Synchronous / Timed Function: your turn

Implementation component

double-click to open

1 **Edit logic and logic_i components**
Ctrl-S to save and prove
Models should be « all green »

2 **Ctrl-D to start the emulator**
Reach « 100% built target model »
Press OK

3 **Click on I1 several times and observe the behaviour of *status* and *board_0_O1***

logic_i.imp | logic.mch

```
1-  MACHINE
2       logic
3-  SEES
4       g_types,
5       g_operators,
6       io_constants,
7       lchip_interface,
8       user_ctx,
9       inputs
10- ABSTRACT_VARIABLES
11      board_0_O1,
12      board_0_O2,
13      status
14- INVARIANT
15      board_0_O1 : uint8_t &
16      board_0_O2 : uint8_t &
17  1/1
18- IN
19
20
21  1/1
22- OPE
23-
24
25
26-         status: (
27              board_0_O1 : uint8_t &
28              board_0_O2 : uint8_t &
29              status : uint8_t
30          )
31      END;
32
33      po <-- get_board_0_O1 =
34-     PRE
35          po : uint8_t
36-     THEN
37          po := board_0_O1
38      END;
39
40      po <-- get_board_0_O2 =
41-     PRE
```


logic_i.imp | logic.mch

```
1-  IMPLEMENTATION
2       logic_i
3-  REFINES
4       logic
5-  SEES
6       g_types,
7       g_operators,
8       io_constants,
9       lchip_interface,
10      user_ctx,
11      inputs
12
13      // pragma SAFETY_VARS
14- CONCRETE_VARIABLES
15      board_0_O1,
16      board_0_O2
17
18  2/2
19- IN
20
21
22
23  2/2
24- IN
25
26      board_0_O2 := IO_OFF;
27      last_i1 := IO_OFF;
28  1/1 status := 0
29- LOCAL_OPERATIONS
30      n_ <-- next(state) =
31-     PRE state: uint8_t & n_ : uint8_t THEN
32  4/4     n_ :: uint8_t - {state}
33      END
34- OPERATIONS
35      user_logic =
36-     BEGIN
37-         VAR i1_, n_ IN
38              i1_ : (i1_ : uint8_t);
39              n_ : (n_ : uint8_t);
40
41              i1_ <-- get_board_0_I1;
42
```


ROBOSTAR

Synchronous / Timed Function: your turn

Hard Work On Going !

Spoiler Alert: Solution on the next slide !

Synchronous / Timed Function : the solution

Implementation component

double-click to open

- MACHINE
  - logic_i
- REFINES
  - logic
- SEES
  - g_types
  - g_operators
  - io_constants
  - lchip_interface
  - user_ctx
  - inputs
- CONCRETE_VARIABLES
  - board_0_O1
  - board_0_O2
  - last_i1
  - status
  - t0_s_2
- INVARIANT
- INITIALISATION
- LOCAL_OPERATIONS
  - next
- OPERATIONS
  - user_logic
  - next
  - get_board_0_O1
  - get_board_0_O2

modification

```
38          user_logic =
39-         BEGIN
40-             VAR i1_, t_, d_ IN
41                  i1_ : (i1_ : uint8_t);
42                  t_ : (t_ : uint32_t);
43                  d_ : (d_ : uint32_t);
44
45                  i1_ <-- get_board_0_I1;
46      3/3         t_ <-- get_ms_tick;
47-                 IF last_i1 = IO_OFF THEN
48-                     IF i1_ = IO_ON THEN
49      3/3                 status <-- next(status);
50-                         IF status = 2 THEN
51      1/1                     t0_s_2 := t_
52                          END
53                      END
54                  END;
55                  last_i1 := i1_;
56                  board_0_O1 := IO_OFF;
57-                 IF status = 2 THEN
58      6/6             d_ := sub_uint32(t_, t0_s_2);
59-                     IF DELAY <= d_ THEN
60                          board_0_O1 := IO_ON
61                      END
62                  END
63              END
64          END;
```

Both *t_* and *d_* are unit32_t

status has just changed to 2.
It is the right time to store the current time

*d_* is meaningful only if *status* is 2

ROBOSTAR

Exercise to Go Further

- I1 {IO_OFF, IO_ON}
- I1 is noisy, changing value up to thousands times per second
- Only I1 constant during at least 100 ms have to be considered and are repeated on O1 – and O2 has to be IO_ON
- If no constant behavior is observed, then O2 has to be IO_OFF – in this case, the status of O1 is not considered (could be any value)
- Summary:
  - O2 is IO_ON when I1 has been constant during at least 100 ms and O1 has the status of the observed I1
  - O2 is IO_OFF when I1 has not been constant during the last 100 ms

ROBOSTAR

Next

From RoboSim To The CLEARSY Safety Platform

Paulo Bezerra