Output Window

straton user guide – Rev. 2

sales@straton-plc.com

CP_R31_006_H

straton

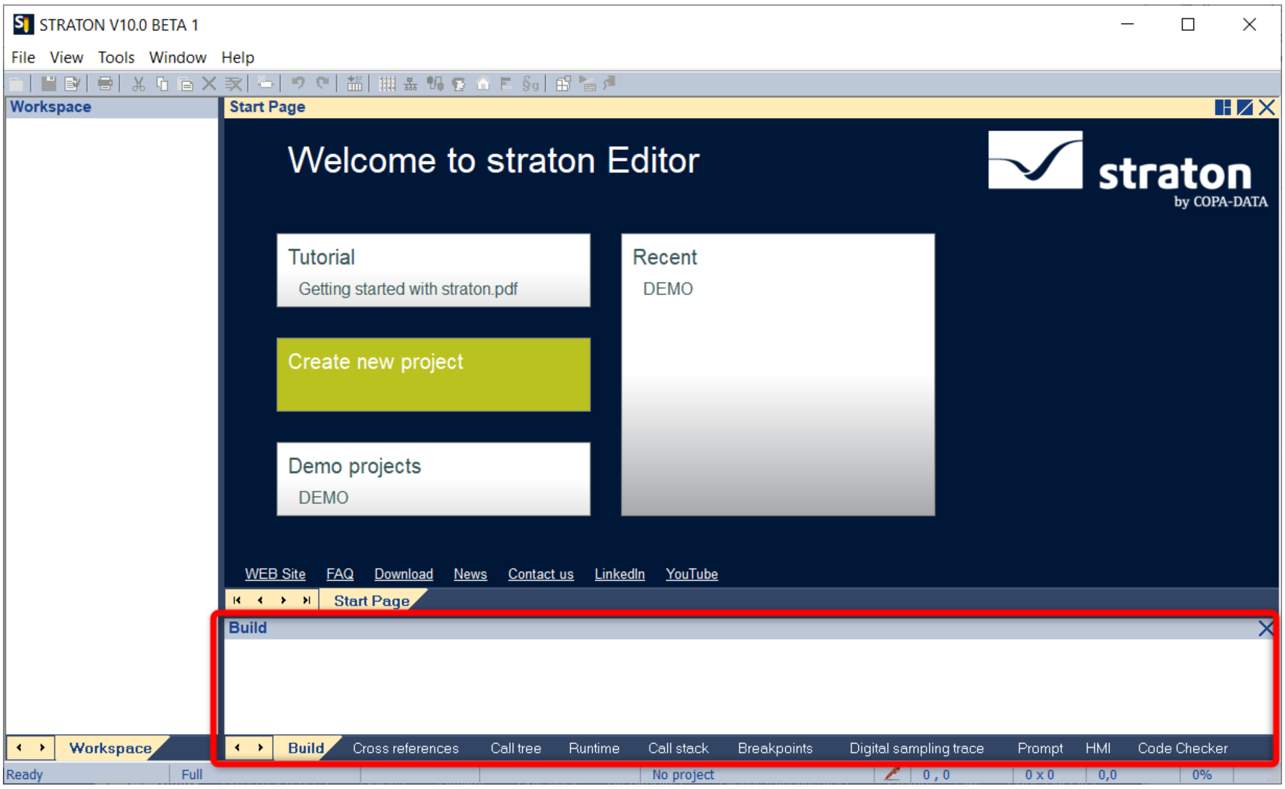
Content

1. Overview

This document describes how to use Output window with straton Editor.

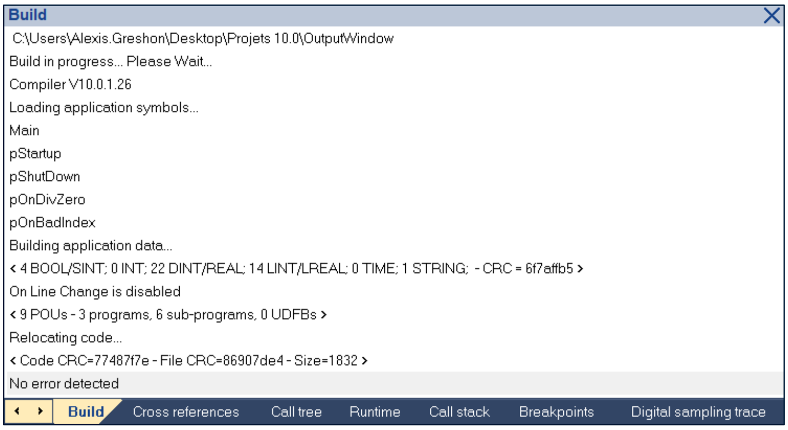
1.1. straton output window

The output window presents the most important diagnostic tools of straton Editor.


STRATON V10.0 BETA 1
File View Tools Window Help
Workspace
Start Page
Welcome to straton Editor
straton by COPA-DATA
Tutorial
Getting started with straton.pdf
Recent
DEMO
Create new project
Demo projects
DEMO
WEB Site FAQ Download News Contact us LinkedIn YouTube
Start Page
Build
Workspace
Build Cross references Call tree Runtime Call stack Breakpoints Digital sampling trace Prompt HMI Code Checker
Ready Full No project 0 , 0 0 x 0 0,0 0%

2. Tutorial

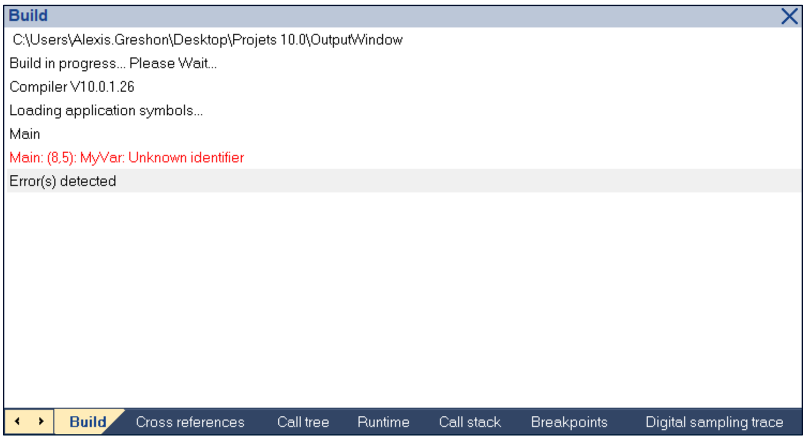
2.1. Build



```
Build
C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow
Build in progress... Please Wait...
Compiler V10.0.1.26
Loading application symbols...
Main
pStartup
pShutDown
pOnDivZero
pOnBadIndex
Building application data...
< 4 BOOL/SINT; 0 INT; 22 DINT/REAL; 14 LINT/LREAL; 0 TIME; 1 STRING;  - CRC = 6f7affb5 >
On Line Change is disabled
< 9 POUs - 3 programs, 6 sub-programs, 0 UDFBs >
Relocating code...
< Code CRC=77487f7e - File CRC=86907de4 - Size=1832 >
No error detected
```

(Build tab)

The Build tab content informs the user of all programs built by the compiler and also several information about the project, the straton version number, the number of variables declared in the project, the number of programs created in the project and it also displays the errors and the warnings in a project:



```
Build
C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow
Build in progress... Please Wait...
Compiler V10.0.1.26
Loading application symbols...
Main
Main: (8,5): MyVar: Unknown identifier
Error(s) detected
```

(e.g.: a variable not declared in straton but used in a program)

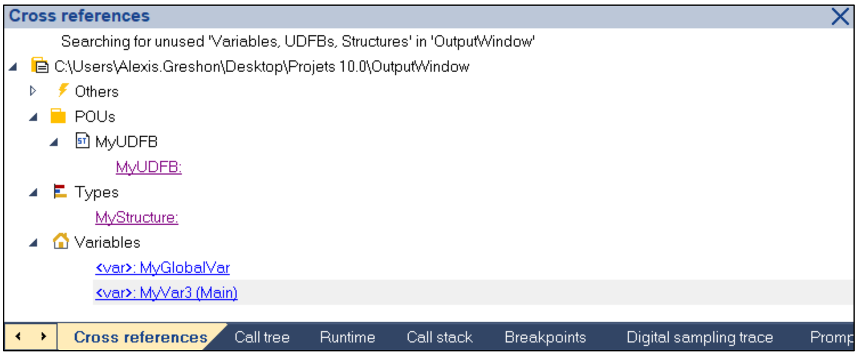
2.2. Cross references

The Cross references tools enable you searching items in the whole project.

This is a list of possible uses of the Cross references tool:

LIST UNUSED ITEMS:

Use the Edit -> Find -> List Unused Items... -> Select the items you want to list ... menu command in order to display the list of items that are not used in the programs of the application. You can click on an item to open it. This command is particularly useful when cleaning a project.

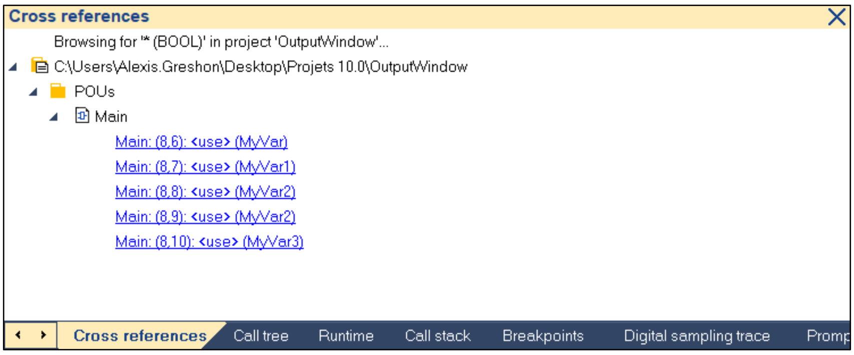


Cross references

Searching for unused 'Variables, UDFBs, Structures' in 'OutputWindow'
- C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow
  - Others
  - POUs
    - MyUDFB
      - MyUDFB:
  - Types
    - MyStructure:
  - Variables
    - <var>: MyGlobalVar
    - <var>: MyVar3 (Main)

FIND VARIABLES:

Use the right click in the cross references output -> Find Variables... in order to find a variable in the project. You can search by the name or by type of the variable you are looking for. You can click on an occurrence to open the program where the variable is used.

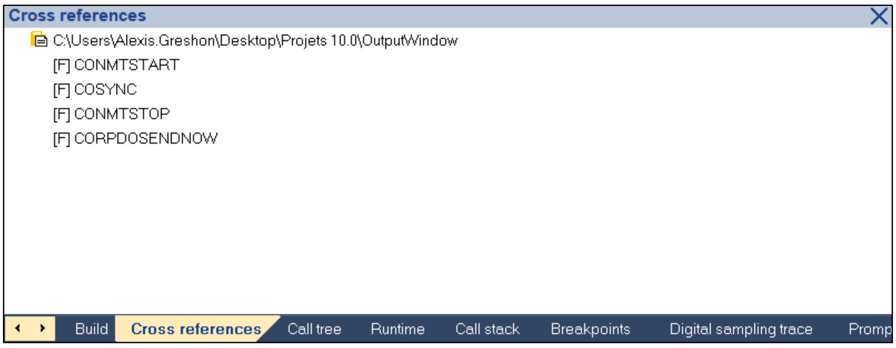


Cross references

Browsing for '*' (BOOL)' in project 'OutputWindow'...
- C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow
  - POUs
    - Main
      - Main: (8,6): <use> (MyVar)
      - Main: (8,7): <use> (MyVar1)
      - Main: (8,8): <use> (MyVar2)
      - Main: (8,9): <use> (MyVar2)
      - Main: (8,10): <use> (MyVar3)

LIBRARY ELEMENTS:

Use the Edit -> Find -> Library Elements menu command to list the I/O devices, functions and function blocks written in “C” that are used in your application.



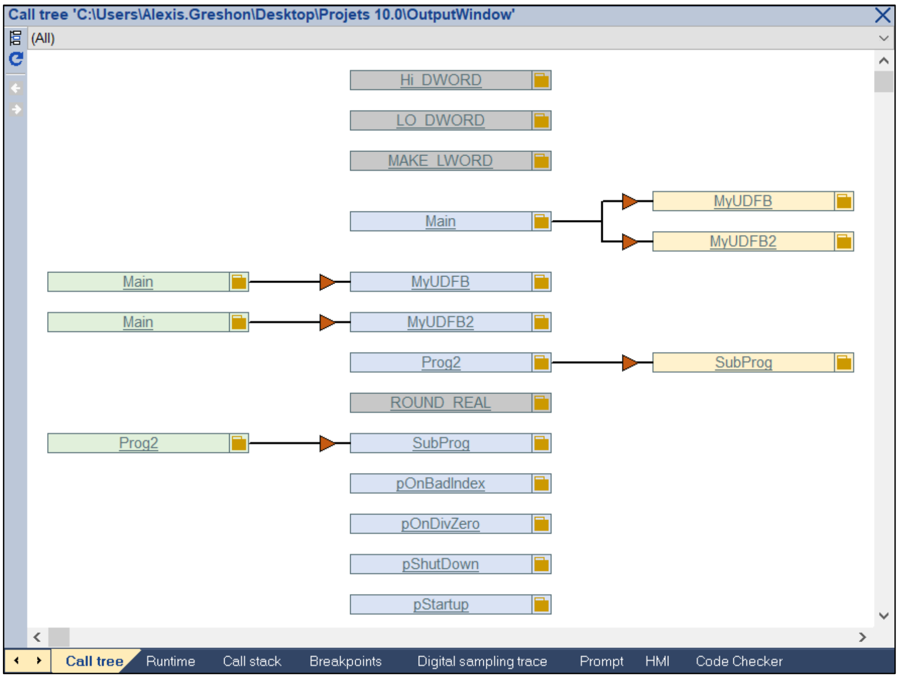
Cross references

```
C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow
    [F] CONMTSTART
    [F] COSYNC
    [F] CONMTSTOP
    [F] CORPDOSENDNOW
```

2.3. Call Tree

The Call Tree view allows to see which program is calling other programs.

In order to see the tree, open the Call Tree from the Output window then click on “Call Tree” button:

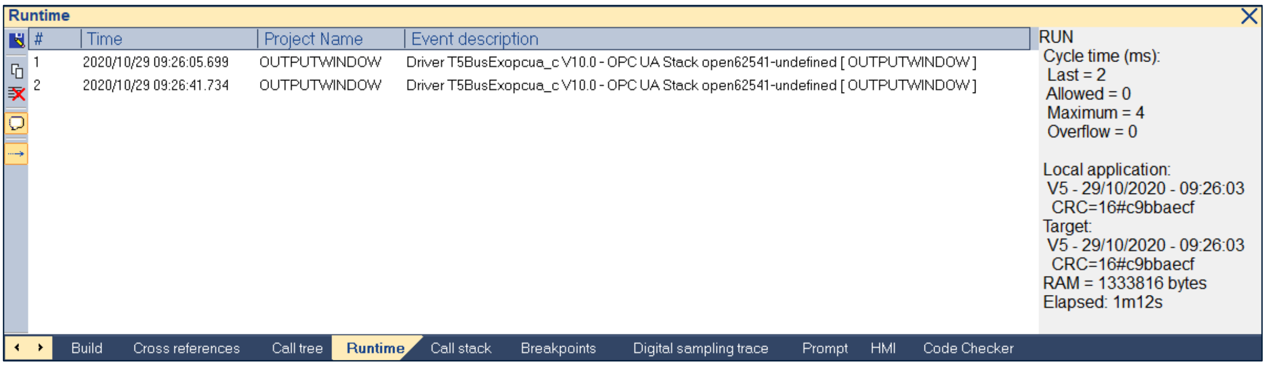


Call tree 'C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow'

(A tree of a project with UDFBs and Sub-Programs which are called by other programs)

2.4. Runtime

The Runtime tab of the output window shows Runtime information when online connected to a target system.



#	Time	Project Name	Event description
1	2020/10/29 09:26:05.699	OUTPUTWINDOW	Driver T5BusExopcua_c V10.0 - OPC UA Stack open62541-undefined [OUTPUTWINDOW]
2	2020/10/29 09:26:41.734	OUTPUTWINDOW	Driver T5BusExopcua_c V10.0 - OPC UA Stack open62541-undefined [OUTPUTWINDOW]

```
RUN
 Cycle time (ms):
  Last = 2
  Allowed = 0
  Maximum = 4
  Overflow = 0

 Local application:
  V5 - 29/10/2020 - 09:26:03
   CRC=16#c9bbaecf
 Target:
  V5 - 29/10/2020 - 09:26:03
   CRC=16#c9bbaecf
 RAM = 1333816 bytes
 Elapsed: 1m12s
```

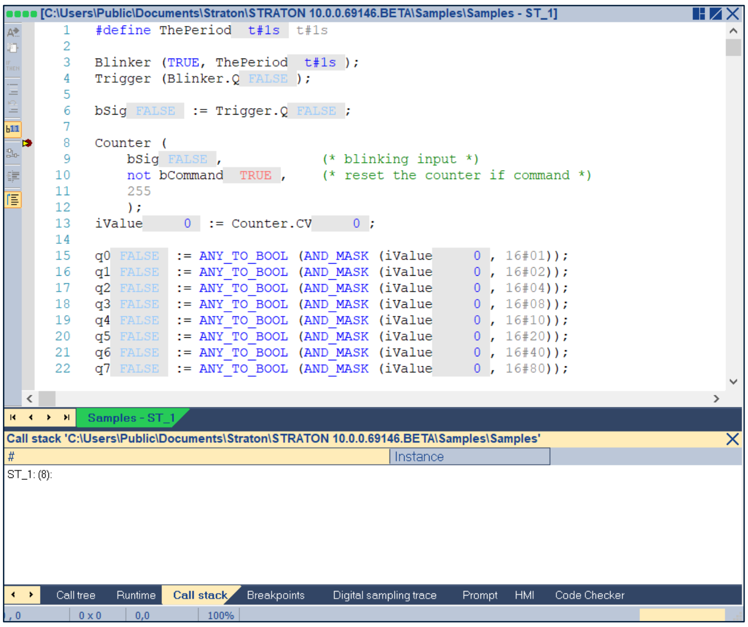
🗨 : If the checkbox “Show RT information” is activated, the information from the runtime will be shown in the output window.

- Some other information is available like the Cycle time of the application on the right side of the output window.

NOTE: the ”Maximum“ cycle time reached by the application as well as the number of cycle-time “Overflow” is reset if the cycle-time is changed while the application is running.

2.5. Call Stack

When you are in debugging mode, the **Call Stack** view allows to see where the program has been stopped on a breakpoint.


[C:\Users\Public\Documents\Straton\STRATON 10.0.0.69146.BETA\Samples\Samples - ST_1]
Samples - ST_1
Call stack 'C:\Users\Public\Documents\Straton\STRATON 10.0.0.69146.BETA\Samples\Samples'
Call tree
Runtime
Call stack
Breakpoints
Digital sampling trace
Prompt
HMI
Code Checker
, 0
0 x 0
0,0
100%

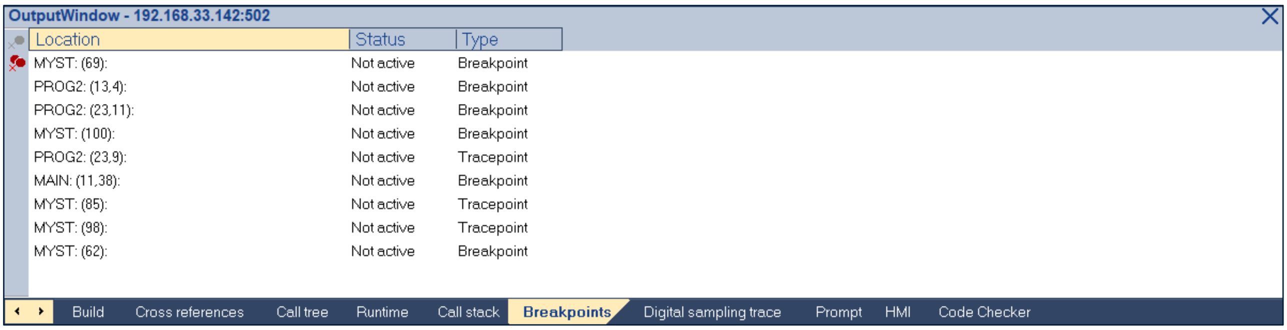
```
#define ThePeriod  t#1s  t#1s

Blinker (TRUE, ThePeriod  t#1s );
Trigger (Blinker.Q FALSE );

bSig FALSE  := Trigger.Q FALSE ;

Counter (
    bSig FALSE ,             (* blinking input *)
    not bCommand  TRUE ,     (* reset the counter if command *)
    255
    );
iValue      0  := Counter.CV      0 ;

q0 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#01));
q1 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#02));
q2 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#04));
q3 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#08));
q4 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#10));
q5 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#20));
q6 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#40));
q7 FALSE  := ANY_TO_BOOL (AND_MASK (iValue      0 , 16#80));
```

#	Instance
ST_1: (8):	

If you double-click on an instance straton will opens where the program stopped.

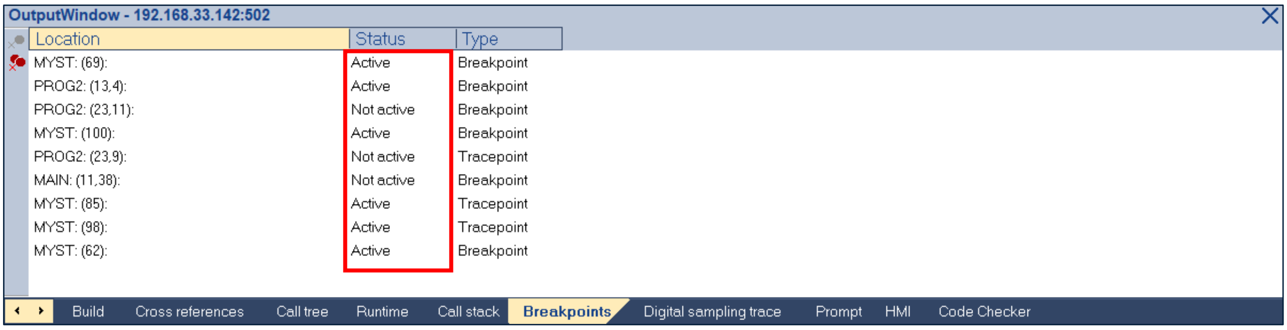
2.6. Breakpoints

Breakpoints tab displays an overview of all **breakpoints** or **tracepoints** set in the project.



Location	Status	Type
MYST: (69):	Not active	Breakpoint
PROG2: (13,4):	Not active	Breakpoint
PROG2: (23,11):	Not active	Breakpoint
MYST: (100):	Not active	Breakpoint
PROG2: (23,9):	Not active	Tracepoint
MAIN: (11,38):	Not active	Breakpoint
MYST: (85):	Not active	Tracepoint
MYST: (98):	Not active	Tracepoint
MYST: (62):	Not active	Breakpoint

When you are in debugging mode or online, the status of a breakpoint / tracepoint can be visible (Active or Not active) in the breakpoints tab.



OutputWindow - 192.168.33.142:502

Location	Status	Type
MYST: (69):	Active	Breakpoint
PROG2: (13,4):	Active	Breakpoint
PROG2: (23,11):	Not active	Breakpoint
MYST: (100):	Active	Breakpoint
PROG2: (23,9):	Not active	Tracepoint
MAIN: (11,38):	Not active	Breakpoint
MYST: (85):	Active	Tracepoint
MYST: (98):	Active	Tracepoint
MYST: (62):	Active	Breakpoint

You can double click on a breakpoint / tracepoint to open it in the project.

There is also the possibility to remove one or several breakpoint directly from the breakpoint tab.

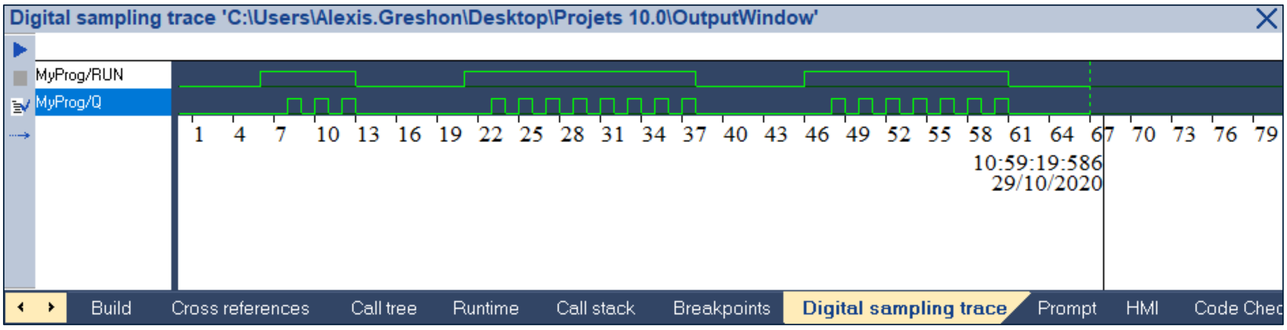
2.7. Digital sampling trace

The runtime system includes a digital sampling trace recorder. The recorder is used to register periodically the state of up to 8 Boolean variables. Samples can be registered either on each cycle or according to a configurable period. The digital sampling trace is a useful tool for tracking Boolean events in the runtime application.

SETTINGS:

Before starting a recording, you need to setup the parameters for the recorder in the Setup Sampling dialog. This includes the list of spied variables, a period, start and stop conditions.

Press the **Set** button to validate the settings.



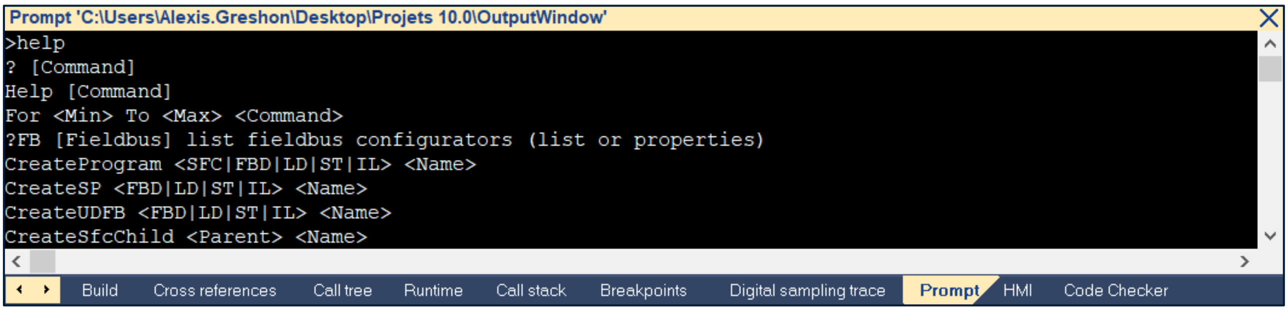
Digital sampling trace 'C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow'

(eg : Trace the Boolean variables of a PLS block)

2.8. Prompt

straton Editor includes a powerful console where you can run some operations in command line text mode instead of using menus.

Open the Prompt tab, then enter the command “?” to get the list of available command and their syntax.

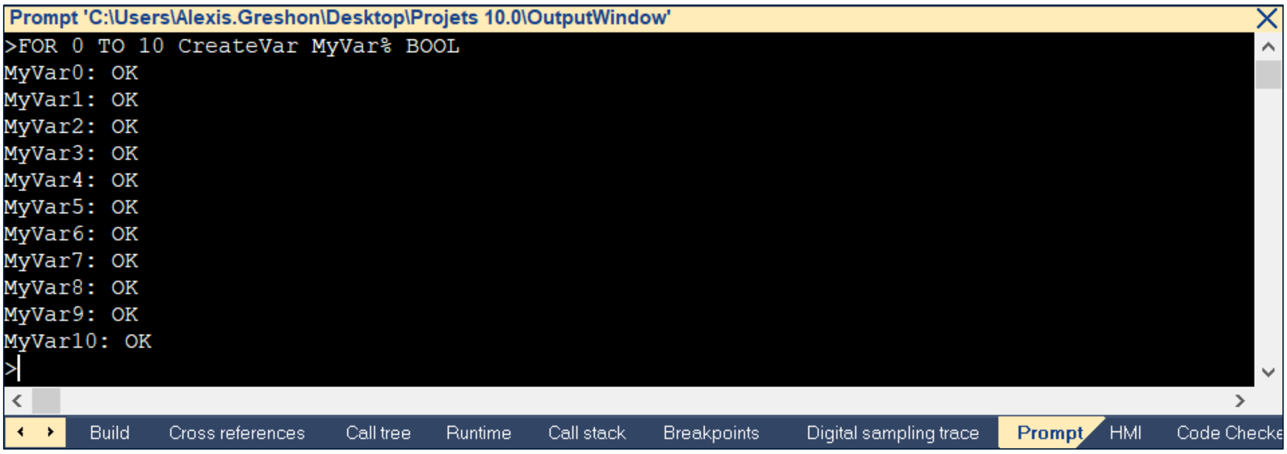


Prompt 'C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow'

```
>help
? [Command]
Help [Command]
For <Min> To <Max> <Command>
?FB [Fieldbus] list fieldbus configurators (list or properties)
CreateProgram <SFC|FBD|LD|ST|IL> <Name>
CreateSP <FBD|LD|ST|IL> <Name>
CreateUDFB <FBD|LD|ST|IL> <Name>
CreateSfcChild <Parent> <Name>
```

EXAMPLE WITH CREATION OF 10 BOOLEAN VARIABLES:

Enter the command: “`For 0 To 10 CreateVar MyVar% BOOL`”



Prompt 'C:\Users\Alexis.Greshon\Desktop\Projets 10.0\OutputWindow'

```
>FOR 0 TO 10 CreateVar MyVar% BOOL
MyVar0: OK
MyVar1: OK
MyVar2: OK
MyVar3: OK
MyVar4: OK
MyVar5: OK
MyVar6: OK
MyVar7: OK
MyVar8: OK
MyVar9: OK
MyVar10: OK
>
```

The 10 boolean variables have been declared in straton’s Dictionnary.

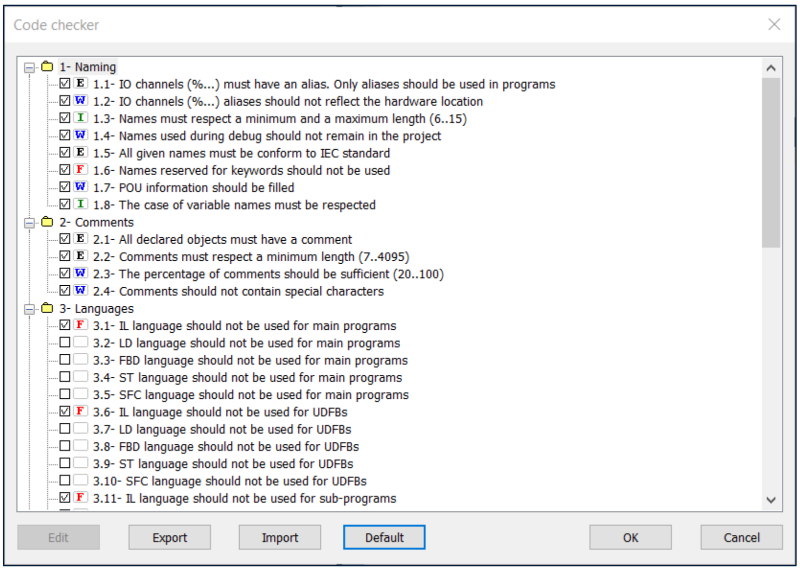
2.9. HMI

The HMI tab is OEM specific.

2.10. Code Checker

The code checker tool allows the user to define and configure rules for his project.

- Open the code checker setting dialog using this button in order to define the code checker settings.



1- Naming
- ☑ E 1.1- IO channels (%...) must have an alias. Only aliases should be used in programs
- ☑ W 1.2- IO channels (%...) aliases should not reflect the hardware location
- ☑ I 1.3- Names must respect a minimum and a maximum length (6..15)
- ☑ W 1.4- Names used during debug should not remain in the project
- ☑ E 1.5- All given names must be conform to IEC standard
- ☑ F 1.6- Names reserved for keywords should not be used
- ☑ W 1.7- POU information should be filled
- ☑ I 1.8- The case of variable names must be respected

2- Comments
- ☑ E 2.1- All declared objects must have a comment
- ☑ E 2.2- Comments must respect a minimum length (7..4095)
- ☑ W 2.3- The percentage of comments should be sufficient (20..100)
- ☑ W 2.4- Comments should not contain special characters

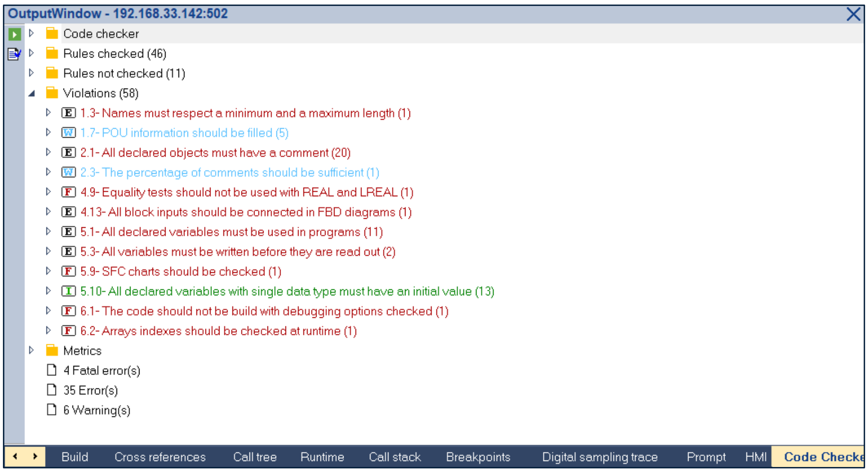
3- Languages
- ☑ F 3.1- IL language should not be used for main programs
- ☐ 3.2- LD language should not be used for main programs
- ☐ 3.3- FBD language should not be used for main programs
- ☐ 3.4- ST language should not be used for main programs
- ☐ 3.5- SFC language should not be used for main programs
- ☑ F 3.6- IL language should not be used for UDFBs
- ☐ 3.7- LD language should not be used for UDFBs
- ☐ 3.8- FBD language should not be used for UDFBs
- ☐ 3.9- ST language should not be used for UDFBs
- ☐ 3.10- SFC language should not be used for UDFBs
- ☑ F 3.11- IL language should not be used for sub-programs

(Code checker settings)

In this settings, you will be able to configure your own code checker settings by checking what you want.

You can also edit the rules by clicking on the rule you want to edit -> Edit.

Once you have finish to edit the rules, you can scan your project… straton will displays all the fatal error(s), error(s), warning(s) and information(s) in the application.



- Code checker
- Rules checked (46)
- Rules not checked (11)
- Violations (58)
  - E 1.3- Names must respect a minimum and a maximum length (1)
  - W 1.7- POU information should be filled (5)
  - E 2.1- All declared objects must have a comment (20)
  - W 2.3- The percentage of comments should be sufficient (1)
  - F 4.9- Equality tests should not be used with REAL and LREAL (1)
  - E 4.13- All block inputs should be connected in FBD diagrams (1)
  - E 5.1- All declared variables must be used in programs (11)
  - E 5.3- All variables must be written before they are read out (2)
  - F 5.9- SFC charts should be checked (1)
  - I 5.10- All declared variables with single data type must have an initial value (13)
  - F 6.1- The code should not be build with debugging options checked (1)
  - F 6.2- Arrays indexes should be checked at runtime (1)
- Metrics
- 4 Fatal error(s)
- 35 Error(s)
- 6 Warning(s)