

Automation & Testing Suite for embedded software / AUTOSAR compatible

# ATS 5.0.0

# **Basic Usage Manual**

SCHLEISSHEIMER SOFT- UND HARDWAREENTWICKLUNG GMBH

www.automation-testing-suite.com www.schleissheimer.com



2022

## Contents

Chapter 1. Getting started
1.1. Project creation4
1.2. Opening existing project
1.3. Removing project
1.4. Saving project9
Chapter 2. Testing files
2.1. Building SimuDLL10
2.2. Adding new test
2.3. Modifying a test
2.3.1. Sequences16
2.3.2. Range values17
2.4. Running tests
2.4.1. Charts
2.5. Importing/exporting tests
2.6. Modifying SimuDLL project
Chapter 3. Additional features of CPP Tests
3.1. ATS Reports25
Chapter 4. Code Generator
4.1. Scripts Control
4.2. Files Control
List of Figures

## **Chapter 1. Getting started**

In this chapter, you will be introduced with activating the license, creating new project and opening the existing one, as well as saving it.

First, run the application. On your screen, the License Tool window will appear (Figure 1). You can import license by loading a V2C file or by inputting the activation key. After successfully activating the license, application needs to be restarted.



License key	Expiry date	Support	Base	Basic	ode Coveraç	Swatt	nabled in V
			Select	license			
nse Update							
	ivation key "xxx	00000-0000-0000	-x000x-x000000x	ooooox" Se	lect a V2C file		

After restarting the application, you will need to select the particular license, which you want to use. To do that, click it and then just click the button *"Select license*". Now, on your screen there will be showed a logging view. After you log in, you can go further to Welcome Window (Figure 2).

Figure 2. Welcome Window



With this window, you are able to:

- open selected project,
- load project from file it allows to open a project by manual selecting a particular .*ats5prj* file,
- create new project,
- remove selected project from list,
- remove missing project it removes a project from ATS recent projects list, that is not existing anymore on your computer (for example a project that has been deleted),
- change license it shows the License Tools
- close application.

## **1.1. Project creation**

In the first step, select a place for your new project and type in the name. It will be saved as *.ats5prj* file. After creating a project, this is how main view of the application looks like (Figure 3).

#### Figure 3. Main View of ATS5

C:/Users/Oliwia/Desktop/TestProject.ats5prj*					
File Simulation Code Coverage View Tools Help					
CPP Tests Code Generator					
🕫 🕄 🛍 🧤 💱 🏹 🗛 🗛 🖻 🖾 🕅					
Stubs viewer @8	3		1	Comment	ØX
	Name	Туре	Value		
	4				
log					Ø 🕱
2022-02-23 09:32:11 View restored. 2022-02-23 09:32:38 Project correctly loaded. 2022-02-23 09:32:39 Looking for MsBuild path. 2022-02-23 09:32:39 Looking for CL Path path.					

Before you start having the files analyzed by ATS, please make sure, that you have set a path to MSBuild and Cl.exe. To check this, go to *Tools* – *Configuration* – *Compilation tools*, as showed on Figure 4. If the fields are empty, use *Suggest* button to set them automatically. In case it does not happen automatically, you will have to set it manually (choose a particular path to those components or install them, if you have not done it yet).



* Configuration - Compilation to	pols		?	×
Type to find	Compilation	tools		
<ul> <li>Application</li> <li>General</li> </ul>	MsBuild.exe	C:/Program Files/Microsoft Visual Studio/2022/Community/Msbuild/Current/Bin/amd64/MSBuild.exe	Sugges	it
Compilation tools Appearance Company C++ Project Desiret	Cl.exe	C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/14.29.30133/bin/HostX64/x64/d.exe	Sugges	it 🗌
Project Database CTC Reports				
Code Generator				

Now, you can start analysing source files and creating tests. To choose files for analysis, click the first left button on the Toolbar (or use CTRL+W key shortcut):

It will display a dialog window with such features, as showed on Figure 5. In here, you can select:

- a way of importing files (by *Visual Studio Solution/Project*, by *Project Root Folder* or by *Source files*),
- importing method (*Replace existing stubs* or *Append to existing stubs*),
- language standard.

Figure 5. Importing files to CPP Tests

😻 ATS5		×	🛷 ATS5	×
Import files for	testing		Import files for	testing
Import:	Visual Studio Solution/Project	•	Import:	Source Files 💌
Path:			Add files:	Append new items to tree 🔹 🛄
Import method:	Replace existing stubs	•	Import method:	Replace existing stubs
Additional includes path:		^	Additional includes path:	· · · ·
Select language standard:	c++14 (default)	•	Select language standard:	c++14 (default) 👻
			Selected source files tree:	
	< <u>B</u> ack <u>N</u> ext >	Cancel		< <u>Back</u> Next > Cancel

Additionally, in here you can also see the path for selected files (if the import way is Visual Studio Solution/Project or Project Root Folder) or select a method for adding files (if the import way is Source Files). The options for that last case are *Append new items to tree*, *Override all items in the tree*.

**Warning**: since now, you are only able to parse files that are using basic variable types. Any other types will cause and display errors.

To describe and clarify the ways of importing files, please get familiar with this information:

- Visual Studio Solution/Project it allows to choose .*sln* or .*vcxproj* files, so you can display files that are included in it.
- Project Root Folder it allows to choose root folder from which files and subfolders will be displayed for further analysis.
- Source Files it allows to add source files which a user wants to have displayed in tree section (right side of Figure 5). Adding source files is available multiple times when *"Add files"* option is selected.

Besides that, application allows to set additional includes path - it can be done in two ways. The first method is to simply click the button on the right side of the field and type in the paths, which you need. The second method is to click the "…" button and select the output folders manually. By setting additional includes path, you can specify paths to folders with files that are needed to be included in analysis, and that are placed outside the project.

By going *"Next*", the application would show a selection section (Figure 6). In here please choose files, using checkboxes, that you would like to have in your project.



Figure 6. Selection section in CPP Tests

The last step is to confirm all selected files. Click *"Finish"* to finalize the process of importing files and to display them in a main view (Figure 7).

Figure 7. Main View of ATS5 with imported project

C:/Users/Oliwia/Desktop/Nowy folder/test1.ats5pri*	—					- 6	×
File Simulation Code Coverage View Tools Help							
CPP Tests Code Generator							
😥 🖾 🔝 🗛 🗛 🔁 🖬 👘 🕼 🐨							
Stubs viewer @ R					Global Variabl	5	08
	Name	Type	Value			News	Make and
Name Result					lype	Name	Value
ATS_CinTests					THE	giobalvariable	2
Gill AIS_ClassDisabledConstructor     GassDisabledConstructorWorkingParam					2 long	globalVariable2	-
ATS_CppTestingPrj					3 double	globalVariable3	-
ATS_CppTestingPri_1     ATS_CTPTCTTA_ANO					4 float	globalVariable5	-
GIAISTESTCLANG     GIAISTESTCLANG     GIAISTESTCLANG							
B Global Functions							
Log							8
2022-03-09 22:17:47 Project correctly loaded. 2022-03-09 22:17:47 Correctly loaded project from db files							<u></u>
2022-03-09 22:17:47 2022-03-09 22:17:47 CodeGen: Config reset.							
2022-03-09 22:28:52 ptrToInt method from ClassWithRefsToPrimitives class couldn't be added to tree. U 2022.03.09 22:28:52 ptrToConstint method from ClassWithRefsToPrimitives class couldn't be added to t	nhandled parameter type. (int ") ree. Uphandled parameter type. (const int ")						
2022-03-09 22:28:52 ptrToConstIntx2 method from ClassWithRefsToPrimitives class couldn't be added to	o tree. Unhandled parameter type. (int *const	)					
2022-03-09 22:28:52 constPtrToConstInt method from ClassWithRefsToPrimitives class couldn't be adde 2022-03-09 22:28:52 clobalek global variable couldn't be added to project. Unhandled type, (const char '	ed to tree. Unhandled parameter type. (const	int "const)		Aktywuj system W			
2022-03-09 22:28:52 Correctly prepared SimudDllx64 project	<i>i</i>						

## **1.2. Opening existing project**

If you already have created a project and now you would like to open it, you can do that by:

- opening selected project from Recent projects list (Figure 8), or
- loading a project from a file.

Figure 8. Recent projects list in Welcome Window



## 1.3. Removing project

If you have deleted a project, or moved it to other folder, you could see this project as disabled element on the list (Figure 9). To remove that element, simply select this project and then click the button *"Remove missing project"*.

Figure 9. Remove missing project in Welcome Window



In case you would like to remove a particular project from Recent projects list, you can do this by selecting it and clicking *"Remove selected project from list"*.

## **1.4. Saving project**

To save a project you can go to *File* and select *"Save project"* (if your intent is to overwrite the existing project file) or *"Save as"* to save but simultaneously create new project file.

File Simulation	Code Coverage	View	Tools	Help
🚯 New project	Alt	+N		
칠 Load project	Alt	+L		
💾 Save Project	Alt	+ S		
🗎 Save as	Ctr	·I+Alt+S	hift+S	
🛞 Exit				

Another way to save project is by using this button from Toolbar:



Figure 10. File – Save options

A user can specify a saving method in *Tools – Configuration – Database*. The options are saving tests as JSON files or saving them in database MongoDB (Figure 11). In this second case, it is required to have MongoDB software to save tree in database.

Figure	11.	Configuration	-	Database
--------	-----	---------------	---	----------

pe to find	Database		
Application	Save method		
Compilation tools	O MongoDb		
Appearance Company	<ul> <li>Save to file</li> </ul>		
C++ Project Project Database		mongodb://localhost:27017	
CTC			
Reports Code Generator	Database name	▼ Add databa	se
	Database file	C:\Users\Oliwia\Desktop\Nowy folder\emptyProject\db .	

## **Chapter 2. Testing files**

In this chapter, you will get to know how to build DLL, prepare your files for analysis, create tests and how to run them.

### 2.1. Building SimuDLL

Now, when you have opened a project or created a new one, there is only one more step to do before testing your files. This step is to build the DLL. It can be done by clicking the third button on the left:



However, before building it, you should decide whether you would like to have it built with CTC enabled or not. If yes, go to the Code Coverage tab and tick the checkbox *"Build SimuDLL with CTC*" (Figure 12).

Figure 12. Code Coverage tab



If you decided to build DLL with CTC enabled, you can set CTC options in *Tools - Configuration – CTC* menu (Figure 13).

```
Figure 13. CTC Tools
```

Configuration - CTC				?	×
Type to find	стс				
<ul> <li>Application</li> <li>General</li> </ul>	Code coverage report type				
Compilation tools Appearance	Statement coverage	O Decision/Branch coverage	MC/DC coverage		
<ul> <li>C++ Project</li> </ul>	CTC report generation				
Project Database	Generate TXT report	Auto-open	after generation		
Reports Code Generator	Generate XML report	Auto-open	after generation		
	Generate HTML report	Auto-open	after generation		
	Source code editor coloring option				
	Accurate	Expanded			
	CTC additional options				
	Analyse header files				
	CTC report threshold		100 %		
				_	
			Save configuration	Can	cel

Last important step to take, is to make sure that all constuctors and destructors are defined correctly.

Constructor and destructor methods are methods, which are used to create and destroy objects of the class with tests. By default, these methods are defined without any parameters, in the way showed on Figure 14.

Figure 14. Defining constructors with non-params

```
ATS_CppTestingPrj:Construct
1 ATS_CppTestingPrj::construct()
2 {
3 this->object = new ATS_CppTestingPrj();
4 };
```

However, in some cases there is a necessity to define them with parameters. In such situations, if the application recognizes it, application will display an information, as shown on Figure 15.

Figure 15. Information while recognizing constructor with parameters

```
ATS_ClassDisabledConstructor::construct()
2 {
3 //Public default constructor required to create stub object not available.
4 //this->object = new ATS_ClassDisabledConstructor(/* Required parameters */);
5 //Comment error below after implementing class constructor
7 #error ATS5: Public default constructor required to create stub object is not available.
8 };
```

After all is set up, successful building the DLL will display a dialog with confirmation (Figure 16). On the other hand, if something fails you will get errors displayed in a log window at the bottom of application with details – what went wrong.

Figure 16. Successfully built DLL notification



After choosing files to analyze and compile the DLL now you are ready to test them.

#### 2.2. Adding new test

Adding test to adapters (tree items named as class methods or global functions) is possible in three ways. First one is to simply double-click on adapter (this option is available only when adapter does not contain any tests yet). Second one is to use context menu on adapter by pressing right mouse button on it (Figure 17).

Figure 17. Adding tests via context menu



And the last, third option is to use the second button from the right side of a Toolbar:

Application allows you to rename test by double-clicking it. Also, there is a possibility to remove test, duplicate it, add sibling test, take a snapshot of it, run it and reset its results.

#### 2.3. Modifying a test

Clicking on test (tree item) shows a new window, that allows user to specify values for input arguments of methods/functions as well as expected return values (Figure 18).

Figure 18. Main View of ATS5 with added tests

File Simulation Code Coverage View Tools Help								
CPP Tests Code Generator								
🕞 🚺 🛍 🤮 💱 🗊								
Stubs viewer	0 X	ATS_CinTests : inputCinToNumber : inputCinToN	lumber_Test1		Sequence Length 1 +	Global Variable	is	6 8
		Name	Туре	Value		Type	Name	Value
Name	Result	Input arguments				1 int	globalVariable	5
★ ATS CinTests		<ul> <li>Expected return value</li> </ul>				1 110	giobalvallabie	
Construct			int			2 long	globalVariable2	2
Destroy		* Global Variables						
	-	Expected value				3 double	globalVariable3	_
T testCinFailed		<ul> <li>User Variables</li> </ul>				4 float	globalVariable5	-
T testCinPassed		Input				4 11040	giobalvallabies	
<ul> <li>InputCinToNumber</li> </ul>	-	Expected value						
T inputCinToNumberFailed								
inputCinToNumber_Test1								
<ul> <li>ATS_ClassDisabledConstructor</li> </ul>								
Construct								
Destroy								
▼ (t) add	-							
T addPassed	-							
T add_Test1								
T add_Test11	-							
(II) add2	-					Comment		6 X
ATS_ClassDisabledConstructorWorkingParam	1							
Construct								
Destroy								
<ul> <li>addToParam</li> </ul>						-		
T addToClassVar	-	Input stream						

Application allows user to input only parameters that are used in a specific method/function. For example, for *add* method, which returns integer and its parameters could be also only integer number, there will be error (marked as red background), if user tries to input other data types (Figure 19).

Figure 19. Setting wrong data type for a test parameter



Global variables can be added by selecting them from the expanding list (Figure above). To use a user variable, first you need to create it in the window with class definition (Figure 20). In the bottom section, double-click User Variables field or right-click it to open an option *"Add variable"*. Now, define the type, name and value of this variable, then push Enter to confirm your inputs. Now, to use such variables, the SimuDLL needs to be rebuild.

Figure 20. Adding user variable section

CPP Tests Code Generator					
🕞 📑 📲 💱 🖍 🖓	Aã Aă 🖻				
Stubs viewer	6 ×	ATS_CinTests			*
		Property		Value	
Name	Result	Name	ATS_CinTests		
→  → ATS_CinTests		Public constructor	1		
Construct		File location:	C:/Program Files/ATS/ATS_(	CPPProjectTesting/ATS_	CPPProjectTesting/
Destroy		Template kind:	None		
▼ III firstInCin	-	1			
T testCinFailed					
T testCinPassed	-				
• E inputCinIoNumber					
I inputCinToNumberFailed	-				
ATS Class Dischlad Capatruster	-				
Construct		Type Name	Value		
Destroy		User Variables			
v (c) add			Add variable		
T addPassed	-				
T add Test1	-				
T add_Test11	-				
🗔 add_Test2					
🖾 add2	-				
<ul> <li>ATS_ClassDisabledConstructorWorkingParam</li> </ul>					
Construct					
Destroy					
▼ I <sup>(0)</sup> addToParam					

In tests, where a parameter can be a reference (e.g. int &), application allows you to use only user (local) variables or global variables (Figure 21).

Figure 21. Reference to global variable in test's parameter

ClassWithRefsToPrimitives : refToInt : refToInt_Test1									
Name		Туре	Value						
<ul> <li>Input arg</li> </ul>	uments								
ref		int &	globalV	ariable					
<ul> <li>Expected</li> </ul>	return value								
		int	8						

If you declare user variable as a pointer, for example in this way: *(type)* int\* *(name)* ptr *(value)* nullptr, it can be then used in test's parameters like this:

Figure 22. User variable as a pointer used in test's parameter

ATS_CppTestingPrj : add : add_Test1								
Name	Type	Value						
<ul> <li>Input arguments</li> </ul>								
a	int &	*ptr						
b	int	1						
с	int	1						
<ul> <li>Expected return value</li> </ul>								
	int	8						

There is also a possibility to use pointers in arguments that are not using references. For example, you can define user variable as a pointer to integer and

then use it as a parameter in argument of int type – simply use \*userVar or userVar[0].

On the other hand, if you want to set value or set expected value of user variable which is pointer type, you can only do that by typing "\*" before variable's name or array index after name like ptrVar[0].

Figure 23. Setting or getting pointer user variable

<u>r</u>		
🔻 User Variables		
<ul> <li>Input</li> </ul>		
ptr[0]	int*	1
<ul> <li>Expected value</li> </ul>		
*ptr	int*	1

#### 2.3.1. Sequences

A particular test can be run in sequences. To add new sequence, click the ,,+" button on the right side of the fields with test params (Figure 24). Also, you can modify the amount of sequences by putting its length by number.

Figure 24. Test sequences



Sequences will be added as well, if you input values in parameters field, separated by semicolon (;). For example, when user types "1;2;3" into value column, it means that the test will be executed three times. Adding additional semicolons without values after them means that we take previous sequence value. Thus, having "1;2;3;;;" means that the test includes 6 sequences, where 3 last sequences will consists of input values equal 3 (1;2;3;3;3).

To sum it up, if you do not define parameters in the following sequences, they will be automatically set as values of earlier defined parameters. For example, as presented on Figure 23, expected return value of user variable *userVarTest* will be always equal 2 in every sequence. But as input for *userVarTest* the value will be "1" for the first and the second sequence, and then in the third and the fourth sequence the value will be "3".

In test, you can also add comments, which will be displayed in generated reports.

#### 2.3.2. Range values

ATS5 allows users to create tests with range values in parameters. Range is specified as [min, max, step]. To use a range, you have to put your values between square brackets "[" and "]", with comas as separator for min and max value and a step (which is optional, by default it will be set to 1). Ranges are presented on Figure 25.

Figure 25. Ranges in tests' parameters

Name	Туре	Value
<ul> <li>Input arguments</li> </ul>		
a	int	[10,50]
b	int	[50, 10, -5]
- Expected return	value	
	int	>0
- Global Variables		
Input		
Expected value		

Another example – range specified as [5,10,2] will run test with given values "5;7;9". If user puts two ranges for the same input argument and in the same sequence, application will combine them, using Cartesian product operation. It is also possible to have range with a negative step. This requires putting a bigger value as a minimum parameter than maximum parameter (e.g. [15, 2, -3]).

Important information about ranges is that they differ as a parameter for return values. For them (return values), you can only specify [min, max] params

(without step). It means that return values specified in a range (e.g. [5,150]), will take every value from that range as positively passed in a test.

Figure 26. Ranges in return values

Name	Type	Value
<ul> <li>Input arguments</li> </ul>		
a	int	143 150
b	int	1 1
<ul> <li>Expected return value</li> </ul>		
	int	[5,150]

As showed on Figure 26, test from the first sequence will be passed (the result is 144, so it contains in given range), but test from the second sequence will fail.

Moreover, application allows you to use special characters for specifying return value. Those characters are:

"<" - values less than;

">" - values bigger than;

"<=" - values less and equal to;

">=" - values bigger and equal to;

"!" - negation (it means that user can expect every value except the ones given in return range if exclamation mark was added);

"\*" - all values are correct.

Figure 27. Special characters in ranges

Name	Туре		Va	lue		
<ul> <li>Input arguments</li> </ul>	0.0255550775					
a	unsigned	i	1	2	3	4
b	unsigned	i	2	6	7	9
* Expected return value						
	unsigned	i	>5	* !	25 <	=89
- Global Variables	unsigned	±	/5	1.15	23	-0.

Usage of these special characters is presented on Figure 27.

#### 2.4. Running tests

After filling in all params that you need for your tests, now you can run them. To start one selected test – click the button in Toolbar:

If your mouse's focus will be set to *class* or *adapter*, clicking *Run Selected Test* will cause running all tests from the selected class/adapter.

Running selected test is also possible using context menu, after rightclicking tree item in the Stubs viewer.

If you would like to run all created tests, simply use the button or again – use a context menu.

test is shown in column "*Result*" in the Stubs viewer tree.

After running tests, an informational dialog will appear. It includes such information as: numbers of tests done correctly and incorrectly, name of executed test, status of the test result (Passed/Failed), time in which the test was performed. In case that some test is not executable (for example due to incorrect data types in params), this dialog will also include that information. Also, status of executed

A view with the results of executed tests could be different – it depends on configurations that were set in Tools. Settings concerning generating reports can be checked in *Tools – Configuration – Reports* (Figure 28).

#### Figure 28. Reports Tools

find	Reports		
olication	Folders		
General Compilation tools Appearance	Reports path		<u> </u>
Company • Project Project Database	Image sub-folder		
CTC Reports	General		
Code Generator	Don't show again "Do you wa	nt open generated report?" Auto-	eneration after executing tests
	Charts		
	Chart width [px]		
	Chart height [px] 72 🗘		
	Floating numbers		
	Precision type		
	O Auto	○ Set ma	nually
	Set precision 6		
	Type of saving photos to the repo	ort	
	only base64	<ul> <li>only image files</li> </ul>	$\bigcirc$ use base64 and save image to file
	Type chart render		
	<ul> <li>only separated charts</li> </ul>	<ul> <li>only summary chart</li> </ul>	🔿 all
	sequence table type		
	• auto	vertical	O horizontal
	Struct of page (with the possibility	y to turn on / off)	
	✓ Table of content		
	✓ Images		

In here, you can choose paths for reports, as well as for other images, and establish where they should be stored. By using checkboxes you can decide whether to auto-generate a report after every test execution or not. There is plenty of settings to choose, that will allow you to individualize ATS5.

If you would like to always show charts after test execution, you will find that option in a tab, called View. There is a checkbox *"Enable charts"* (Figure 29).

Figure 29. View tab

File Simulation Code Covera	ge <mark>View</mark> Tools Help	
CPP Tests Code Generator	Show/Hide Log output	
	Enable charts	
🛛 🥂 🕖 DÌL 🐌	🗧 🗃 Clear Log	Alt+Shift+C
Stubs viewer	Restore view	Cir

After setting this on, every executed test will automatically show charts with results.

#### **2.4.1. Charts**

Charts are presenting test's results – they can be very simple or pretty complicated, depending on given parameters value and number of sequences. Charts consist of input arguments, expected return values for variables and actual received return values.





In the middle part of the *Charts* widget, you can find generated chart, tools for manipulating the chart and buttons for exporting the results. On the right side of it, there is a table with parameters' values. These values will be changing in real-time when your mouse will be hovering points on the chart.

The display tools consist of a button for resetting the view and three checkboxes to turn on/off displaying cursor, errors and axis on the chart.

The following two sections concerning axis X and axis Y include tools for changing the scale of displayed chart – you can zoom it in or zoom it out.

Also, there is a slider for moving the graph to right or left (for X axis) and to up and down (for Y axis).

Furthermore, in the section called File, there are three buttons for exporting. The first one is used to generate PNG file with the displayed chart, the second one is used to export the results to CSV file, and the same happens, when user clicks the last button, with the only difference that the file with results will be in a format of JSON.

#### 2.5. Importing/exporting tests

Application allows you to import ready tests (in a format of .JSON files) to the project. It can be done by right-clicking a class or method in Stubs viewer and selecting the option *"Import method tests"/ "Import tests for the whole tree*". Other way to import tests is to select a button *"…*", that is placed next to Sequence Length in tests parameters field or in class/method definition.

Figure 31. Importing/exporting buttons in class definition



Exporting tests is equally simple – you can find this option in context menu of tree items or – after selected a specific test – export it via button, placed next to Sequence Length. As you can see, application allows you to export tests as CSV and also as JSON files.

Figure 32. Importing/exporting buttons in tests' parameters section

CPP Tests Code Generator			
🔎 🖏 DĩL 🦛 💱 🕨 🏷 Ar Az 🖻			
Stubs viewer 2018	ATS_CinTests : inputCinToNumber : inputCinToNumberFailed	Sequence Length 1 + -	
	Name Type	Value	
Name Result	Input arguments		
<ul> <li>ATS CinTests</li> </ul>	<ul> <li>Expected return value</li> </ul>		- es - l
Construct	int	96	D.
Destroy	<ul> <li>Global Variables</li> </ul>		6639 T
+ El fratinCin	Input		<b>A</b>
* IRI incutCinTeNumber	Expected value		<u> </u>

The difference between importing/exporting files in format of JSON or CSV, is that while using .JSON files, the test is imported as a new test, and exported test is also exported as a whole element. Importing by CSV file will cause loading only tests' values, and exporting as CSV file will save only tests' values.

#### 2.6. Modifying SimuDLL project

In case there will appear any error during SimuDLL project compilation, user can open SimuDLL project via ATS application by clicking *Simulation* menu, then *"Open SimuDLL project in Visual Studio*". After that, *.vcxproj* file containing SimuDLL will be opened.

Figure 33. Simulation – Open SimuDLL in VS

File Simulation Code Coverage View Tools Help	
CPF N Open SimuDII project in Visual Studio	
DIL Build SimuDII	
🖉 🌠 Force SimuExe close	
Stubs viewer	0 ×
	Name
Name	Result
▼ ☐ ATS_CinTests	
Construct	

Proper SimuDLL project configuration looks like this:

Figure 34. SimuDLL Configuration

Configuration Properties	✓ General Properties	
General	Output Directory	<pre>\$(SolutionDir)Bin\\$(ProjectName)\\$(Configuration)\</pre>
Advanced	Intermediate Directory	<pre>\$(SolutionDir)Build\\$(ProjectName)\\$(Configuration)\</pre>
Debugging	Target Name	\$(ProjectName)
VC++ Directories	Configuration Type	Dynamic Library (.dll)
▷ C/C++	Windows SDK Version	10.0 (latest installed version)
▷ Linker	Platform Toolset	Visual Studio 2022 (v143)
Manifest Tool	C++ Language Standard	ISO C++17 Standard (/std:c++17)
XML Document Generator	C Language Standard	Default (Legacy MSVC)
Browse Information		
Build Events     Custom Build Ston		
P Custom build Step		

Most important is to specify "*Platform Toolset*" to "*Visual Studio 2022* (*v143*)". Otherwise, there may appear errors during compilation. One of the common errors that appear (if "Platform Toolset" is not specified) is that our

application cannot find included system headers in files that we are trying to analyse.

## **Chapter 3. Additional features of CPP Tests**

Besides main features that were described before, ATS5 has some other functionalities. On the Figure 34 there are buttons marked in red, yellow, green and blue.

Figure 35. T	Foolbar a	dditio	nal fea	itures						
	File Sim	ulation	Code	Coverage	e View	Tools	Help			
	CPP Tests Code Generator									
		<u></u>	ni i	le					Ē	<b>⊡</b> ⊗¶ ⊤+1
	· · · +	03	JLL				-			·o •o

Button in blue frame concerns refreshing project files. It will work, if the application finds any changes in files, that user is currently using in a project (in .h or .cpp files). After clicking the button, if there had been any changes made to the files, the application will update them, remaining all the created tests by user.

Buttons in red area are related to increasing/decreasing font size for constructor and destructor methods' definitions.

Buttons in yellow frame concern generating CTC and ATS reports.

Buttons in green frame concern adding new test (on the left) and deleting selected stub (on the right).

In case of a problem with executing tests, you can force SimuDLL to close. To do that, go to *Simulation – Force SimuExe close* (Figure 36).

Figure 36. Force SimuExe close



Additionally, in Tools tab, you can find an option called *"Load original source*". It is used for restoring imported file to its original version – without any added tests, variables or snapshots. Snapshots are used to make shots of a test, which cannot be modified but they can be used to restore its values.

#### Figure 37. Tools tab



## **3.1. ATS Reports**

Those reports are generated as HTML file. They include Table of contents in the top of the page, then the titles of classes that contain done tests, names of the methods with their test results and their definition titled as Function Code.

Optionally there could be included comments section. In the middle and the bottom of the page there is a chart with a legend of params, and below that the report includes a table with the values.

## **Chapter 4. Code Generator**

Code Generator is a functionality that allows you to load JavaScript files, modify them, create new one, and then use them to generate dynamic code between customizable tags in .hpp, .cpp and .h files.

Figure 38. Code Generator basic button



In Code Generator tab, there are 3 basic buttons. The first on the left is used to run all JS scripts, loaded to a project. The second in the middle is used to save single, selected script. And the last one saves all created or modified scripts.

#### 4.1. Scripts Control

In this tab you are able to load existing JS scripts and open them in application (Figure 39). To load scripts from your computer, click *Add* button. To create new JavaScript script, select *Create*.

Figure 39. Buttons for Scripts Control

2	Scripts Co	ontrol	ð×
Con		JS scripts	
Files		Add Create Remove  ↑	
		Script	
utro	1 🗸	sample.js	
Ŭ S	2 🗸	sample2.js	
cript	3 🗸	sample2Outside.js	
°.	4 ✓	writeToTextFile.js	

If you would like to remove loaded .js file from the Script list, select the item and then click *Remove* button. All scripts on a list will be executed in ascending order (from 1 to n). Down and up arrows buttons allow you to place particular .js file lower or higher on a list, which will cause changes in scripts execution order. Checkboxes are used to enable or disable files from the list, without removing them – if you do not want some script to be executed but you want to save it on a list, simply uncheck the box. In this case, it will not be executed.

The blue circle (Figure 39) placed near the name of a file means that this file has been changed and stays unsaved. It will disappear after saving the script.

Going further, in the middle of the screen (Figure 40) there is a modifiable field with JS code. You can add commands from *Available commands list*, which is placed on the right side – just click the line and area, where you would like to have the command inserted and double-click the needed item from the list.

Figure 40. Scripts Control view in Code Generator

CF	P Tests	Code Generator		
	15} (15)			
trol	Scripts Co	ontrol	© 🗷 C:/Users/Oliwia/Desktop/CodeGeneratorSamples/JS/sample.js	Info Ø 🕅
ol Files Con		JS scripts	<pre>1 var strl = "// some sample generated code\r\n";</pre>	Available commands list
		Add Create Remove  ↑  ↓	<pre>2 strl += "uint8 generatedVar;\r\n"; 3</pre>	<search available="" commands="" in=""></search>
		Script	<pre>4 var str2 = "// other generated code STR2\r\n";</pre>	Array.contains(obj)
b	1 ✓	<ul> <li>sample.js</li> </ul>	5 str2 += "uin32 gen32Var;\r\n";	Array.join(separator)
20	2 🗸	sample2.js	<pre>6 Array.join(separator); CharToInt(ch);</pre>	CharToInt(ch)
-te	3 🗸	sample2Outside.js	/ InsertCode("SAMPLE_1", strl);	Eval(jsCode)
0	4 ✓	writeToTextFile.js	9 // tag start prefix + patam0 + tag start postfix => /* @ATS S	Exec(jsCode)
			10 //SystemExecute("C:\Users\Oliwia\Desktop\CodeGeneratorSamples	InsertCode(tag, insertCode)
			11 //system execute will start this file, cmd will be paused (it	Integer.valueOf(str)
			12 //- please manually unpause cmd (any button/Enter)	IsHex(str)
			13 14 // is the short weekin and the week is not defined in the ser	IsInt(str)
			15	Math abs(a)
				Math.acos(a)

To sum up, by modifying JS files there is a possibility to interact with all the selected files and generate code from custom templates (by tags).

### 4.2. Files Control

Files Control basic buttons are used for importing files to the project, adding new, single files or removing the selected one (Figure 41). Files to import can be selected from Visual Studio projects or added separately by user.

By using *add* button, you can add .h, .cpp or .hpp files to the project.

*Remove* button allows to delete a single file but also to delete entire loaded folder or a project.

In the tree with imported files, you can find output path and a button "…" that allows you to manually specify an output path for JS methods (InsertCode, ReplaceCode). When these methods will make any changes to the files, those changes will be saved just in this output path. It is set by default to the path of the imported file.

Yellow triangle with an exclamation mark inside informs about warning – a file cannot be found.

Figure 41. Basic button in Files Control

2	Files Control	ð×
s Cont	Import files Add Remove	
Ē	Input Path Output Path	
	▼ ▲ ATS_CPPProjectTes	
E.	🔻 📊 Header Files	
- E	✓ h ATS_ClassDi C:\Users\Oliwia\Desktop\ATS_CPPPro	)
pts	✓ 🖻 ATS_ClassDi C:\Users\Oliwia\Desktop\ATS_CPPPro	)
Scri	🗹 🛕 ATS_CPPTes C:\Users\Oliwia\Desktop\ATS_CPPPro	)
_	✓ h ATS_CppTes C:\Users\Oliwia\Desktop\ATS_CPPPro	
	✓ h ATSTESTCL C:\Users\Oliwia\Desktop\ATS_CPPPro	
	✓ ▶ ClassWithR C:\Users\Oliwia\Desktop\ATS_CPPPro	
	✓ b globalFun.h C:\Users\Oliwia\Desktop\ATS_CPPPro	]
	Source Files	
	🖌 🔤 ATS_ClassDi C:\Users\Oliwia\Desktop\ATS_CPPPro	
	🖌 🔤 ATS_ClassDi C:\Users\Oliwia\Desktop\ATS_CPPPro	
	🖌 🔤 ATS_CppTes C:\Users\Oliwia\Desktop\ATS_CPPPro	
	🖌 🔤 ATSTESTCL C:\Users\Oliwia\Desktop\ATS_CPPPro	
	✓ 🔤 main.cpp C:\Users\Oliwia\Desktop\ATS_CPPPro	]
	Additional Files	

Clicking *Import files* button opens new window with selecting VCXProj file (Figure 42).

Figure 42. Selecting file to import in Files Control

	×
<i>←</i>	
Select VCXProj file	
Path to Visual Studio project:	
	Next Cancel

By going *"Next*", there is a window that allows you to choose files, that you would like to include in a project (Figure 43) – you can select single elements or whole folders. To include them, select the item and click the right arrow. It will move the content of the selected item to the right side *"included files*".

Figure 43. Selecting particular files to include in Files Control

Files to include:	Included files:
<ul> <li>ATS_CPPProjectTesting</li> <li>Header Files</li> <li>ATS_ClassDisabledConstruct</li> <li>ATS_ClassDisabledConstruct</li> <li>ATS_CPPTesting_Nested.h</li> <li>ATS_CPpTestingPrj.h</li> <li>ATSTESTCLANG.h</li> <li>ClassWithRefsToPrimitives.h</li> <li>globalFun.h</li> </ul>	<ul> <li>ATS_CPPProjectTesting</li> <li>Source Files</li> <li>ATS_ClassDisabledConstruct</li> <li>ATS_ClassDisabledConstruct</li> <li>ATS_CppTestingPrj.cpp</li> <li>ATSTESTCLANG.cpp</li> <li>main.cpp</li> </ul>

At the end of the importing process there should be displayed a window with information that the importing was successful. Such imported files will be displayed in a tree and selecting one of its items displays the code of the file in the field on the left (Figure 44).

Figure 44. Files Control view in Code Generator



# **List of Figures**

Figure 1. License Tools	3
Figure 2. Welcome Window	
Figure 3. Main View of ATS5	5
Figure 4. Compilation Tools	5
Figure 5. Importing files to CPP Tests	6
Figure 6. Selection section in CPP Tests	7
Figure 7. Main View of ATS5 with imported project	8
Figure 8. Recent projects list in Welcome Window	8
Figure 9. Remove missing project in Welcome Window	9
Figure 10. File – Save options	9
Figure 11. Configuration - Database	
Figure 12. Code Coverage tab	
Figure 13. CTC Tools	
Figure 14. Defining constructors with non-params	
Figure 15. Information while recognizing constructor with parameters	
Figure 16. Successfully built DLL notification	
Figure 17. Adding tests via context menu	13
Figure 18. Main View of ATS5 with added tests	14
Figure 19. Setting wrong data type for a test parameter	14
Figure 20. Adding user variable section	
Figure 21. Reference to global variable in test's parameter	
Figure 22. User variable as a pointer used in test's parameter	
Figure 23. Setting or getting pointer user variable	15
Figure 24. Test sequences	16
Figure 25. Ranges in tests' parameters	17
Figure 26. Ranges in return values	
Figure 27. Special characters in ranges	
Figure 28. Reports Tools	20
Figure 29. View tab	21
Figure 30. Charts section	21
Figure 31. Importing/exporting buttons in class definition	
Figure 32. Importing/exporting buttons in tests' parameters section	

Figure 33. Simulation – Open SimuDLL in VS	23
Figure 34. SimuDLL Configuration	23
Figure 35. Toolbar additional features	24
Figure 36. Force SimuExe close	24
Figure 37. Tools tab	25
Figure 38. Code Generator basic button	25
Figure 39. Buttons for Scripts Control	26
Figure 40. Scripts Control view in Code Generator	27
Figure 41. Basic button in Files Control	28
Figure 42. Selecting file to import in Files Control	28
Figure 43. Selecting particular files to include in Files Control	29
Figure 44. Files Control view in Code Generator	29