UM1886
User manual

uTester

Introduction

uTester is a user-friendly tool developed creating a focus on ECU level diagnostic tools as the analysis of traceability information and first level diagnostic in order to have an early way to improve analysis cycle time.

uTester allows to execute the tests on the field to analyze issue on system modules or to validate features on a device.

This document gives an overview of uTester and illustrates the first steps to use basic functionalities. In particular the first chapter introduces the tool, describes requirements and provides notes to download and to install it. The second one explains how to use it and some basic and special widgets to build simple applications demo.

Contents

List of figures

1 Requirements

uTester (PCs) has been designed to work correctly using Windows 7 platform and no particular hardware requirements are requested. The tool needs to communicate with the target using JTAG port. Two interfaces are available to perform this connection: Raisonance RLink or SPC5-UDESTK.

2 uTester overview

uTester is often used to analyze an issue on a customer module, to detect a possible issue on a device, to perform a functional verification of a device or to develop a new system.

uTester allows to easily perform these tasks in an automated way with results logging.

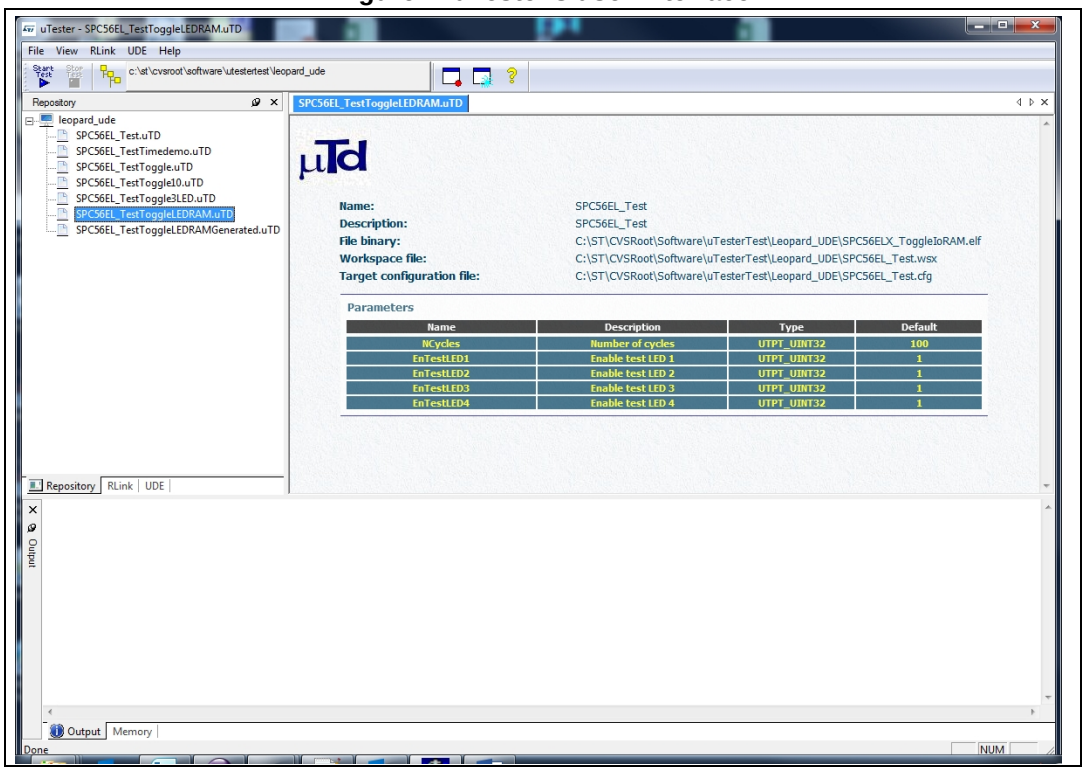
Execution of the tests sequence is controlled by the host machine connected to the JTAG interface of the target.

2.1 uTester's user interface

uTester's user interface integrates menu, toolbars and windows that allow to load and to execute the tests easily.

- Menu (*Section 2.2*)
- Command toolbar
- Repository window
- Project window
- Output window
- UDE window
- RLink window
- Memory window

Figure 1. uTester's user interface



2.2 Menu

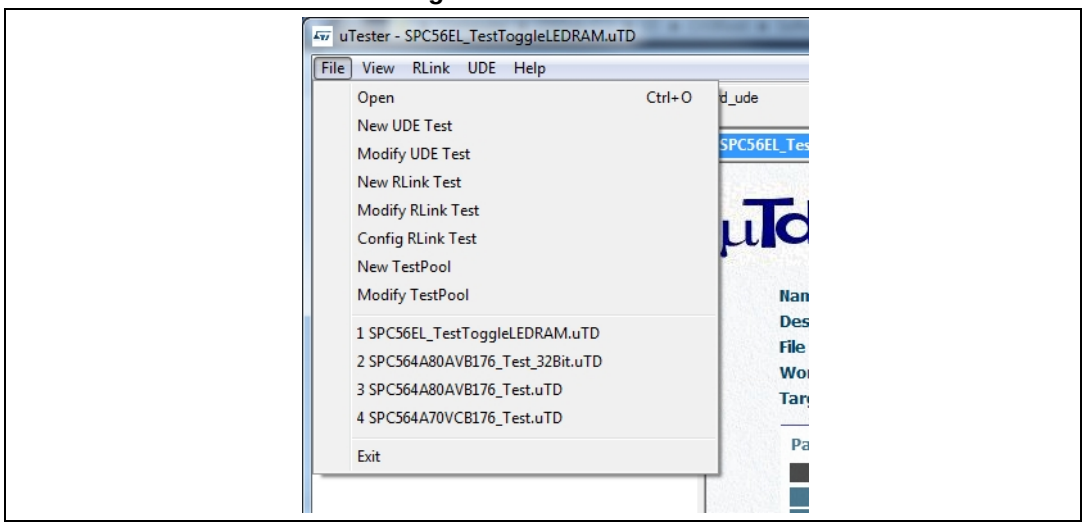
Menu has the following items:

- **File** (*Section 2.2.1*)
- **View** (*Section 2.2.2*)
- **RLink** (*Section 2.2.3*)
- **UDE** (*Section 2.2.4*)
- **Help** (*Section 2.2.5*)

2.2.1 File

File item allows to create, open and modify an uTD test and test pool as it is showed in *Figure 2*.

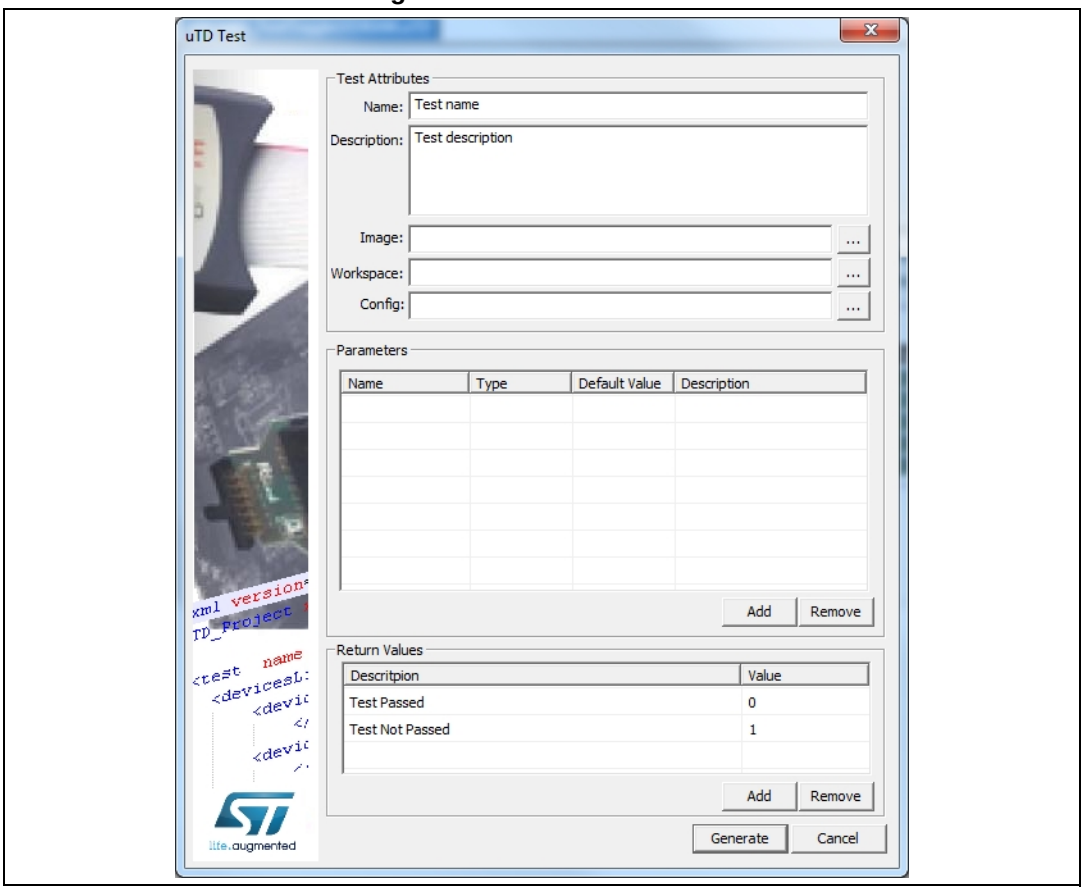
Figure 2. Menu- "File"



File item includes the following items:

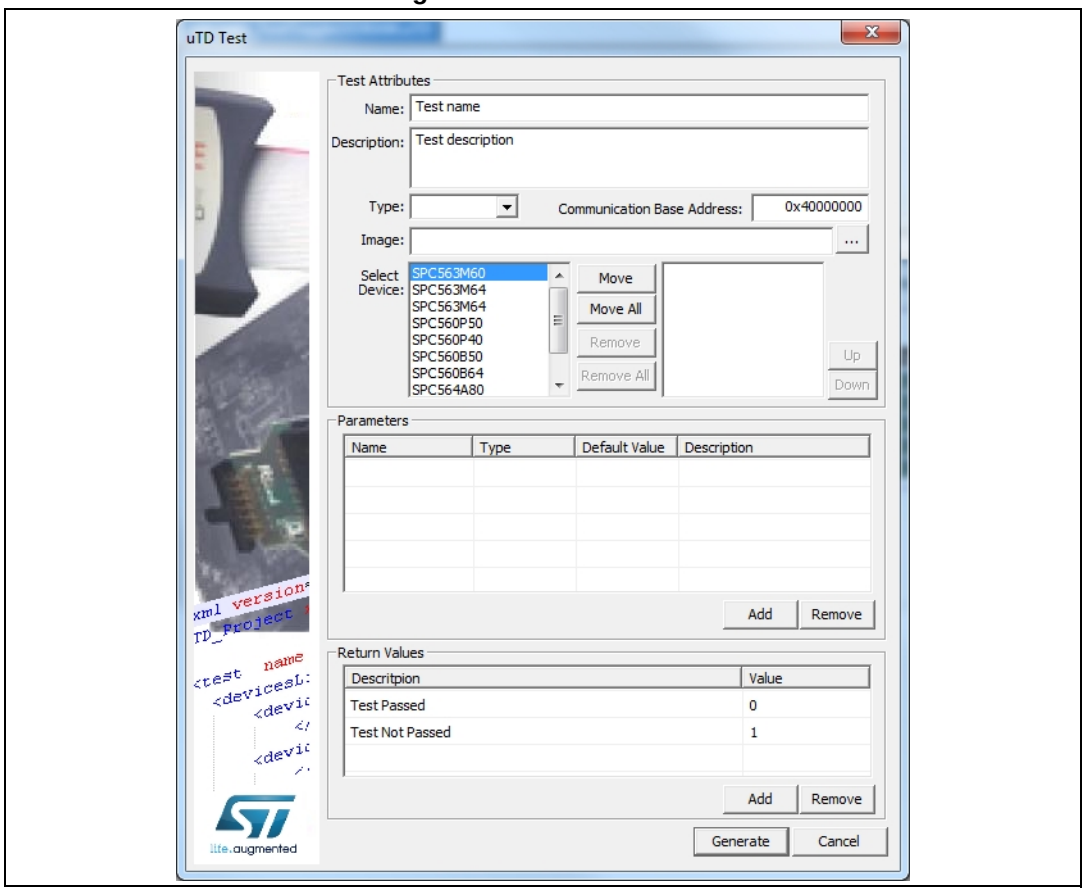
- **Open** item that allows to open an existent test or an existent test pool.
- New **UDE Test** item that allows to create a new uTD file working with UDE interface. After the selection of **New UDE Test** item, a new dialog with all fields blank is showed (see *Figure 1* and *Figure 3*). This dialog allows to generate an uTD file that contains the XML code just to describe the new test. The user can insert the name, the description and the image file of the test (hex format), the workspace file name and the configuration file name. In addition the user can insert the parameters list of the test and the return values list. When the user has terminated to insert the data for the test, he has to push **Generate button**. In this case, a browse window allows the user to insert the test file name and the file is created. Pushing **Cancel button**, the dialog closes and the operation is aborted.

Figure 3. UTD UDE new test



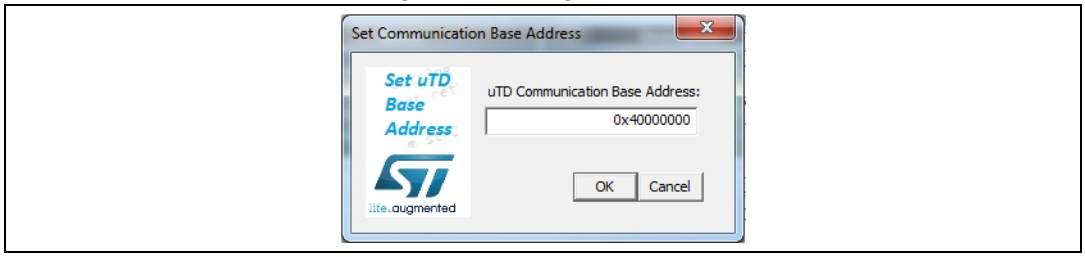
- **Modify UDE Test** item allows to change an existent uTD UDE file. After the selection of **Modify UDE Test** item, a new dialog with all fields containing the data of the test to update is showed (*Figure 1* and *Figure 3*). The user can modify all parameters. When the user has terminated to insert the data for the test, he has to push **Generate button** and the updating is performed.
- **New RLink Test item** allows to create a new uTD file. After the selection of **New Test item**, a new dialog is showed (*Figure 4*). This dialog allows to generate an uTD file that contains the XML code just to describe the new test. The user can insert the name of the test, the description of the test, the type of the test (RAM or Flash) the image file of the test (hex format). The user can also select the list of the devices valid for the test, the parameters list of the test and the return values list. When the user has terminated to insert the data for the test, he has to push **Generate button**. In this case, a browse window allows the user to insert the test file name and the file is created. Pushing **Cancel button**, the dialog closes and the operation is aborted.

Figure 4. New RLink test



- **Modify RLink Test** item allows to modify an existent UTD file. After the selection of **Modify RLink Test** item, a new dialog is showed (*Figure 4*). This dialog shows all information related to the test opened. The user can modify all parameters. When the user has terminated to insert the data for the test, he has to push **Generate button** and the update is performed.
- **Config RLink Test** item allows to update the communication base address for the **RLink** test. A new dialog is showed (*Figure 5*).

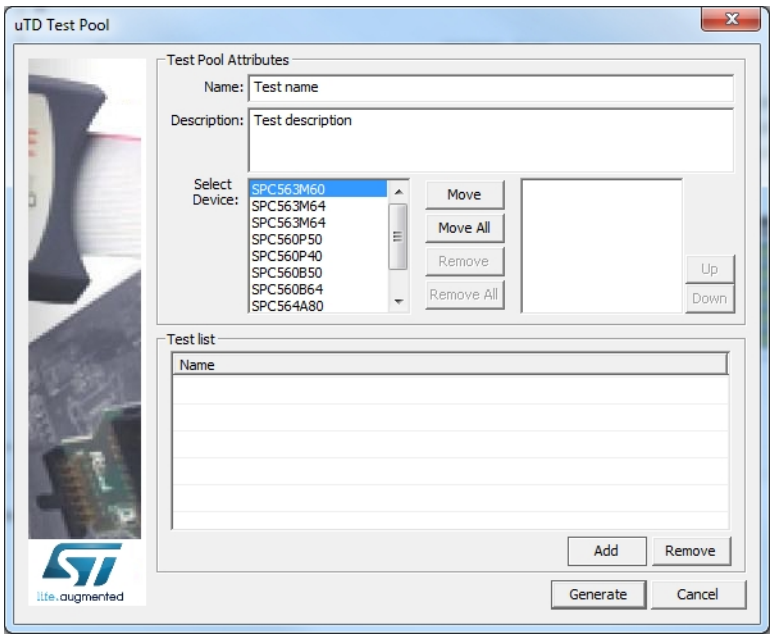
Figure 5. Config RLink test



- **New TestPool** item allows to create a new UTD test pool file. After the selection of **New TestPool** item, a new dialog is showed (*Figure 6*). This dialog allows to generate an uTD file that contains the XML code just to describe the new test pool. The user can insert the name of the test pool and the description of the test pool. Also, the user can select the list of the devices valid for the test pool, and the test list of the test pool. When the user has terminated to insert the data for the test, he has to push **Generate**

button. In this case, a browse window allows the user to insert the test pool file name and the file is created. Pushing **Cancel button**, the dialog closes and the operation is aborted.

Figure 6. New uTD test pool

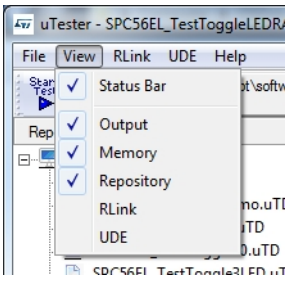


- **Modify TestPool** item allows to modify an existent uTD test pool file. After the selection of **New TestPool** item, a new dialog is showed (*Figure 6*). This dialog shows all information related to the UTD test pool opened. The user can modify all parameters. When the user has terminated to insert the data for the test pool, he has to push **Generate button** and the update is performed.
- List of the last test opened.
- **Exit** item to close uTester.

2.2.2 View

View item allows to select the uTester parts that show by the GUI (*Figure 7*).

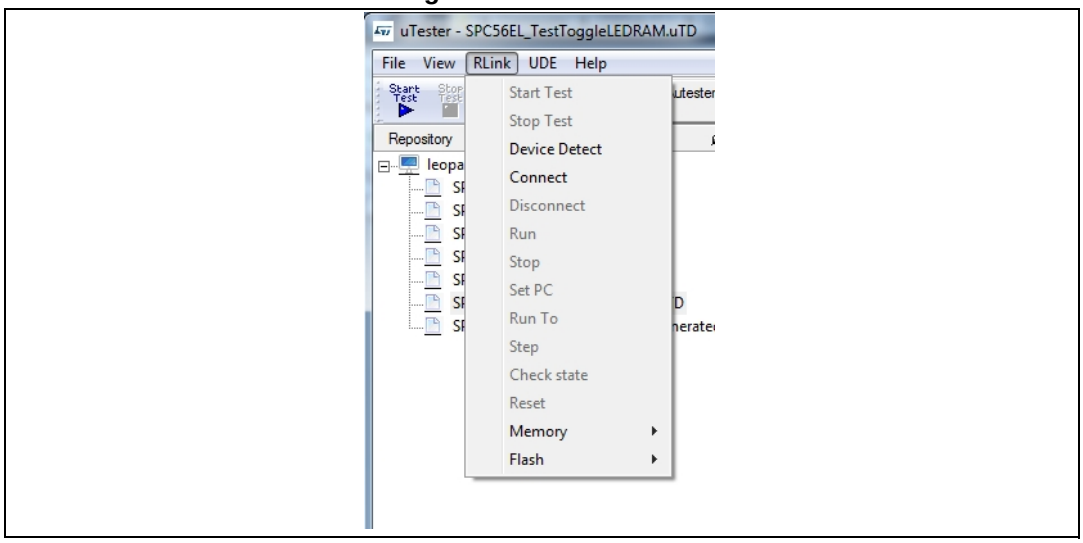
Figure 7. Menu - View



2.2.3 RLink

RLink item allows to execute all command available by **RLink** interface (*Figure 8*).

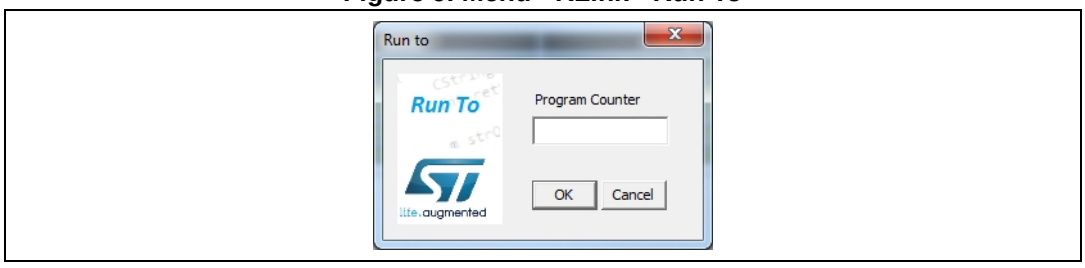
Figure 8. Menu - RLink



RLink has the following items:

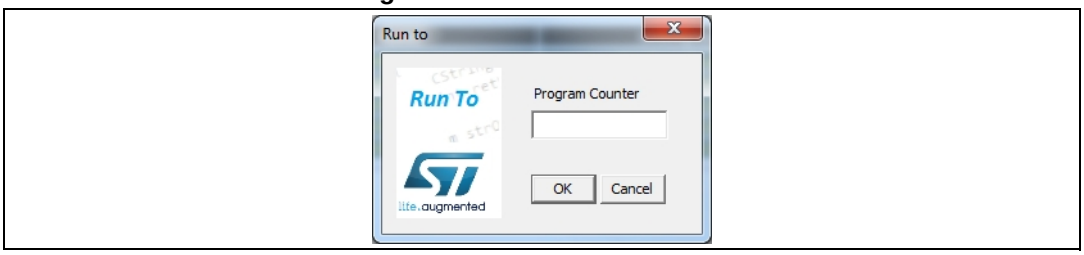
- **"Start Test"** to start the execution of an UTD test using **RLink** interface.
- **"Stop test"** to stop the execution of UTD test using **RLink** interface.
- **"Device Detect"** to detect the device connected by **RLink** interface.
- **"Connect"** to start the connection with the device connected by **RLink** interface.
- **"Disconnect"** to stop the connection with the device connected by **RLink** interface.
- **"Run"** to start the execution of the code.
- **"Stop"** to stop the execution of the code
- **"Run To"** to execute the code until the address set.

Figure 9. Menu - RLink - Run To



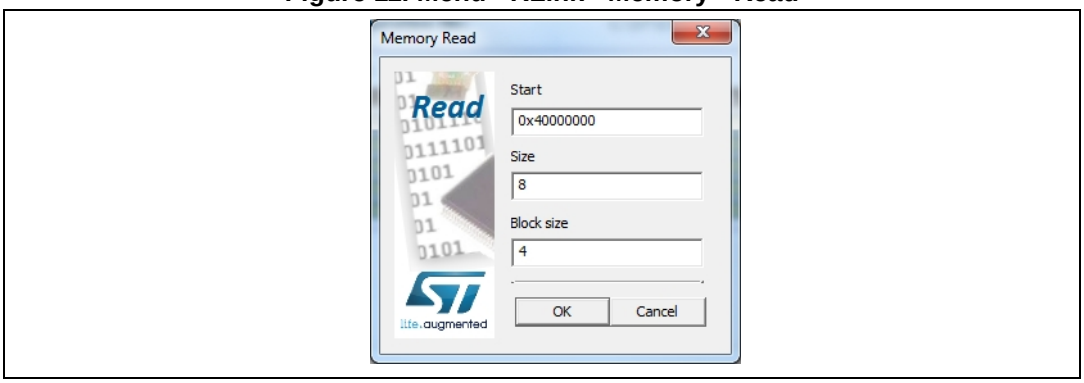
- **"Set PC"** to set the program counter to the specified position. Selecting this item, uTester shows a dialog showed in *Figure 10*.

Figure 10. Menu - RLink - Into



- **“Step”** to execute the code step by step.
- **“Check state”** allows to report the state of the device and the current value of the program counter.
- **“Reset”** allows to reset the target device connected.
- **“Memory”** allows to access to the following menu:
  - **“Read”** allows to read the memory from the connected target device. The user can choose the start address, the size of block of the memory and size of the block of the memory that are showed on memory window. Pressing this button is shown the memory read dialog.

Figure 11. Menu - RLink - Memory - Read



  - **“Write”** allows to write the memory of the target device connected and it shows the memory write dialog. The user can insert the address and the value.

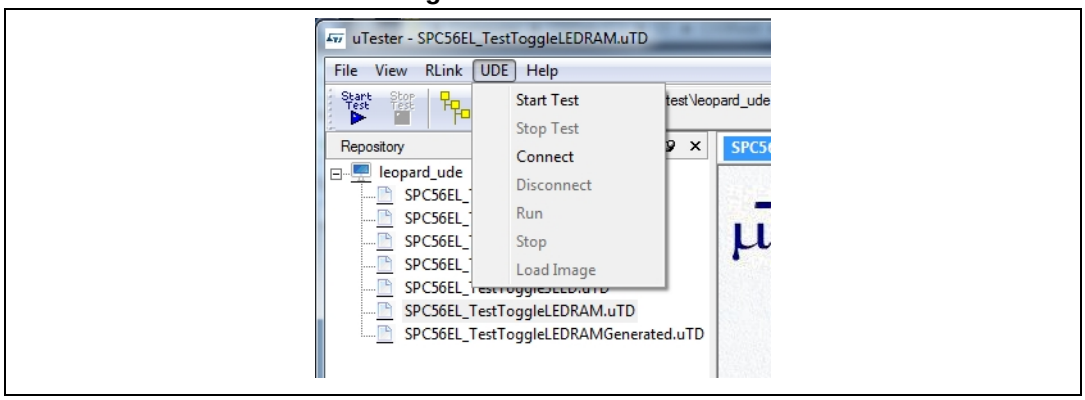
Figure 12. Menu - RLink - Memory - Write

- **"Flash"** allows to access to the following menu:
  - **"Dump"** allows to dump of all flash of the target device connected that is showed on the memory windows
  - **"Load"** allows to load an image file (hex format) on the target device connected. Pushing this button is opened a file dialog that allows to choose the image file.
  - **"Erase"** allows to erase the flash of the target device connected.

2.2.4 UDE

UDE item allows to execute all command available by **UDE** interface (*Figure 13*).

Figure 13. Menu - UDE



UDE has the following items:

- **"Start Test"** to start the execution of an UTD test using UDE interface.
- **"Stop test"** to stop the execution of UTD test using UDE interface.
- **"Connect"** to start the connection with the device connected by UDE interface.
- **"Disconnect"** to stop the connection with the device connected by UDE interface.
- **"Run"** to start the execution of the code
- **"Stop"** to stop the execution of the code
- **"Load Image"** to load the binary image indicated into the field **"Image file name"** of UDE window. If the image is for RAM, it loads directly into the RAM. If the image is for Flash, it opens a new dialog to manage the Flash. Pushing **Program** button the image loads into the Flash.

2.2.5 Help

Help item shows a dialog containing the version of uTester (*Figure 14*).

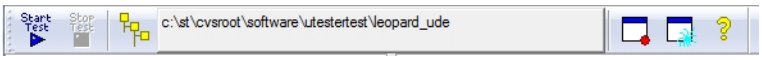
Figure 14. Help

2.3 Command toolbar

Command toolbar showed in *Figure 15* contains the following buttons:

- **"Start test"** allows to execute the active test.
- **"Stop test"** allows to stop the test during the execution.
- **"Test Repository"** allows to select the folder containing UTD test. "Repository" window shows all UTD test and test pool available in the folder selected.
- **"Save output window"** allows to save the content of "output" window in a log file.
- **"Clear output window"** allows to clear the "output" window.
- **"About"** allows to show the dialog containing the version of uTester (*Figure 14*).

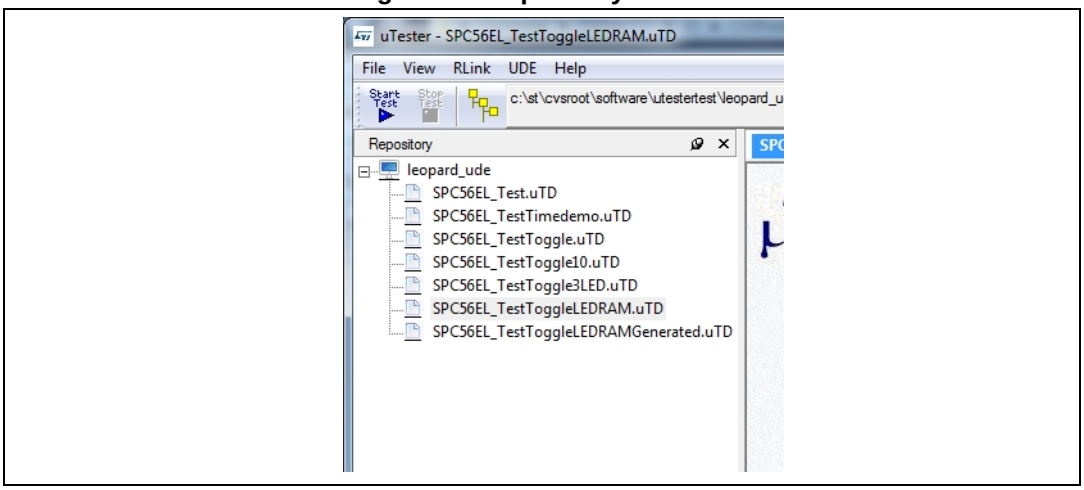
Figure 15. Command toolbar



2.4 Repository window

Repository window shows the tests and test pool available on the folder selected by repository folder (*Figure 16*).

Figure 16. Repository window



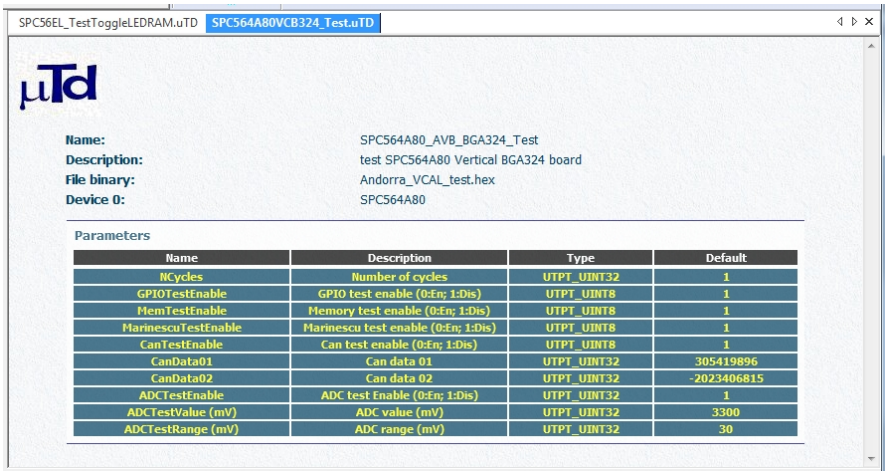
2.5 Project window

Project window shows the test and the test pool projects (*Figure 27*). Each test project or test pool project is described by xml format. This window shows all information formatted.

The test project window shows all information related to each test. If the test is designed for **RLink** interface, the information displayed are (*Figure 17*):

- Test name
- Test description
- Image file name
- Parameter list

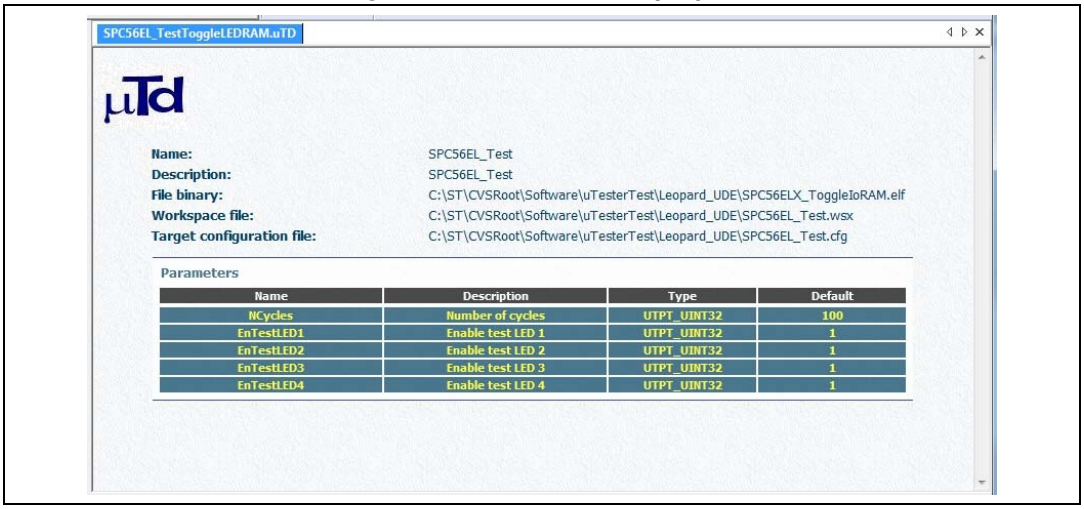
Figure 17. UTD RLink test project



If the test is designed to work with UDE interface, the information displayed are (*Figure 18*):

- Test name
- Test description
- Image file name
- Workspace file name
- Target configuration file

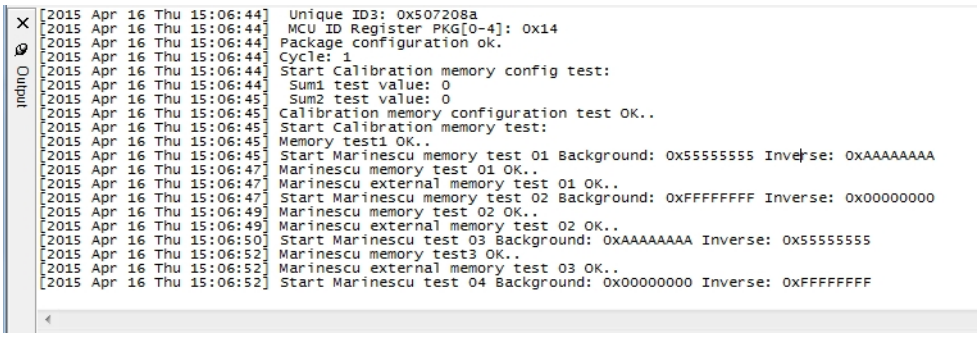
Figure 18. UTD UDE test project



2.6 Output window

Output window shows the output of the uTester (*Figure 19*).

Figure 19. Output window



2.7 UDE window

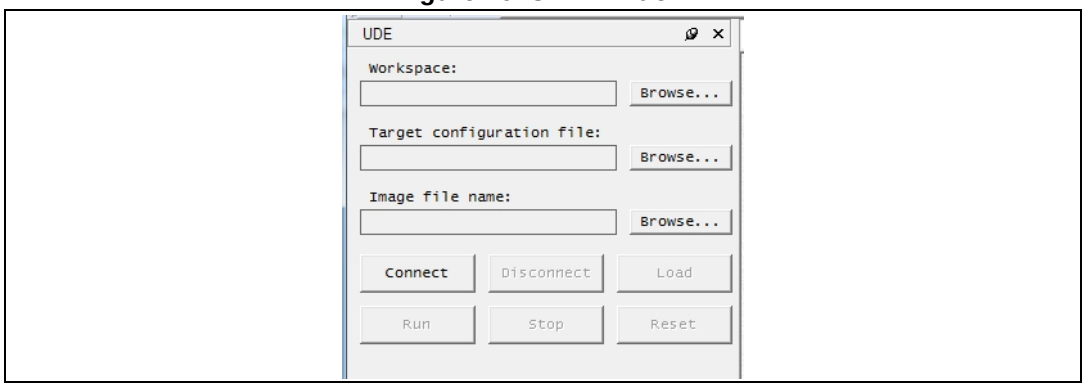
UDE window allows to set all parameters to work using SPC5-UDESTK interface. It is possible to select the workspace file name, the target configuration file name and the image file name.

The windows on the bottom side contains also some buttons to perform specific commands

(*Figure 20*):

- **“Connect”**: to connect the target
- **“Disconnect”**: to disconnect the target
- **“Load”**: to load the image selected in the image file name edit box
- **“Run”**: to execute the code
- **“Stop”**: to stop the execution of the code
- **“Reset”**: to reset the target

Figure 20. UDE window

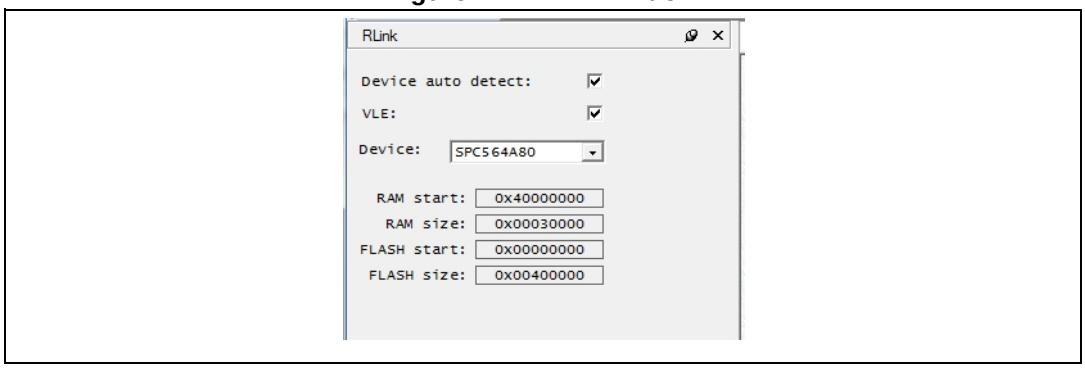


2.8 RLink window

RLink window allows to set some parameter to configure **RLink** interface (*Figure 21*):

- “**Device auto detect”**: to set device auto detection during the connection phase.
- **“VLE”**: to set if VLE instruction set is used.
- **“Device”**: if **“Device auto detect”** is set, after connection phase, this field shows the target connected. If **“Device auto detect”** is not set, the user has to select the correct device.
- **“RAM start”**: RAM start address of the device selected.
- **“RAM size”**: RAM size of the device selected.
- **“Flash start”**: Flash start address of the device selected.
- **“Flash size”**: Flash size of the device selected.

Figure 21. RLink window

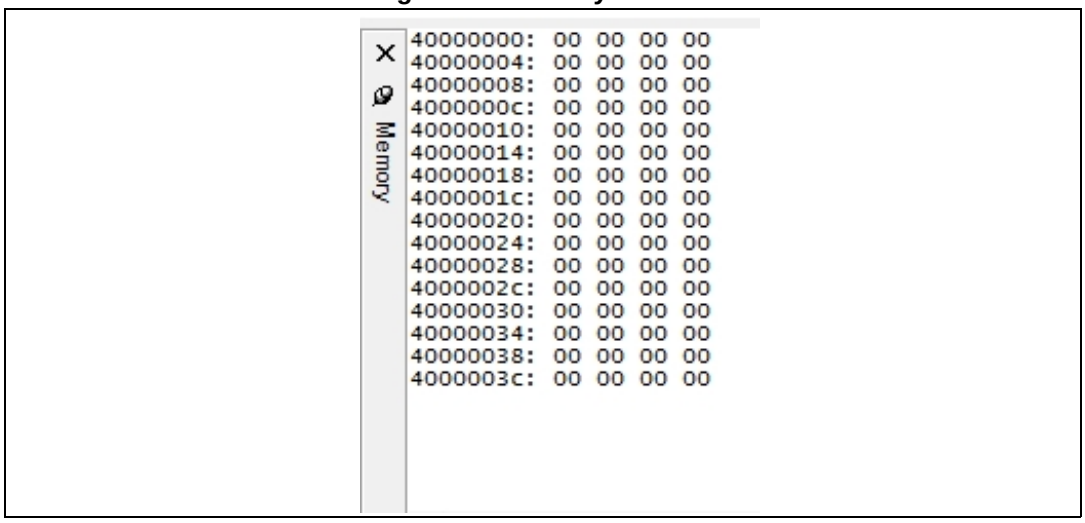


2.9 Memory window

Memory window shows the memory of the device selected by memory read dialog (*Figure 22*).

Each row contains from left side to right side the address and the memory content. The user can choose by memory read dialog the start address, the size of the memory to read and the block size that represent the length of each row.

Figure 22. Memory window



3 Diagram overview

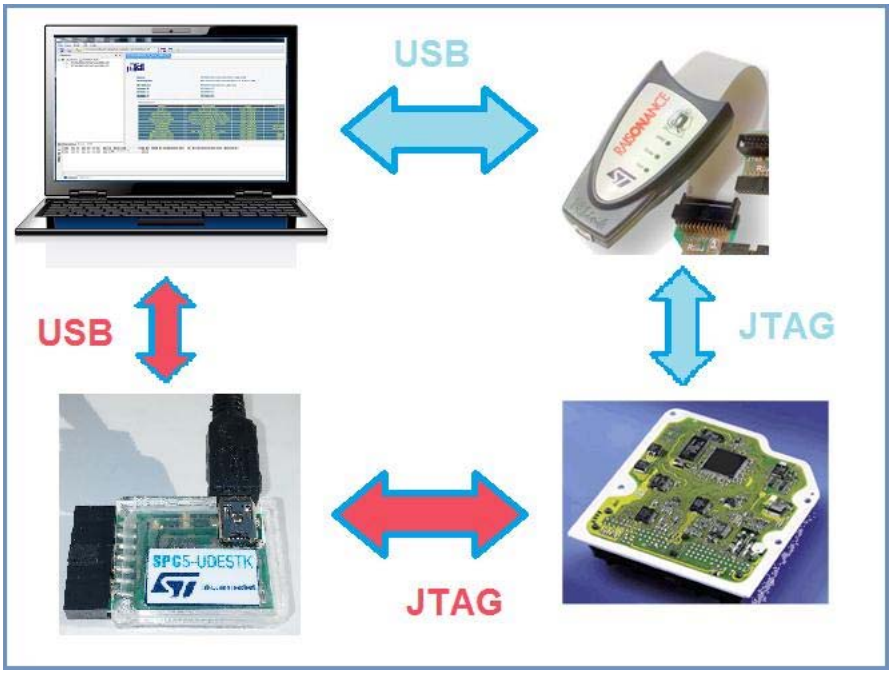
This chapter gives a description of the system used to work with uTester tool. The execution of the tests sequence is controlled by the host machine connected to the debug interface JTAG of the target. In this way the user is able to interact with the test and to control the test execution.

uTester allows to use two different solutions depending which JTAG interface the user want to use:

- RLink solution in blue:
  - To use this configuration the user needs to have Raisonance Rlink dongle installed and licensed on his PC.
  - The test firmware and the uTD file must be created to use Rlink solution.
- UDE solution in red:
  - To use this configuration the user needs to have SPC5-UDESTK dongle installed and licensed on his PC.
  - The test firmware and the uTD file must be created to use UDE solution.

The top level diagram of the uTester is showed on *Figure 23*:

Figure 23. Top level diagram



4 Demo project

4.1 How to use

To use uTester the user has to follow the steps listed below:

- Generate the test firmware target side using uTester library. This step is different if the interface used is **RLink** or **UDE**. The difference is descripted in *Section 4.2*.
- Start uTester tool.
- Generate the uTD file. Also uTD file is different depending the interface used. The difference is descripted in the next sections.
- Load the uTD file.
- Run the test. The test execution flow is the same for both interfaces.

4.2 RLink solution

4.2.1 Test design target side

To write the test code target side, the user has to use specifics macros and API. These macros and API allow to define the test environment and they allow to control the test execution.

To define the parameters, the user must use the following macros:

- PARAMETER_DECLARATION: this macro must be used inside a #pragma section in order to link the test environment variable in a fixed memory address.
- BEGIN_PARAMETERS_MAP: this macro must be used to declare the parameters array.
- ADD_PARAMETER(<parameter size>,<parameter address>): this macro must be used to add a parameter to the parameters array
- END_PARAMETERS_MAP: this macro must be used to close the parameters array declaration.

To control the test execution, the user must uses the following API:

  - *BeginTest(NUM_PARAMETER)*: this API initializes the test environment before test code starts.
  - *EndTest(u32TestResult)*: this API allows to release the test environment and to send the result of the test to uTester tool.
  - *WriteToGui(cTestMessage, strlen(cTestMessage))*: this API allows to send a message to the GUI.

4.2.2 Demo project source code example

Parameters declaration section:

```
...
#pragma ghs section data=".CommUTester"
PARAMETER_DECLARATION
#pragma ghs section data=default
...
```

```
Parameters map definition:
...
uint32_t blinkingLedTime[LED_NUM]={10000,10000,10000};
uint8_t blinkingLedEnable[LED_NUM]={1,1,1};
BEGIN_PARAMETERS_MAP
ADD_PARAMETER(1,&blinkingLedEnable[0])
ADD_PARAMETER(4,&blinkingLedTime[0])
ADD_PARAMETER(1,&blinkingLedEnable[1])
ADD_PARAMETER(4,&blinkingLedTime[1])
ADD_PARAMETER(1,&blinkingLedEnable[2])
ADD_PARAMETER(4,&blinkingLedTime[2])
END_PARAMETERS_MAP
...
LED blinking test main function code:
//
*************************************************************************
**
// ***************************** Functions
***********************************
//
*************************************************************************
**
int32_t main(int32_t argc, const char *argv[])
{
int i=0;
int count=0;
char cTestMessage[WRITEBUFFER_SIZE];
char* str;
uint32_t u32TestResult = 0;
memset(cTestMessage,0x00 ,WRITEBUFFER_SIZE );
sys_init();
peripherals_init();
for (i=0; i<GPIO_NUM; i++)
{
SIU.PCR[GPIO[i]].R = 0x20E; //init
SIU.GPDO[GPIO[i]].R = 1; //turn off all led
}
SIU.GPDO[GPIO[0]].R = 0;//turn on LED 1
BeginTest(NUM_PARAMETER);
strcpy(cTestMessage,"START TEST");
WriteToGui(cTestMessage, strlen(cTestMessage));
for (i=0; i<LED_NUM; i++)
{
if (blinkingLedEnable[i])
{
SIU.GPDO[GPIO[i]].R = 0;
```

```
STM_Delay(blinkingLedTime[i]);
SIU.GPDO[GPIO[i]].R = 1;
STM_Delay(blinkingLedTime[i]);
memset(cTestMessage,0x00 ,WRITEBUFFER_SIZE );
itoa(i,str,10);
strcpy(cTestMessage,"Blinking...");
strncat(cTestMessage,str);
WriteToGui(cTestMessage, strlen(cTestMessage));
}
else
{
SIU.GPDO[GPIO[i]].R = 1;
}
}
memset(cTestMessage,0x00 ,WRITEBUFFER_SIZE );
strcpy(cTestMessage,"END TEST");
WriteToGui(cTestMessage, strlen(cTestMessage));
u32TestResult = 1234;
EndTest(u32TestResult);
while (1); //wait forever
return 0; //this point should be never reached
}
```

4.2.3 Test executions

In order to execute a test, the user has to open the uTD file describing the test. The uTD file can be opened using the menu (File->Open), the toolbar (**Open** button) or the repository window (double click on the uTD file).

The project window (*Figure 17*) shows the uTD file opened.

The user has to press start **Test** button from the command toolbar (*Figure 16*) to start the test.

4.2.4 Test pool executions

In order to execute a test pool, the user has to open the uTD file describing the test pool.

The uTD file can be opened using the menu (File->Open), the toolbar (**Open** button) or the repository window (double click on the uTD file). The project window shows the uTD file opened.

The user has to press **Start** test button from the command toolbar (*Figure 15*) to start the test pool.

4.3 UDE solution

4.3.1 Test design target side

To write a test target side the user has to include two files:

- uTD_UDETest.c
- uTD_UDETest.h

These files contains the following functions:

- *void uTD_InitTest()*: this function is used to initialize uTester flags.
- *void uTD_BeginTest()*: this function is used to synchronize the starting of the test. uTester sets a flag to begin the execution of the test.
- *void uTD_EndTest(uint32_t u32TestResult)*: this function allows to communicate to uTester the end of the test and the result of the test.
- *void uTD_WriteToGui(uint8_t u8Buff[UTD_WRITE_BUFFER_SIZE])*: this function is used to print a message on Output window of uTester.

The user can define some parameters that are controlled by uTester. These parameters can be used for example to enable a specific test or to customize the test defining some value.

The test firmware contains the following section:

- Test parameters definition.
- Core and peripherals initialization functions.
- The call to *uTD_InitTest();* uTester flags is initialized.
- The call to *uTD_BeginTest();* uTester function starts the test.
- Test body containing the use of test parameters
- The call to uTD_EndTest(TestResult);

The following code is an example how to write the firmware target side:

```
…
//Parameters definition
unsigned int EnTestLED1 = 1;
unsigned int EnTestLED2 = 1;
unsigned int EnTestLED3 = 1;
unsigned int EnTestLED4 = 1;
unsigned int NCycles = 100;
unsigned int TestResult = 0;
…
//main function
void main(void)
{
    ….
    //Core and peripherals initialization
    INIT_Environment();
….
    //uTester init function
    uTD_InitTest();
….
```

```
    uTD_WriteToGui("Start uTester test");
….
    //uTester Begin function
  uTD_BeginTest();  //the test execution remains blocked here until uTester
set the start flag.
….
    //Test body: all test enable flag will be controlled by GUI.
   //Will toggle only the LED with the correspondent flag enabled.
    while (i<NCycles)
    {
   if (EnTestLED1)
       IoToggle(LED1);
   if (EnTestLED2)
       IoToggle(LED2);
   if (EnTestLED3)
       IoToggle(LED3);
   if (EnTestLED4)
       IoToggle(LED4);

   //Prepare a string to print the cycle number
   strcpy(TmpBuffer,"Cycle:");
   uTD_itoa(i,TmpBuffer2,10);
   strcat(TmpBuffer, TmpBuffer2);

   //Print the string
   uTD_WriteToGui(TmpBuffer);

   Delay(DELAY);
i++;
    }//while

    uTD_WriteToGui("End uTester test");

    TestResult = 1;

    //uTester end function
    uTD_EndTest(TestResult);
}
```

4.3.2 uTD UDE file generation

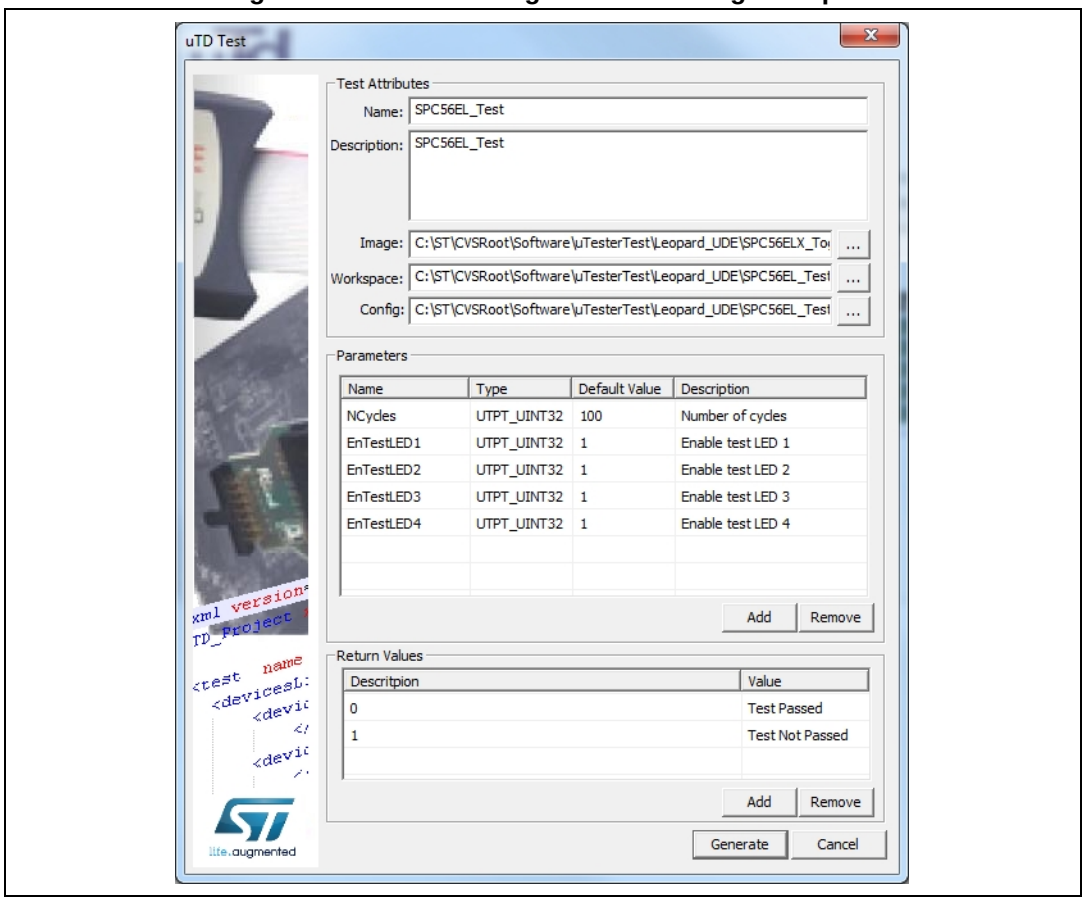
The second step to project an uTester test is to generate the uTD UDE file. This is an xml file and it is done automatically using uTester. Selecting the menu item **File->New UDE Test**, opens a specific dialog (UTD UDE new test).

The user has to fill all fields into the dialog. An important note is that the parameters defined into the firmware target side must be reported using the same name into “Parameters” section.

Pushing **Generate button**, the uTD file is generated.

Figure 24 shows an example of uTD UDE test dialog filled with the same information of the example showed in **“Test target side”** example.

Figure 24. uTD Test file generation dialog example



The xml file generated by uTester is the following:

```
<?xml version="1.0" encoding="UTF-16"?>
<uTD_UDE_Test_Project xmlns:dt="urn:schemas-microsoft-com:datatypes">
<test name= "TestProva01" description="Test description01"
binary="SPC56ELX_ToggleIoRAM.elf"
   WorkspaceFile="SPC56EL_Test.wsx"
TargetConfigurationFile="SPC56EL_Test.cfg">
<parametersList>
<parameter name= "NCycles" type="UTPT_UINT32" defaultValue="100"
description="description…">
</parameter>
<parameter name= "EnTestLED1" type="UTPT_UINT32" defaultValue="1"
description="description">
</parameter>
```

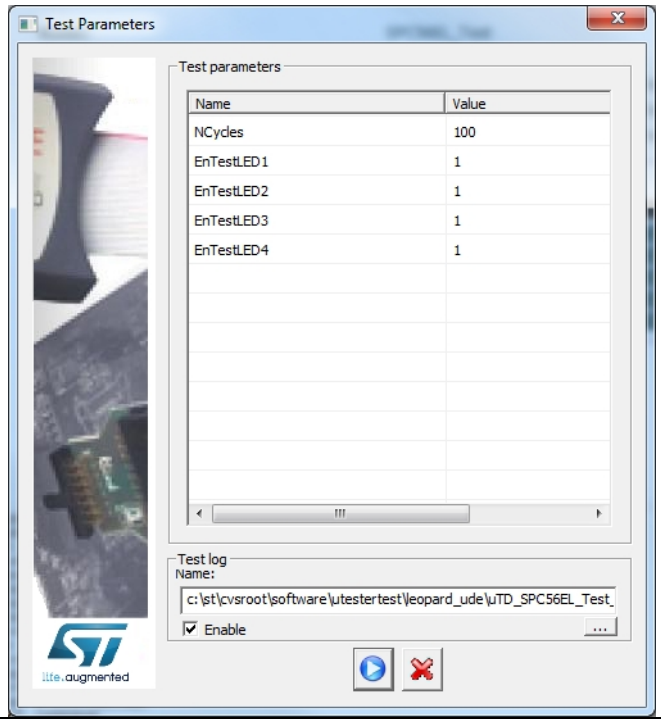
```
.....
</parametersList>
<returnValuesList>
<returnValue description= "Test Passed" value= "0"></returnValue>
<returnValue description= "Test Not Passed" value= "1"></returnValue>
</returnValuesList>
</test>
</uTD_UDE_Test_Project>
```

4.4 uTester test execution

The steps to execute a test are the following:

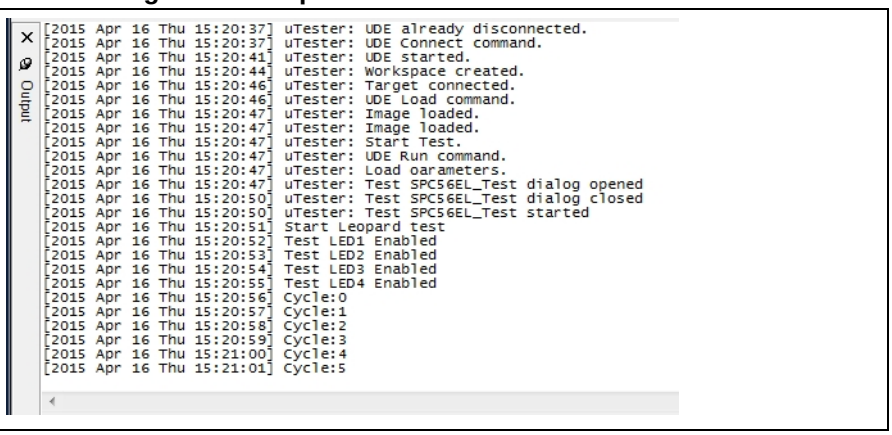
- Select the test folder containing test files from toolbar
- «Repository» window loads all test available
- Select and open the test
- Push «Start» button from toolbar
- «Test Parameters» dialog (*Figure 25*) allows to modify the test parameters and to enable/disable saving of test log. Push Start test button to start the test

Figure 25. Test parameter dialog



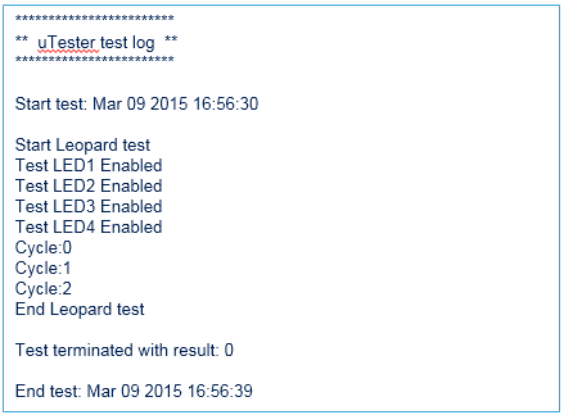
- Output window shows the result of the test (*Figure 26*).

Figure 26. Output window test result



- If the Test log check of test parameter dialog was enabled, a file containing the log of the test is produced (*Figure 27*).

Figure 27. uTester Log file



5 Revision history

Table 1. Document revision history

Date	Revision	Changes
07-May-2015	1	Initial release.