

# Altera NIOS II Compiler and Debugger

## Abstract

This application note describes the usage of the Altera compiler and debugger for the NIOS II microcontroller. The connection to the TRACE32 debugger also requires the preparation of a board support package like described within this document.

## Table of Contents

1	Introduction .....	2
2	Preparing the Target Board Configuration.....	3
3	Preparing necessary Settings within TEE .....	6
4	Running the Test.....	7
4.1	Executing Tests Automatically .....	7
4.2	Debugging the Test Application .....	8
5	Troubleshooting .....	11
5.1	Connection to the Target Board fails.....	11
5.2	Problems when Running the Test .....	11

## 1 Introduction

The Altera NIOS II debugger uses a normal GDB debugger as backend. This backend will be controlled by Tessy when executing tests. The test execution runs fully automated in this case.

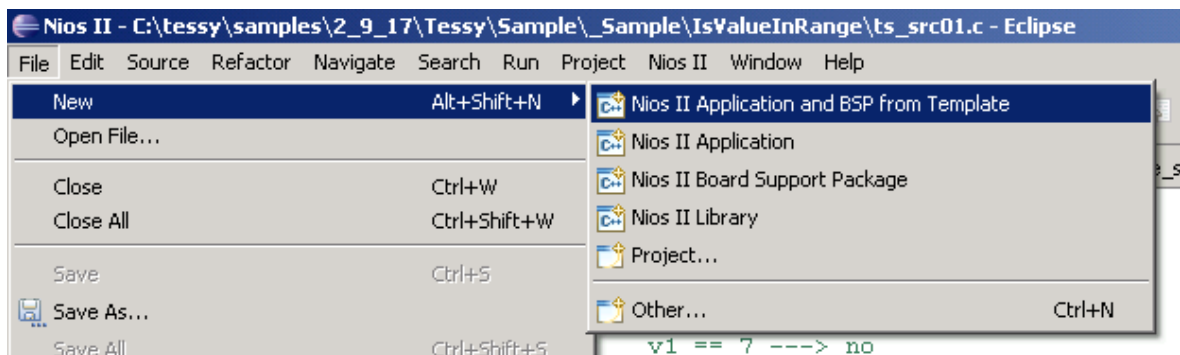
If you want to debug the test application with the test case values provided within TDE, you need to rebuild the test application in a special mode (with the input values compiled into the application). You may then download the test application using the normal GUI of the Altera NIOS II debugger and step through the test cases.

Before being able to execute tests, you need to prepare a BSP (board support package) for your desired target board and load this BSP into the target board. Please refer to Altera support for any questions on creating/preparing a BSP for your target board.

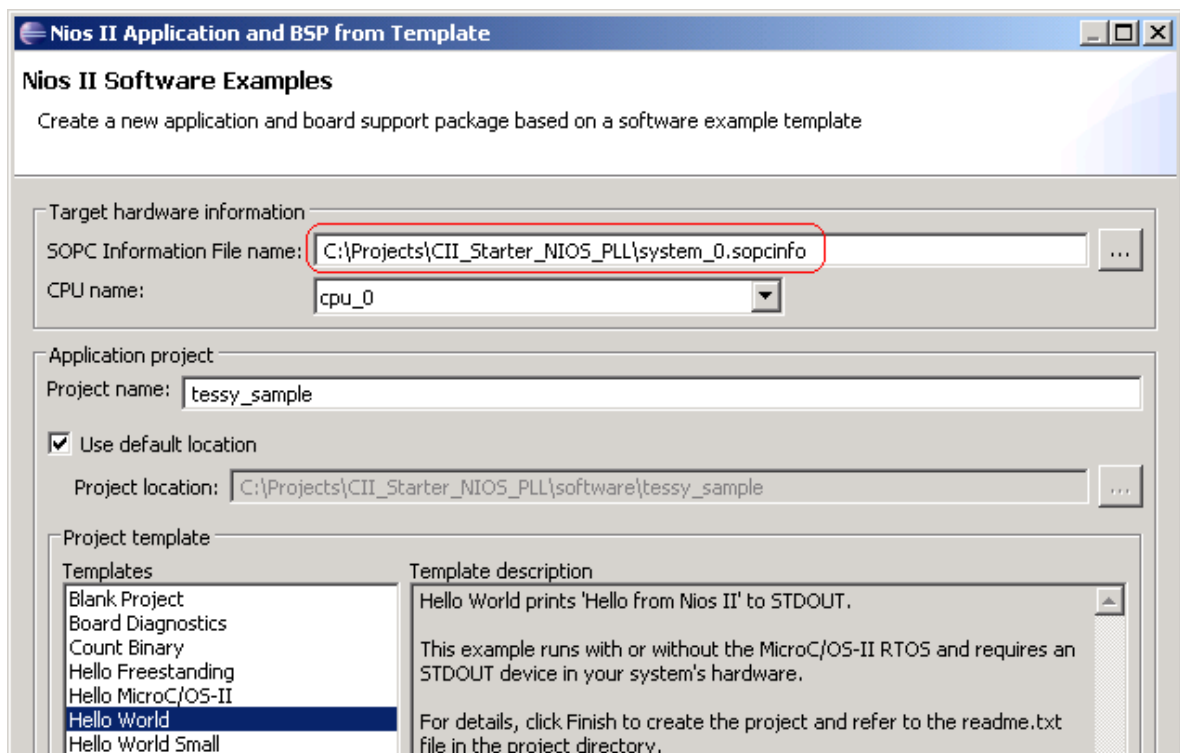
## 2 Preparing the Target Board Configuration

The first step in setting up the Altera NIOS debugger configuration is the selection of a suitable BSP and configuration file (“sopcinfo” file) for the target board. This may be created using the Altera tools or you may get a readily prepared configuration for your target board. Please refer to the Altera support for any information about the “sopcinfo” file.

If you have a suitable BSP and configuration package available, unzip this package into any location on your local disk. Then start the Altera NIOS Eclipse IDE and create a new application and board support package using the following menu entries:

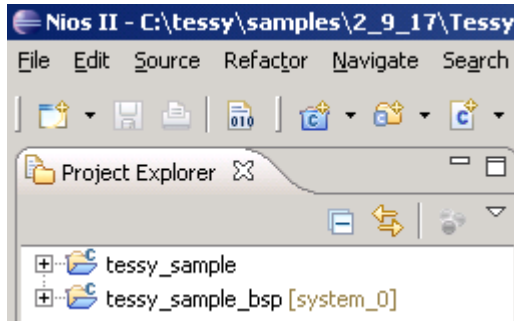


Add the following settings (i.e. select the suitable “sopcinfo” file):

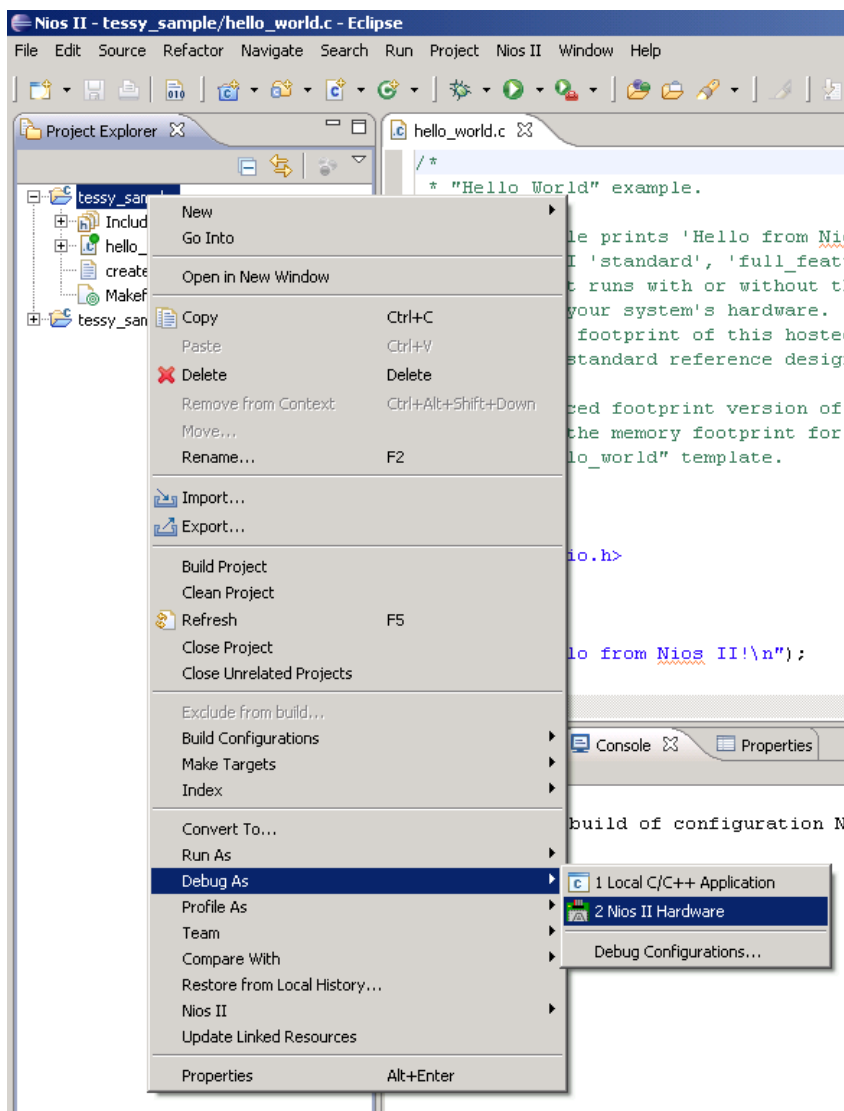


## Tessy Application Notes

This will create a new code project and a new BSP project with analogous names.

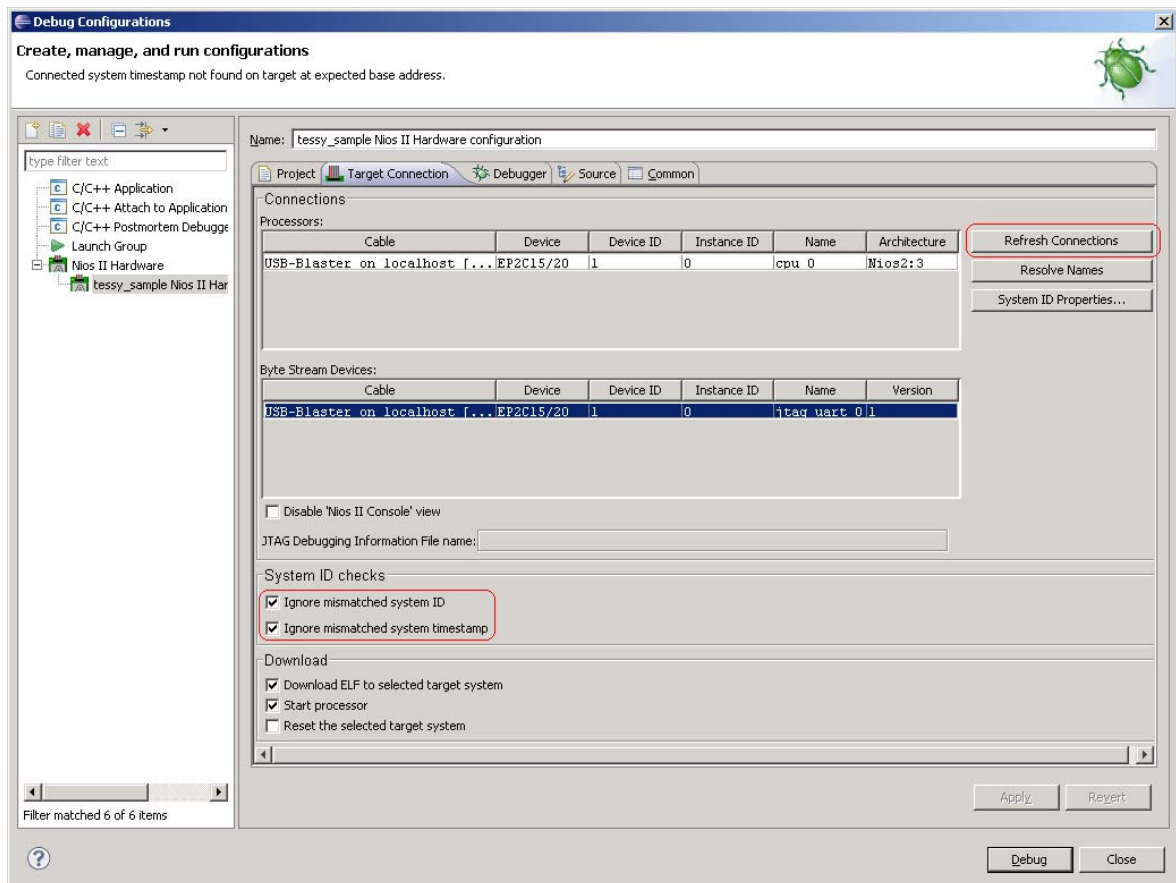


You may then compile, link and run the project on the target board:



**Please note:** Make sure that the target board is already connected and switched on before debugging. If the target connection fails, you may need to restart the Altera NIOS IDE.

When starting the debugging the first time or in case of any problems, the following configuration dialog will appear:



If the **Connections** fields are empty, you need to press the **Refresh Connections** button. This should fill the table with the corresponding board settings. You may need to adjust the device.

It may also be required to disable the **System ID checks** like shown above.

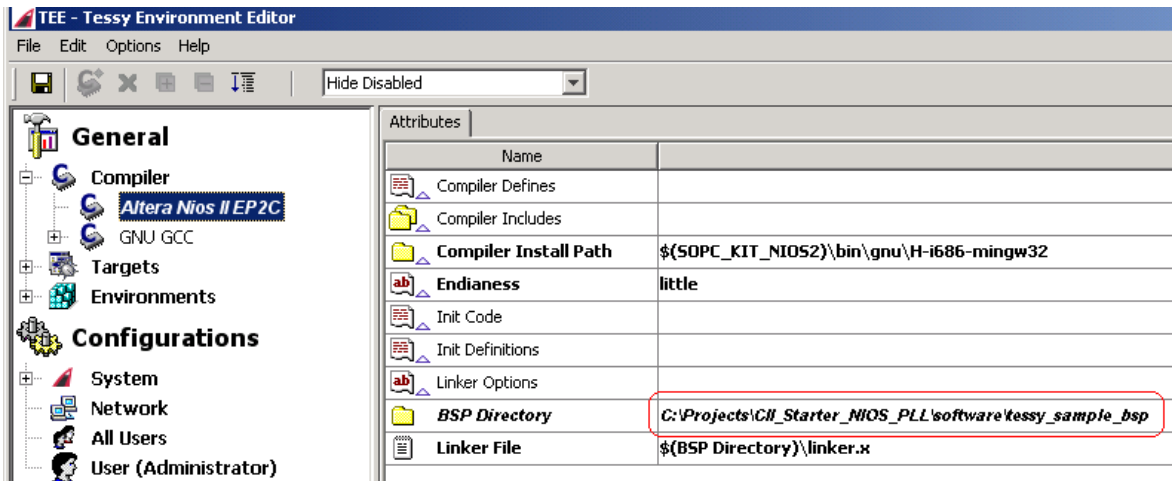
**Please note:** If the connection to the target board fails when pressing the **Debug** button, you may delete this debug configuration and create a new one with the same settings.

Please start the debugger and step through the sample project. If everything works fine, you may start preparing the Tessy settings like described below.

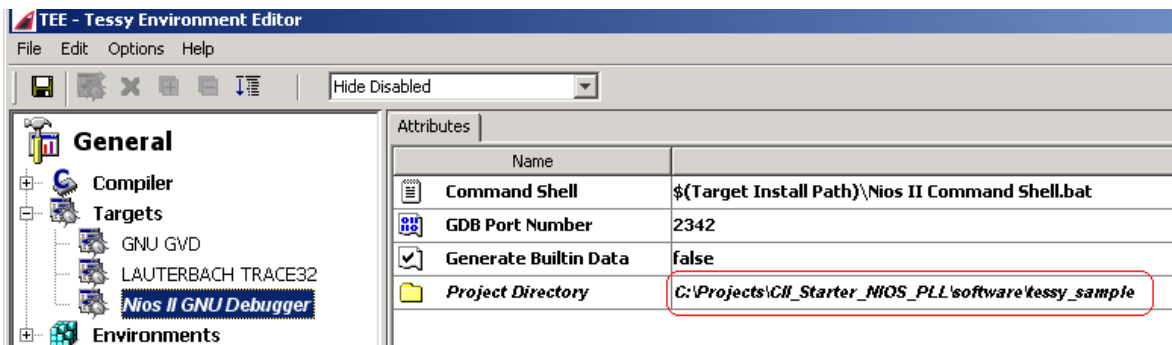
### 3 Preparing necessary Settings within TEE

There are only two settings required within the environment of Tessy. These path settings refer to the BSP package directory where you unzipped the BSP package.

Please open the environment editor TEE and change the path to the BSP within the **BSP Directory** attribute like shown below:



The other path is the path to the newly created project which should be entered into the **Project Directory** attribute like shown below:

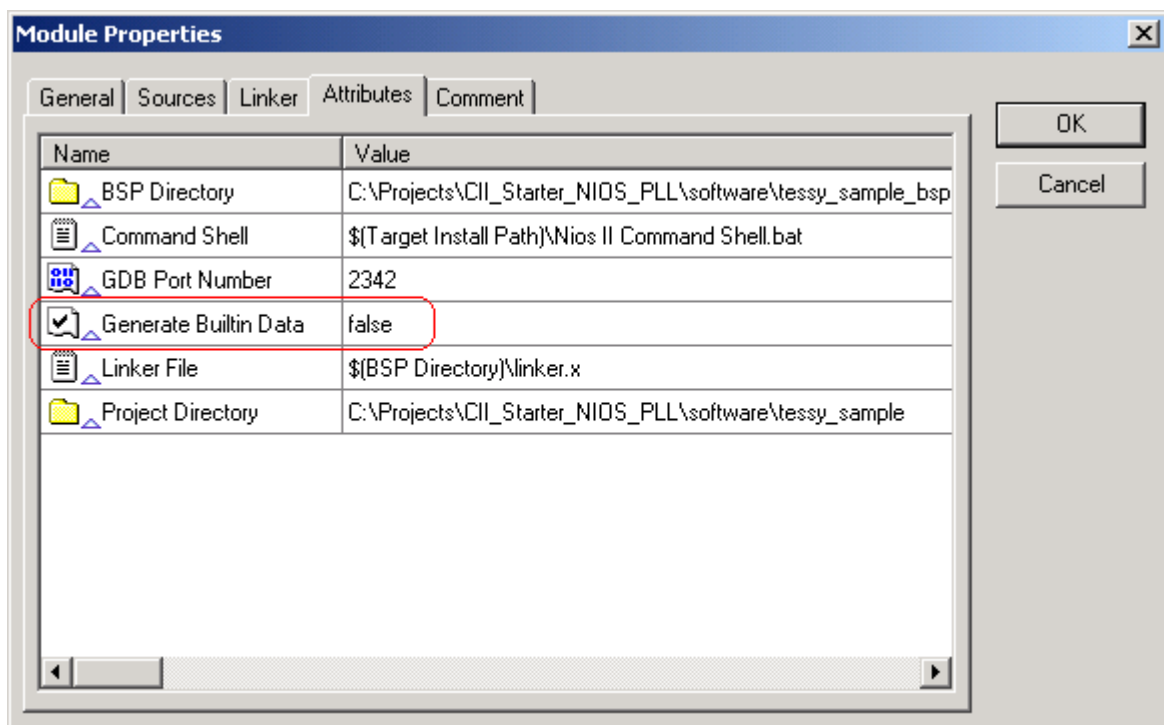


Save these settings and close the TEE.

## 4 Running the Test

You may now create a Tessy module, select the respective Altera NIOS II environment and execute the tests.

The mode of test execution depends on the **Generate Builtin Data** setting within the module properties. This setting is **false** by default, which indicates that the test shall be executed without GUI by running the Altera NIOS II GDB debugger from the command line.



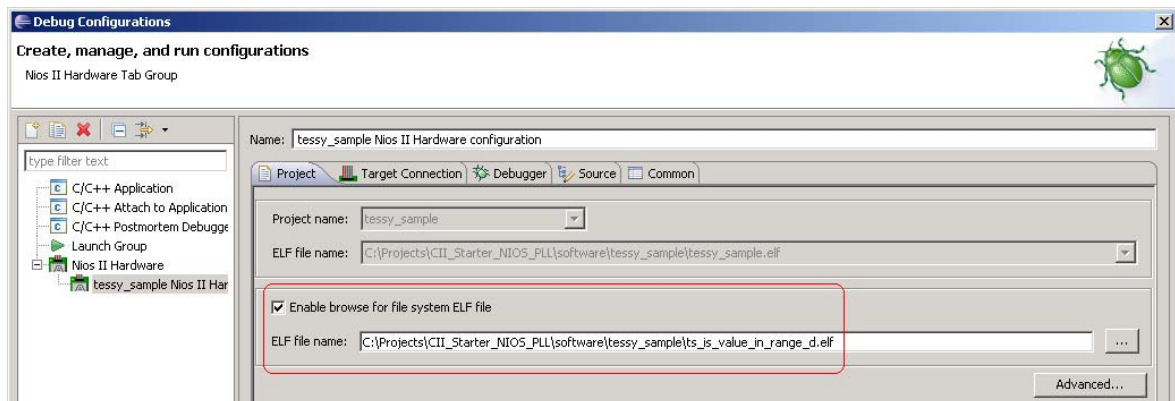
If you change this setting to **true**, Tessy will generate a test application with the input data built into the application. You may download this .elf file and debug it within the Altera NIOS II Eclipse IDE.

### 4.1 Executing Tests Automatically

With the **Generate Builtin Data** setting set to false, Tessy will build the test driver, compile and link everything and start the Altera NIOS II GDB debugger on the command line. The test will be executed automatically without controlling the Altera NIOS II Eclipse GUI.

## 4.2 Debugging the Test Application

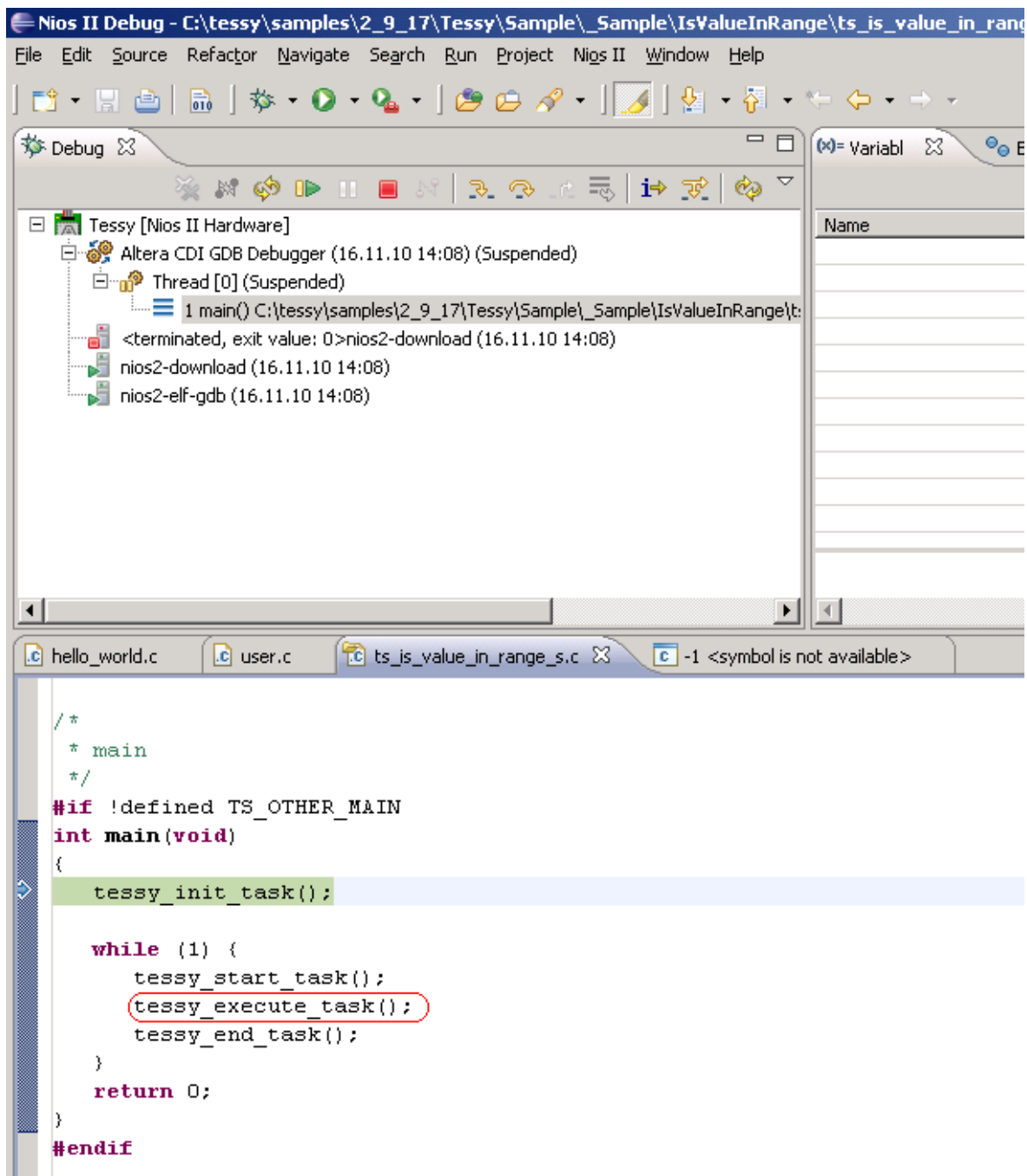
With the **Generate Builtin Data** setting set to true, Tessy will generate a test application binary with the input data for the test object compiled into the application. You may load the resulting .elf file into the Altera NIOS II debugger manually in order to debug the test object.



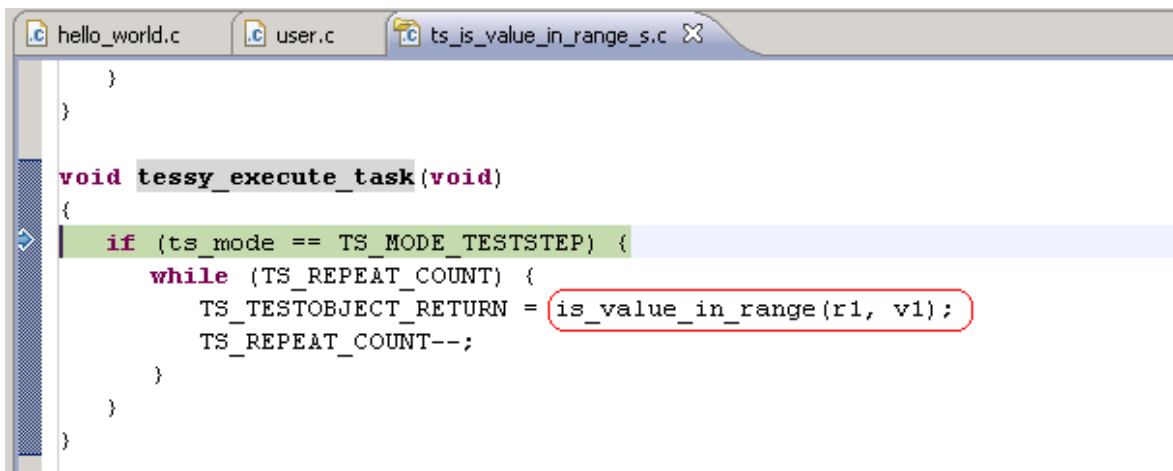
You need to create a new or edit an existing debug configuration and specify the Tessy generated binary .elf file like shown above. The **Enable browse for file system ELF file** button needs to be ticked in order to specify the binary name.

**Please note:** The path specified here needs to be *exactly* the same as the path specified for the TEE attribute **Project Directory** which was described in section 3.

After starting debugging with this debug configuration, you will be able to step through the test application. The main() function of the test application looks like this:



Now, run until **tessy\_execute\_task()** and step into the function. There you will see your test object (e.g. `is_value_in_range()` in this example):



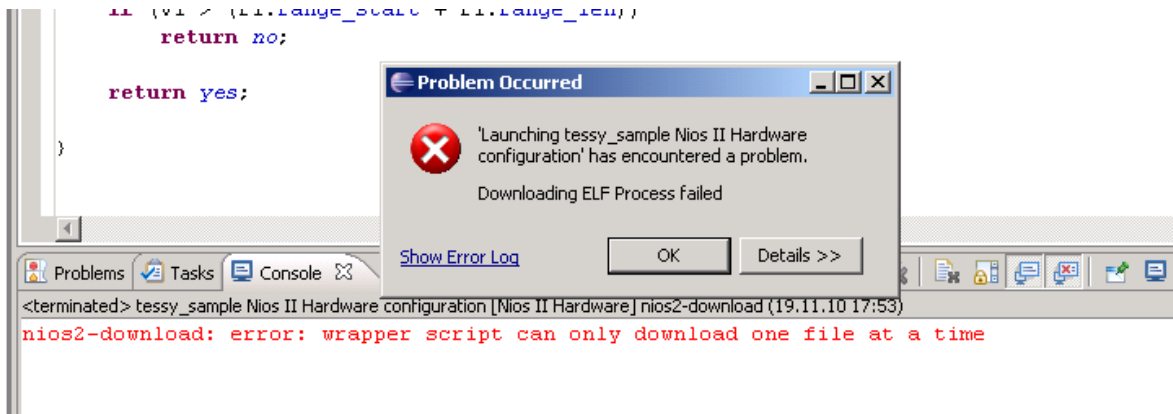
```
hello_world.c user.c ts_is_value_in_range_s.c X
}
}
void tessy_execute_task(void)
{
if (ts_mode == TS_MODE_TESTSTEP) {
while (TS_REPEAT_COUNT) {
TS_TESTOBJECT_RETURN = is_value_in_range(r1, v1);
TS_REPEAT_COUNT--;
}
}
}
```

You may now step into your test object, set breakpoints, and start running to the next test step or test case.

## 5 Troubleshooting

### 5.1 Connection to the Target Board fails

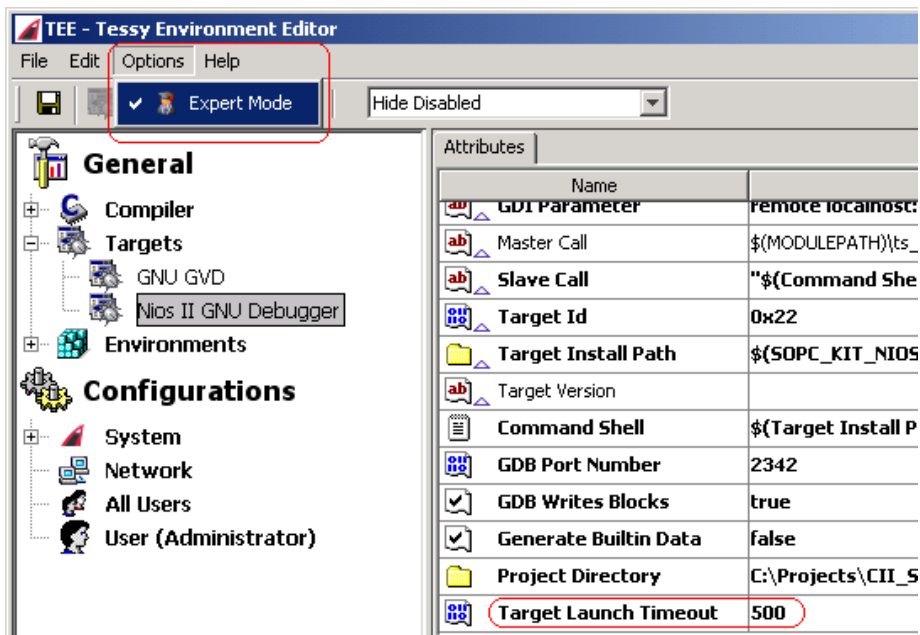
If the connection to the target board fails, you may get an error message like shown below:



In this case, simply remove the current debug configuration and create a new one.

### 5.2 Problems when Running the Test

In some situations, the launch of the Tessy processes required for communication with the Altera NIOS II GDB debugger may have synchronization problems. In this case, you may increment the **Target Launch Timeout** attribute within TEE.



The default value of 500 milliseconds should be ok for most cases, you may increment this value, if there are problems running the test. In order to see this attribute, you need to switch to **Expert Mode** within TEE like shown above.