

# TESSY Hardware Adapter Interface (THAI)

## Abstract

This document describes the hardware adapter interface that TESSY provides to enable stimulation and measurement of hardware signals as well as execution time measurement during the unit test execution.

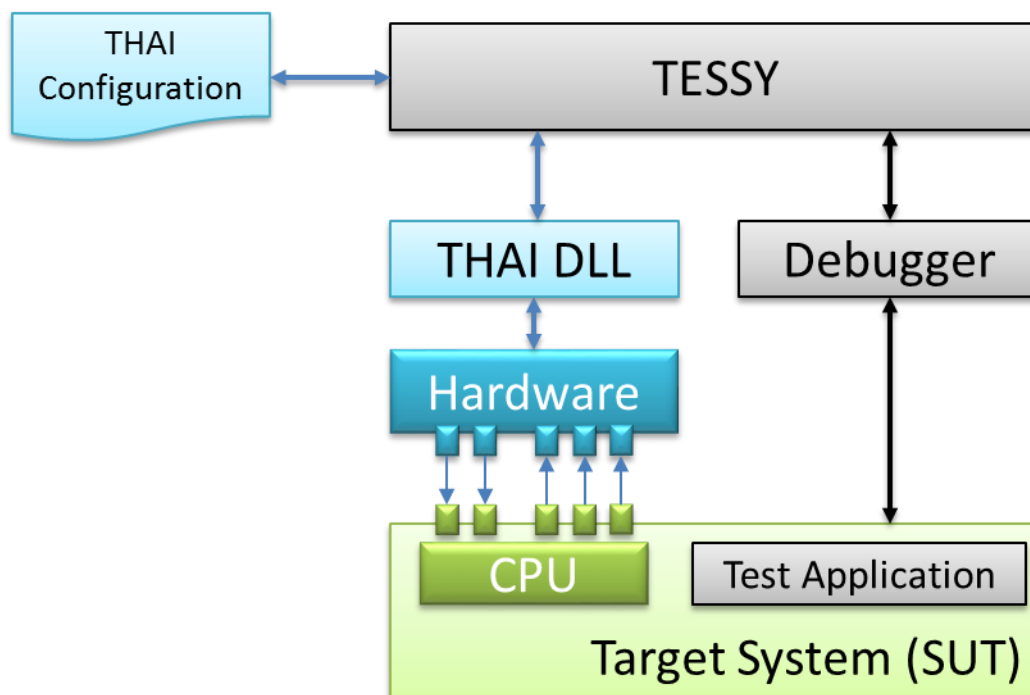
## Table of Contents

Abstract .....	1
1 Introduction.....	3
2 Configuration file .....	4
3 Sequence diagram for hardware device control .....	4
4 Interface DLL.....	6
4.1 THAI_connect(configuration file) .....	6
4.2 THAI_reset_signals().....	6
4.3 THAI_set_signal(name, value) .....	6
4.4 THAI_flush_signals() .....	6
4.5 THAI_get_signal_measurements().....	6
4.6 THAI_get_signal(name, ...).....	6
4.7 THAI_disconnect().....	6
4.8 THAI_get_last_error(...) .....	7
4.9 THAI_specify_logging(...) .....	7
4.10 THAI_get_time(...) .....	7
5 Attribute settings.....	8
5.1 Fill the required THAI Attributes .....	8
5.2 Enable THAI functionality within TEE .....	10
6 Example implementation .....	12
6.1 Building the THAI DLL.....	13
6.2 Run the raspberry pi server.....	13
6.3 Reading/writing digital signals .....	14
6.4 Reading analog signals .....	15

## 1 Introduction

In order to enable hardware I/O stimulation and measurement during unit testing, TESSY provides a hardware adapter interface allowing control of external measurement hardware. This hardware device has to implement a configuration interface as well as reading and writing methods for hardware signal data. The following figure shows the architecture of the system within the TESSY unit testing framework using a Raspberry Pi based HIL system as an example implementation (Refer to chapter 6).

In addition to the normal test execution where the test application is controlled by the respective target debugger (e.g. TRACE32), the hardware I/O pins of the SUT are stimulated and measured using a hardware adapter controlled via the THAI interface DLL.



**Integration of a hardware adapter into the TESSY unit test execution**

During module analysis, TESSY reads the configuration of the hardware device in order to determine the available interface (i.e. the available I/O signals). This list of signals (including passing directions) will appear within the interface of the TESSY module (for each test object). The signals may be edited within the test data editor (TDE) as any other normal test object variable.

During test execution, the input signals will be stimulated from the input values provided within TDE and the actual measurements will be saved to the respective output signals. The synchronization between stimulation/measurement and the unit test execution will be controlled by the debugger running the test object code: Callback functions that stimulate/measure the signal values will be executed before and after calling the test object.

Timing measurement may also be supported by the hardware adapter device, this will be carried out using dedicated pins of the hardware.

## 2 Configuration file

Within the configuration file for the hardware adapter, there is all configuration data necessary for TESSY as well as the configuration data necessary for the hardware device (in XML format). The configuration file will be specified as “**THAI Configuration File**” attribute within the test environment editor (TEE) of TESSY.

TESSY extracts the available hardware signals from the following XML data structure. This example configuration is based on two discrete digital output pins "sw0" and "sw1" of the THAI hardware (which are inputs for the SUT and though inputs for the test) and one digital input port "leds" with 16 bits of the THAI hardware that is connected to some SUT output pins. This output port shall be measured (as an expected output for the test). The required tags and attributes that are mandatory for TESSY are printed in bold.

```
<project name="sample project" description="THAI test project">
  ...more hardware-related tag entries possible

  <signals>
    <signal name="sw0" passing="IN" type="uint8" />
    <signal name="sw1" passing="IN" type="uint8" />
    <signal name="leds" passing="OUT" type="uint16" />
  </signals>
</project>
```

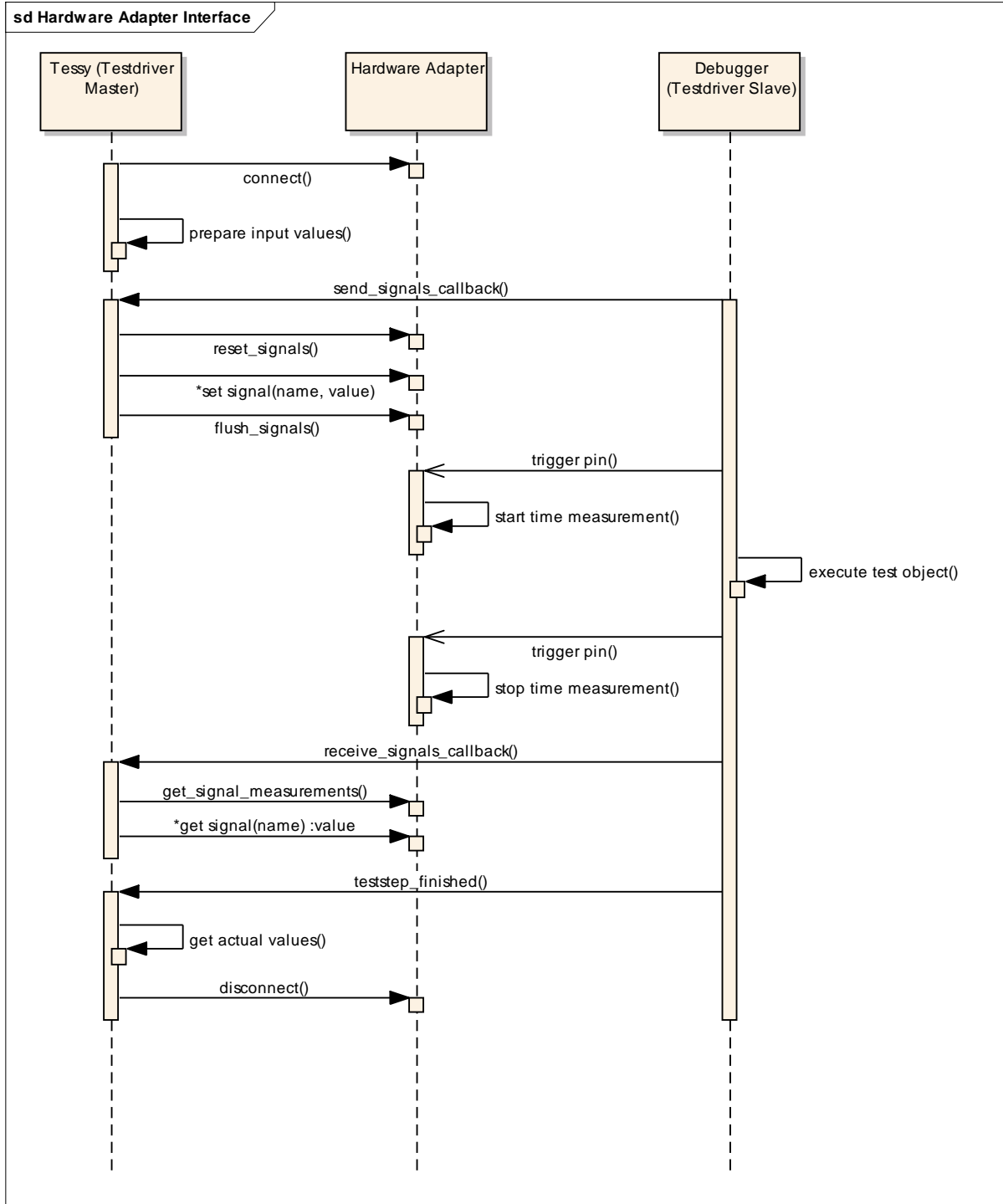
Please note the following important hints:

- The passing direction is from the TESSY point of view, i.e. the passing IN is a signal that shall be stimulated and though the hardware adapter needs to provide an output signal that is connected to the respective input pin of the SUT.
- The currently supported types are uint8, uint16 and uint32.

There may be additional tags and attributes for the hardware device configuration, which will be skipped by TESSY. TESSY will only read the “signals” and the corresponding “signal” tags and will create the available unit test interface from these entries.

## 3 Sequence diagram for hardware device control

The following sequence diagram shows the control flow of the test execution and the access to the hardware device (for one test step). The control flow starting after “connect()” until before “disconnect()” will be repeated for each test step.



## 4 Interface DLL

A DLL implementing the THAI interface functions needs to be created in order to integrate the hardware adapter into the TESSY unit test. The functions described within this chapter need to be implemented within this DLL.

The DLL will be specified as “**THAI DLL**” attribute within TEE. A sample implementation with simulated functions that just print outputs to the TESSY console is available within the installation directory:

```
Program Files\Razorcat\TESSY_4.3\Examples\THAI
```

The makefile builds the DLL from the source file “thai\_sample.c”. The respective header file “thai.h” with the declaration of the DLL functions is also available there.

### 4.1 THAI\_connect(configuration file)

Establishes a connection to the hardware device. The argument is the name of the configuration file containing all necessary configuration information. The implementation needs to get all required information from the configuration file.

### 4.2 THAI\_reset\_signals()

Initiates a sequence of write operations to stimulated signals and successive read operations from the measured signals. This function will be called for each test step.

### 4.3 THAI\_set\_signal(name, value)

Writes the given value to the signal with the given name. The signal value may be in either of the following formats (for type integer):

- Integer number (“1234”)
- Hex number (“0x1234”)

### 4.4 THAI\_flush\_signals()

Writes all stimulation signals (with direction “to-target”) to the hardware device. The hardware device outputs shall now have the values specified for each individual signal.

### 4.5 THAI\_get\_signal\_measurements()

Starts the measurement for all signals with direction “from-target” on the hardware device and keeps the signal values locally (on the device or buffered in the PC). Later calls to THAI\_get\_signal() return the buffered values for each signal.

### 4.6 THAI\_get\_signal(name, ...)

Returns the measured value of the given signal. Depending on the type, the value may be formatted as integer or float number.

### 4.7 THAI\_disconnect()

Disconnects from the hardware device.

#### **4.8 THAI\_get\_last\_error(...)**

Return an error message string related to the last erroneous function call.

#### **4.9 THAI\_specify\_logging(...)**

The log level and the file to be logged into may be specified with this function.

#### **4.10 THAI\_get\_time(...)**

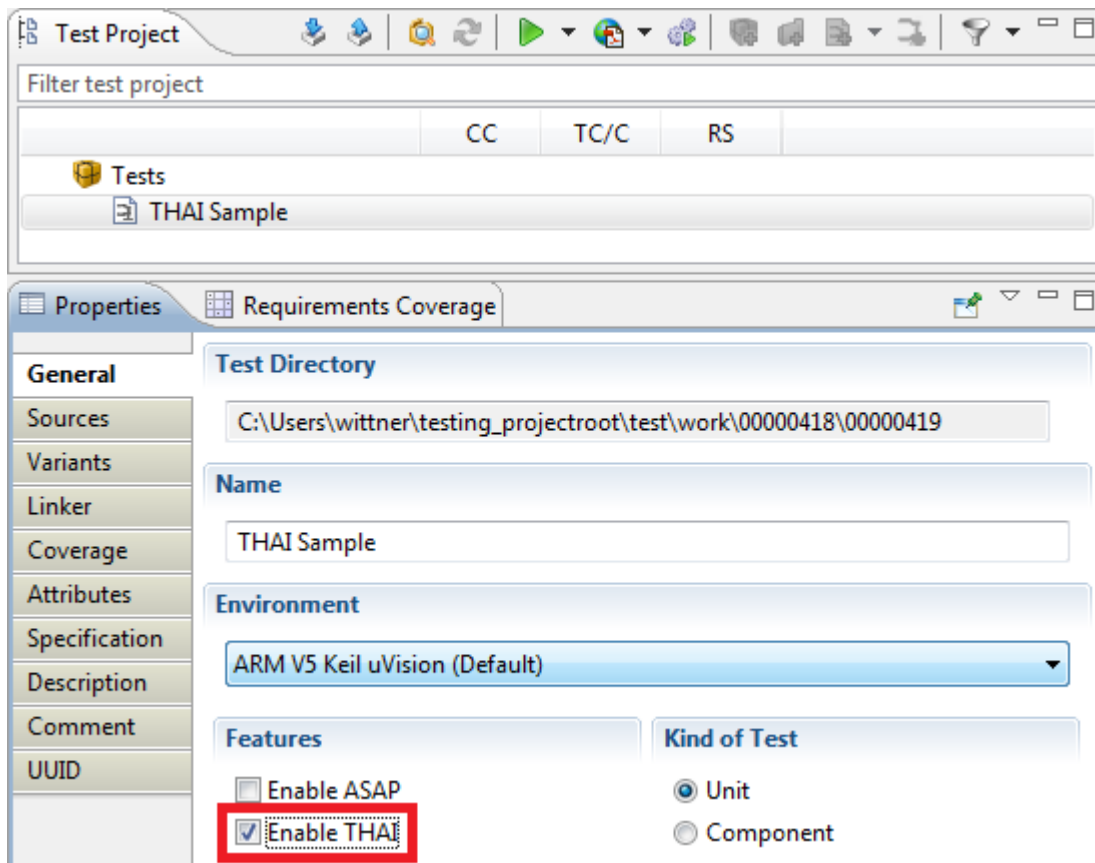
When measuring execution time, this function returns the time between the start and end toggle of a certain hardware pin denoted as timer control pin. If no timer measurement could be done, the function shall return zero as timer value.

## 5 Attribute settings

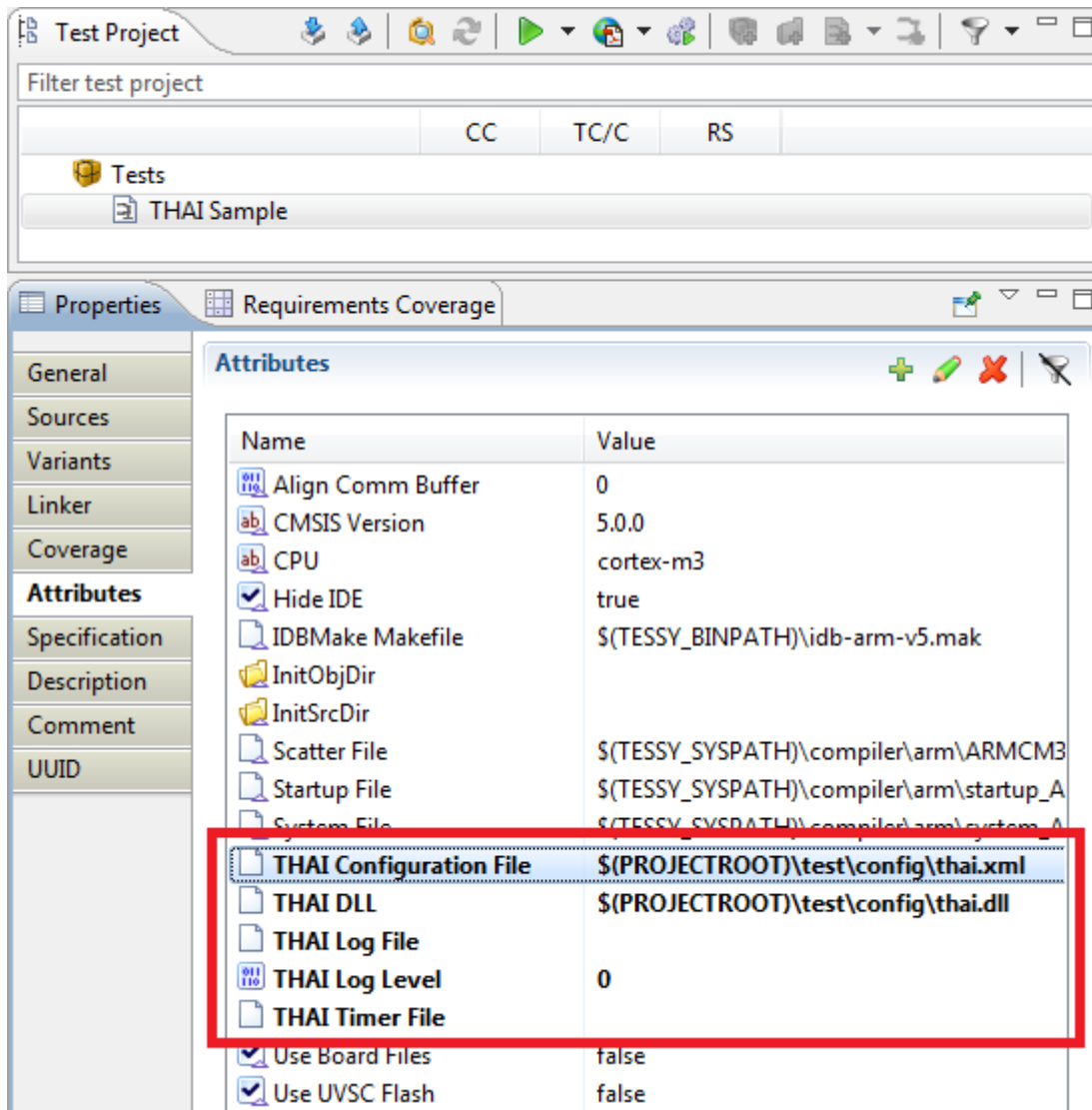
The **Enable THAI** attribute controls the availability of the THAI functionality. It can be enabled on module level as well as within the environment editor (TEE).

### 5.1 Fill the required THAI Attributes

Create a new module within TESSY and enable THAI within the **General** tab of the properties view. Within the **Features** section the **Enable THAI** toggle button will be available.



If you select the **Enable THAI** check box, TESSY will add the required THAI attributes within the **Attributes** tab. Switch to the **Attributes** tab to see the required THAI attributes.

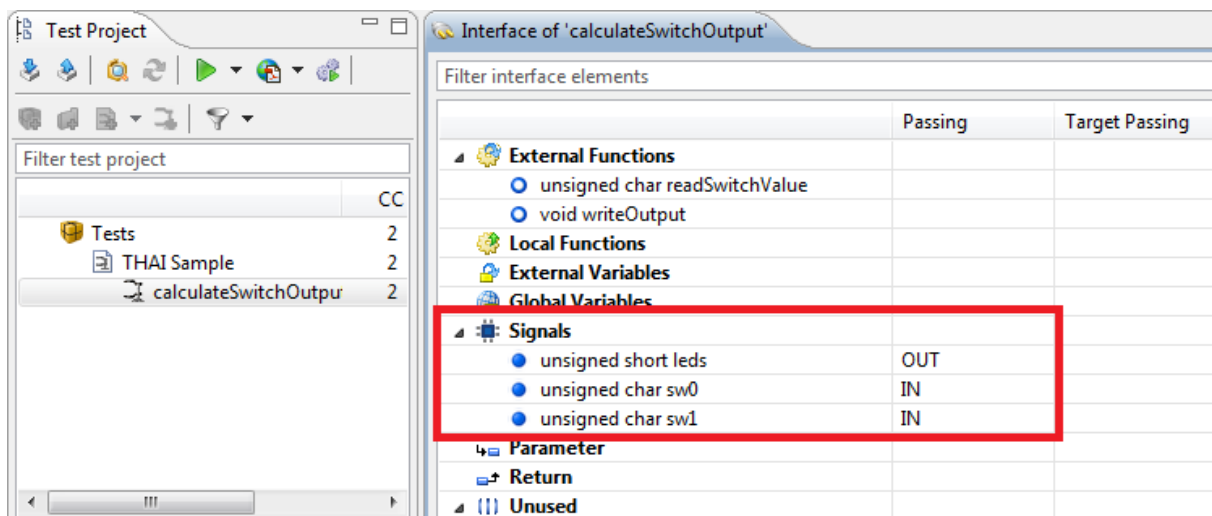


You need to enter the **THAI Configuration File** attribute. Select a suitable configuration file for your hardware device. The **THAI DLL** attribute references your implementation DLL of the THAI interface. The **Log Level** attributes are optional.

The **THAI Timer File** attribute is optional and can be left empty. You will need to set this attribute, if you are using the timing measurement feature.

Attribute	Description
Enable THAI	Activates the THAI functionality.
THAI Configuration File	This references the configuration file for your hardware device including the available hardware interface description for TESSY.
THAI DLL	References your implementation DLL of the THAI functionality.
THAI Log Level	Value from 0 (no logging) to 2 (full logging). The actual log content depends on the implementation of the DLL.
THAI Log File	The output file for logging
THAI Timer File	References a C source file with an implementation of the <code>ts_start_timer()</code> and <code>ts_stop_timer()</code> functions. The default implementation containing empty functions can be found within file <code>\$(TESSY_SYS)\src\comm\ts_timer.c</code> . These functions shall toggle a specific hardware pin for timing measurements.

After analyzing the module, the signals will be shown within the interface of the test object.



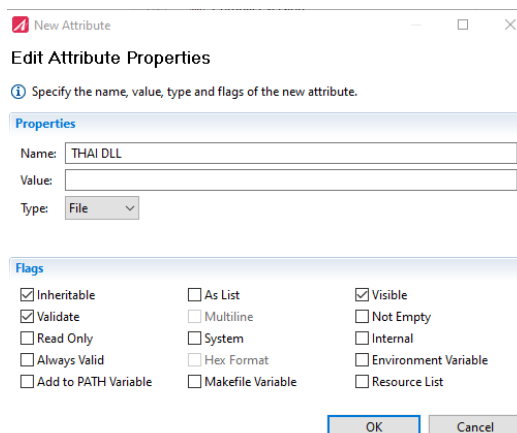
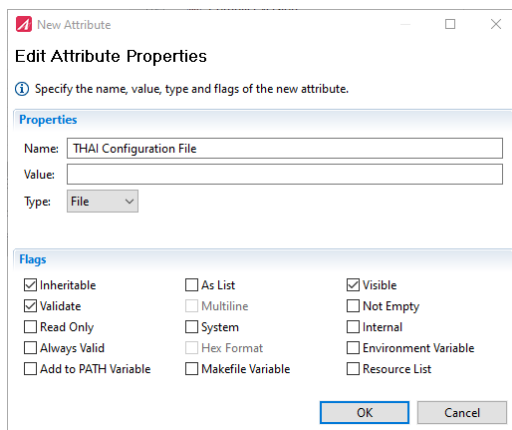
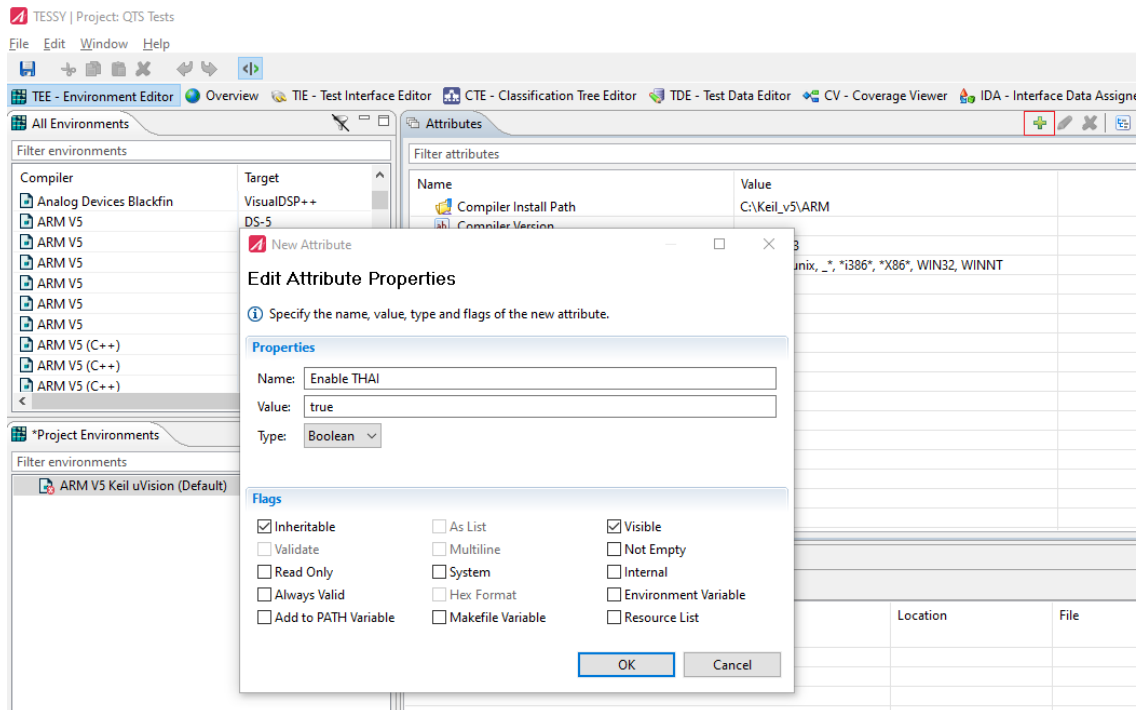
## 5.2 Enable THAI functionality within TEE

In order to enable the THAI functionality globally for all modules of your project, you can create the attributes mentioned within section 5.1 manually using the environment editor TEE. It is recommended to add the attributes within the **File** node within TEE:

Create at least the following attributes with the corresponding type:

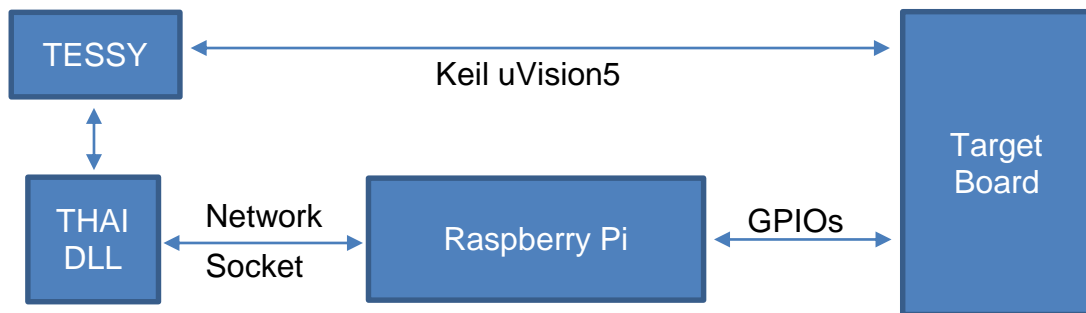
- Enable THAI (Boolean)
- THAI Configuration File (File)
- THAI DLL (File)

# TESSY Application Notes



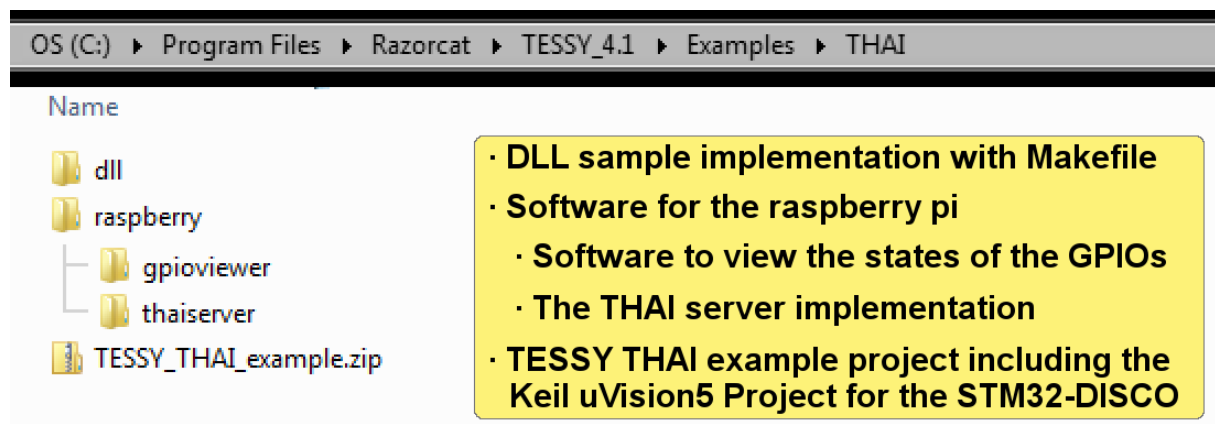
## 6 Example implementation

The following example shows you how to use a raspberry pi as a testing device to provide an interface between TESSY and the hardware board. For this demonstration we will be using an STM32F0308-DISCO as the target platform and Keil uVision5 as the run-time environment. Please make sure to have the appropriate uVision5 board packages installed on your local computer (using the uVision5 Pack Installer).



To allow TESSY to run the tests on the target platform, you need to configure the environment as well as the IDE that you usually use to program the microcontroller. To simplify this for you there is already a preconfigured TESSY and Keil uVision5 project included in your TESSY installation. You can find this and other example files in the Examples directory of your TESSY installation:

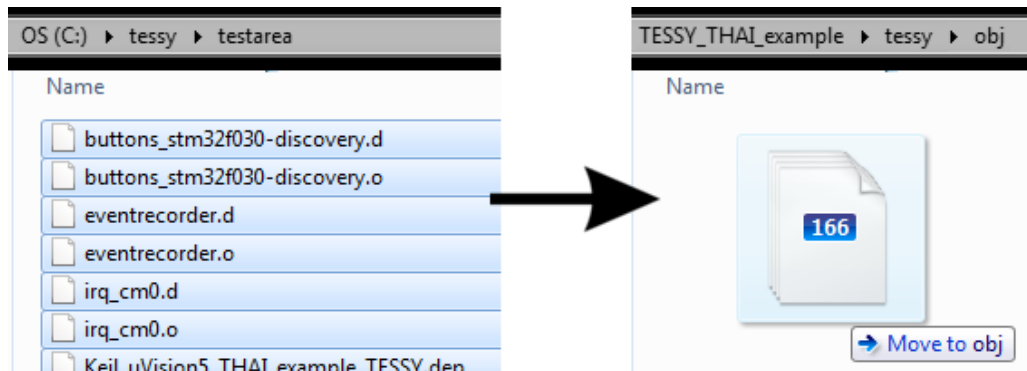
```
..\Program Files\Razorcat\TESSY_4.3\Examples\THAI
```



Before running any tests, you need to build the Keil project and place the generated files (excluding the main.o / main.d) which will be outputted to

C:\tessy\testarea

into the tessy\obj folder of the example TESSY project.

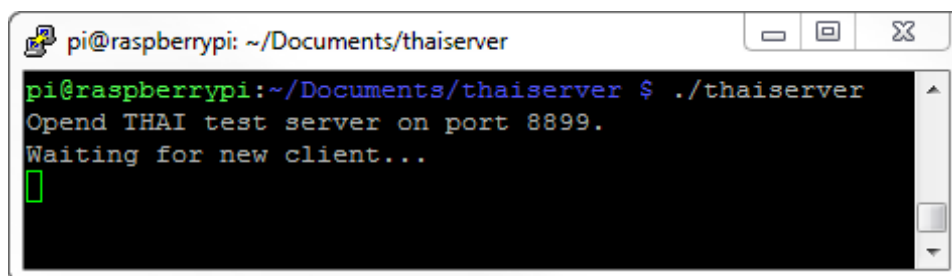


## 6.1 Building the THAI DLL

TESSY will connect to the raspberry pi using a network socket. The required logic for that is already implemented in the thai\_sample.c. This C file can be compiled to a DLL using the included Makefile. If you get an error while compiling you are probably missing the libxml2 library which can be found on [xmlsoft.org/downloads.html](http://xmlsoft.org/downloads.html). This DLL needs to be provided to your TESSY project as seen in chapter [5.2](#).

## 6.2 Run the raspberry pi server

The required software for the raspberry pi can be found in Examples\THAI\raspberry. The port on which the socket is opened can be configured in the thaiserver.h by modifying the PORT definition (default is 8899). After that you can use the Makefile to compile the server and run it. Please note that you need the wiringPi as well as the libxml2 library installed on your pi to do so (both should be default on raspbian).



To allow TESSY to connect to your server you need to add the IP address and port of the pi to your thai.xml configuration file by adding this line:

```
<server ip="the pi's ip address" port="your portnumber" />
```

```
Example: <server ip="192.168.0.197" port="8899" />
```

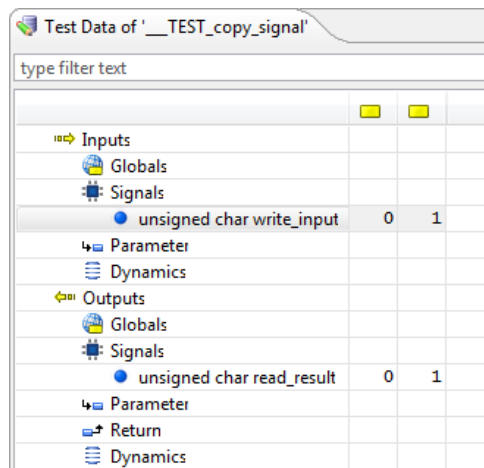
If you configured TESSY correctly you should already be able to establish a connection to the THAI test server on your pi.

### 6.3 Reading/writing digital signals

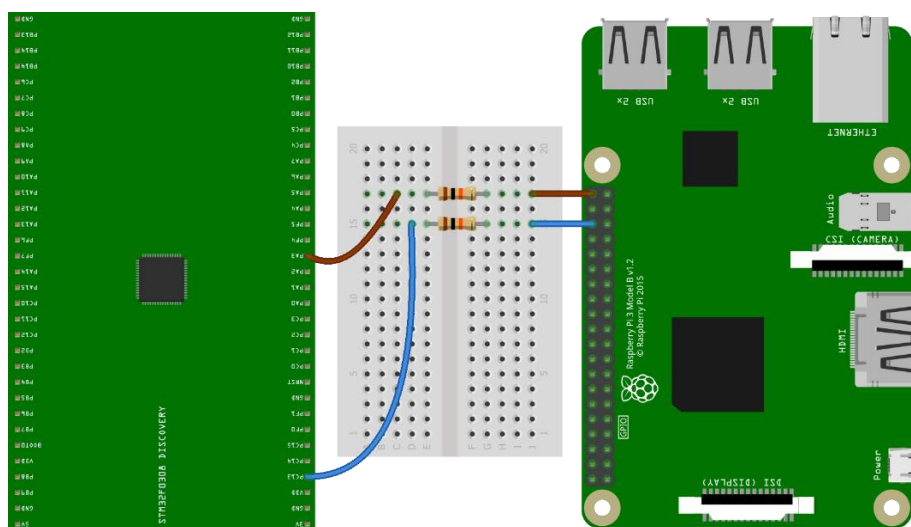
In this example we want to test a function on a target board which should read the state of an input pin and then clone this state to an output pin. The required signals for this test and their specification like passing direction, name or corresponding GPIO pin on raspberry pi needs to be defined in the thai.xml config:

```
<signals>
  <signal name="read_result" passing="OUT" type="uint8" pin="29" />
  <signal name="write_input" passing="IN" type="uint8" pin="27" />
</signals>
```

After analyzing the module, they can be used in test cases:

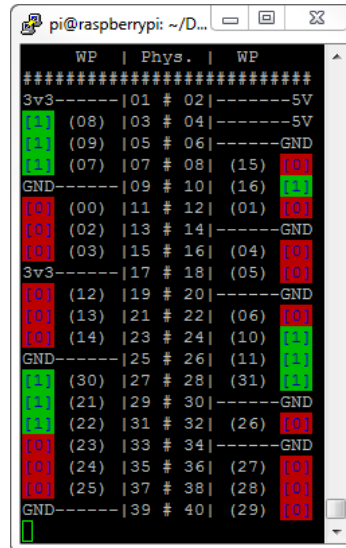


To run the tests from the example project you should connect the boards as shown in the following diagram:



The resistors are only there to protect the boards from overvoltage and they don't need to have the exact same resistance as shown (10kΩ).

If you want to keep track of the pi's GPIO states and get more information about each pin you can compile the `gpioviewer.c` example and run it to get a graphical representation.



## 6.4 Reading analog signals

The raspberry pi is by default not able to read analog signals since it only has digital pins. If you want to achieve that anyways you need an extension board, for example the ADC Differential Pi which will provide 8 analog read-only channels. For simplicity of the following example we are only going to read the value of an analog potentiometer and show the results in TESSY. To tell the thaiserver on the raspberrypi that you want to read an analog signal you need to add the "float" tag to the signal config:

```
<signal name="read_analog_result" passing="OUT" type="uint16" pin="8" float="true" />
```

For easier interpretation of the results in TESSY, the float values will get multiplied by 1000 and then converted to an integer (e.g. 0,723267 → 732) by the raspberrypi. Connect the potentiometer to the pi as shown in the following schematic and run the `analog_example` module in the TESSY project while turning the handle of the potentiometer to see different values in the test cases.

