

Timing Measurements

Abstract

This document describes how to enable the timing measurement features of TESSY and explains the restrictions with respect to accuracy and target hardware dependencies.

Table of Contents

Abstract	1
1 Introduction.....	2
1.1 Measuring Procedure	2
1.2 GNU/GCC based generic timing measurement	3
1.3 Microcontroller based timing measurement.....	4
1.3.1 C166 timing measurements	4
1.3.2 Microcontroller specific timing measurements.....	5
1.4 TRACE32 runtime based measurement.....	6

1 Introduction

TESSY provides different timing measurement implementations which are only available for selected targets and microcontrollers:

- Generic time measurement based on execution cycle count for GNU/GCC
- Execution time measurement based on microcontroller features (e.g. the C166 timer register T3).
- Execution time measurement based on the runtime feature of TRACE32

The timing measurement for GNU/GCC is based on execution cycle count of the PC processor, so that it can only serve as a rough estimate because the number of cycles highly depends on the current work load and other influences of the processor. Subsequent test runs with the same test cases will provide different results in most cases.

The second measurement is available for the C166 microcontroller and its derivatives only but it can serve as a template for own timing measurement implementations on your specific microcontroller. The third measurement is available for all microcontrollers supported by TESSY in conjunction with TRACE32.

The raw execution time values are converted into time durations using user definable conversion parameters. Refer to the following sections for details on how to adapt these settings to your specific hardware configuration.

1.1 Measuring Procedure

The timing measurements are executed using start/stop timer function calls before and after calling the test object within the test driver. This introduces slight deviations of the timing results from the actual execution time of the test object itself. The execution sequence during TESSY test execution is as follows:

- call the `ts_init_timer` function once
- for each test step:
 - o call the `ts_start_timer` function
 - o call the test object
 - o call the `ts_stop_timer` function

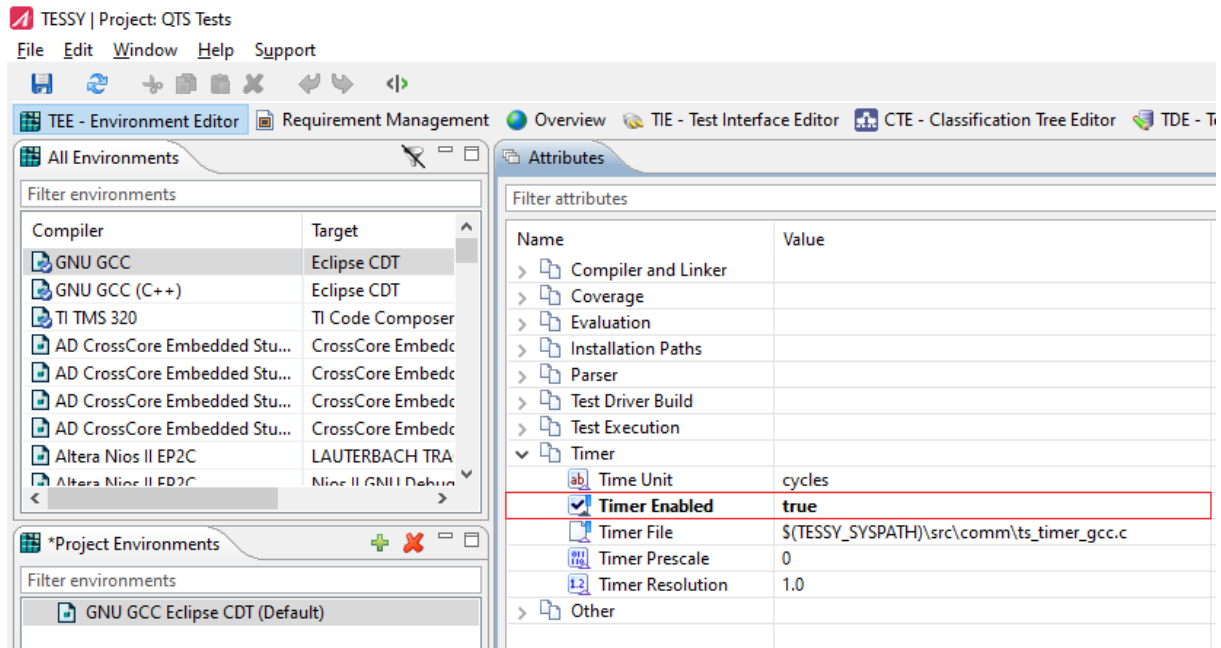
This sequence introduces a number of extra CPU cycles that are included into the total test object execution time measurement.

Important note: *When measuring execution time for very short functions, this may introduce a quite high deviation from the actual execution time of the test object.*

The TRACE32 runtime measurement will also be started and stopped using conditional breakpoints on the start/stop timer functions. In this case, the timer functions are just dummy functions for use with the TRACE32 timer breakpoints.

1.2 GNU/GCC based generic timing measurement

The timing measurement for GNU/GCC is based on CPU cycles of the host PC. It can be enabled within the environment editor (TEE) by setting the **Timer Enabled** attribute to “true” as shown below:



The timing measurement will appear within the test report after a successful test execution as shown below showing the shortest and longest timings:

Execution Time Measurement	
Shortest Test Step	Test step 6.1, 4748.00 cycles
Longest Test Step	Test step 3.1, 6704.00 cycles
Measurements	
Test Step	Duration
1.1	5280.00 cycles
2.1	5984.00 cycles
3.1	6704.00 cycles
4.1	6048.00 cycles
5.1	6188.00 cycles
6.1	4748.00 cycles
7.1	5960.00 cycles

1.3 Microcontroller based timing measurement

Timing measurement on a microcontroller requires timer registers or other timing features being available on the respective controller. As an example, TESSY provides an implementation for the timer registers of the C166 family of microcontrollers.

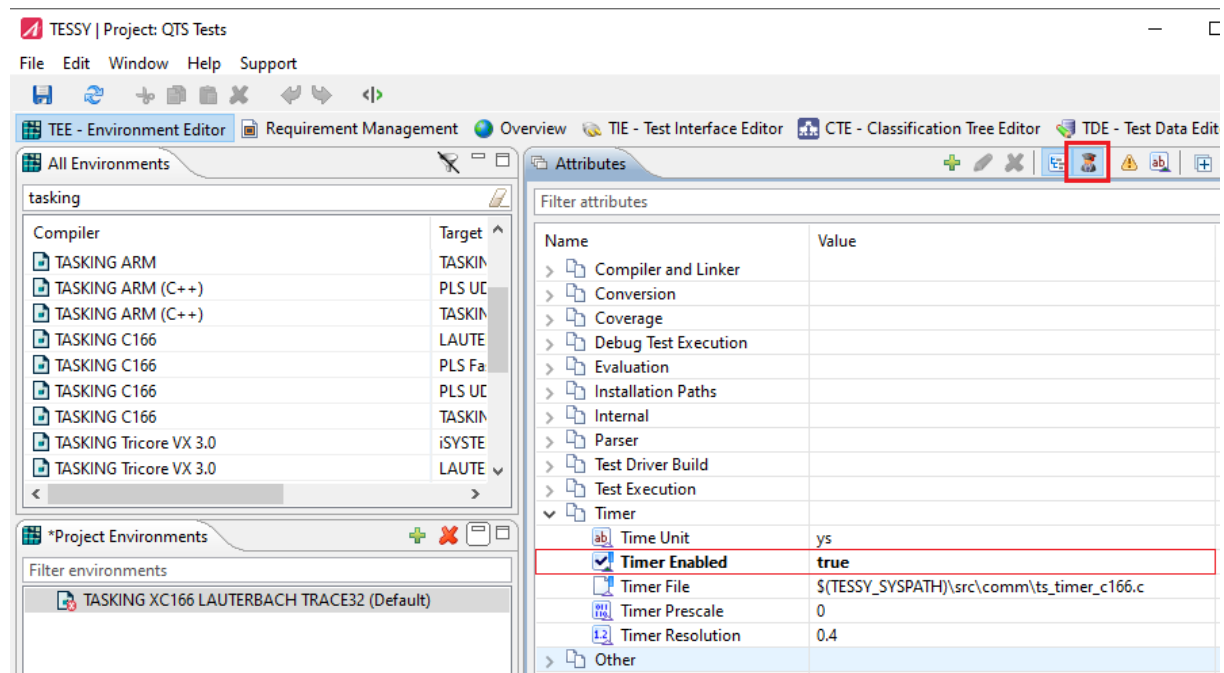
1.3.1 C166 timing measurements

The timing measurement on the C166 controller family uses the timer register T3. Based on the timer ticks as raw values, TESSY calculates the execution time. The time for each tick of timer T3 depends on the settings of the timer control register T3CON and on the actual frequency of the controller in use. Refer to the C166 data sheets for details on the T3 timer settings.

Within the **Environment Editor (TEE)**, you can enable the timing measurements for the following compilers:

- TASKING C166/XC166
- Keil C166/XC166

You may adjust the timer T3 prescale factor and the calculated time duration based on the T3 timer raw value as follows:



The timer settings above have the following meanings:

- The **Time Unit** is the unit description that is used to display the calculated time duration within the TESSY report.
- Setting the **Timer Enabled** value to true enables the timing measurement.

- The **Timer File** attribute contains the implementation of the init/start/stop functions of the timing measurement.
- The **Timer Prescale** value will be used to set the prescale bits of the T3CON register. This setting changes the time duration that each timer tick represents.
- The **Timer Resolution** value will be used to calculate the printed time duration from the timer T3 raw value. In the example above, one timer tick represents 400 ns execution time, so that we will get microseconds (μs) when multiplying the tick count with "0.4".

Please note: The timing measurement depends on the frequency of the microcontroller, the prescale factor for the T3 register and the **Timer Resolution** value specified within the environment editor.

The implementation of the timing measurement is done within the file "ts_timer_c166.c" specified within the **Timer File** attribute. This implementation is specific for the C166/XC166 controller.

1.3.2 Microcontroller specific timing measurements

If you want to implement timing measurement for your currently used microcontroller, you can adapt the file "ts_timer_c166.c" which is available for the C166 microcontroller family (e.g. use another timer register or another timing resource available on the respective microcontroller).

The following functions need to be implemented:

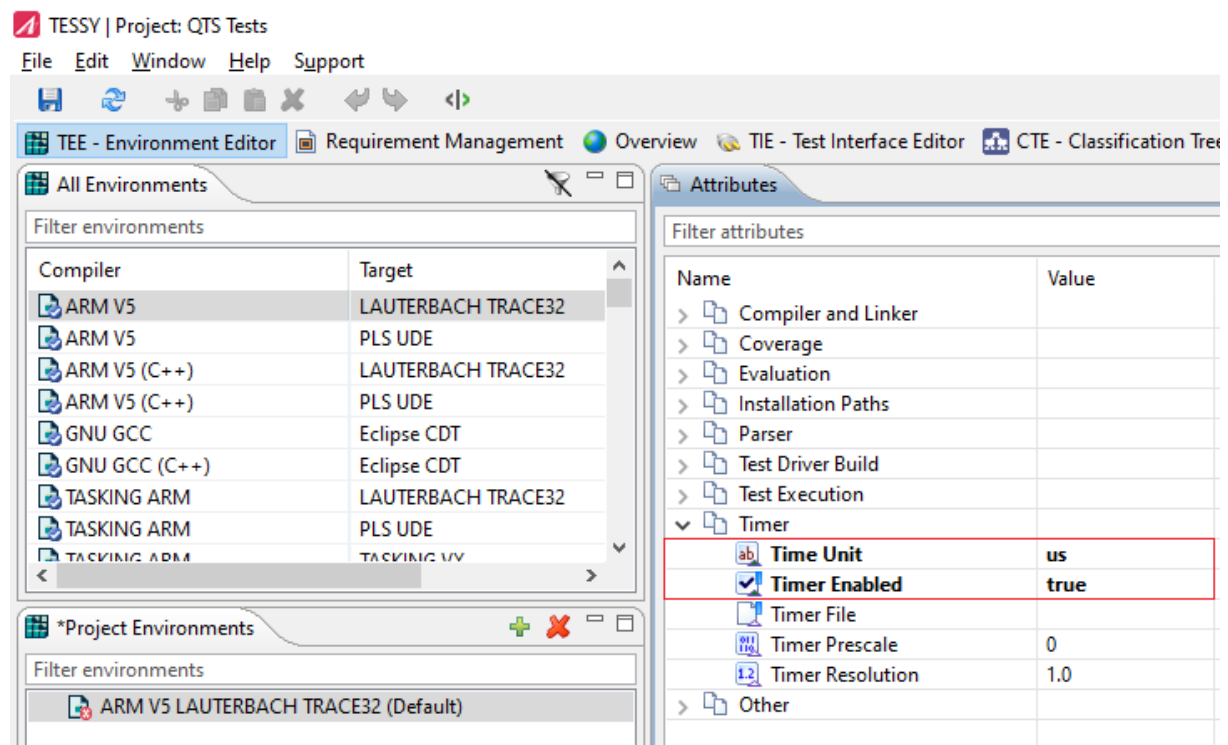
Function	Description
ts_init_timer	Initializes the timing measurement. Use this function to setup timer registers or other sort of initialization.
ts_start_timer	Starts the timing measurement for one test step.
ts_stop_timer	Stops the timing measurement for one test step. The result of the measurement (the integer raw value) needs to be stored temporarily.
ts_send_timer_result	Sends the measurement result of the last test step (the integer raw value) to the TESSY master process.

1.4 TRACE32 runtime-based measurement

When using TRACE32 as target environment, the timing measurements are based on the RUNTIME feature of TRACE32. TESSY uses the same start and stop timer functions as described in the above measurement procedure. This also introduces the above-mentioned deviation of the actual test object execution time.

Important note: TESSY gets the execution time in " μ s" from the TRACE32 RUNTIME function. You should verify that the given measurements reflect the actual time duration of the test object with respect to the frequency of the microcontroller in use.

The respective entries within the **Environment Editor (TEE)** are as follows:



The resolution of the timer is 1μ s. To enable the timing features, you need to change the **Timer Enabled** value to "true" and **Time Unit** to "us".