

# Using KungFU GDB Debugger

## Abstract

This document describes the usage of the KungFu32 GDB debugger as target debugger. At the time of writing this document version 7.8 of the debugger was tested.

**Important Note:** You need a **functional** ChipON IDE development project which can successfully build a target binary and launch a debug session.

**Please note:** The KungFU GDB debugger adaption does not support interactive debugging features when executing tests with TESSY. (See 4 to learn how to debug interactively having your test data statically built into the target binary.)

## Table of Contents

Abstract .....	1
1 Introduction.....	2
2 TESSY Environment Settings.....	2
3 Parallel Built and Parallel Execution .....	2
4 Interactive Debugging.....	3
5 Troubleshooting.....	4
5.1 The debugger crashes with signal SIGSEGV, Segmentation fault.....	4

## 1 Introduction

The communication between TESSY and the KungFu32 GDB debugger is based on `kf32-gdb.exe` which comprises an emulator and a simulator. The `kf32-gdb.exe` is started by TESSY by executing the command line found in TEE attribute **GDB Client Debugger**. TESSY also reads all commands of the file pointed to by TEE attribute **GDB Client Script**. After startup these commands are sent to the debugger. The KungFu32 simulator is supported by TESSY's **KungFu32 GDB Debugger** environment. For the KungFu32 emulator choose TESSY's **KungFu32 Emulator** environment.

In order to debug the test application interactively with the test case values provided from within TDE you need to rebuild the test application in a special mode, i.e. the input values will be compiled into the application. Unfortunately, it is not possible to load this generated binary into the ChipON IDE by now. So, you will have to utilize the `kf32-gdb.exe` from command line. Please refer to chapter 4 for further details.

## 2 TESSY Environment Settings

At first, set the **Compiler Install Path** and the **Target Install Path** and check which errors remain after toggling the **Show Errors/Warnings** toggle button, which is found in the **Attributes** view's toolbar. Please adjust all remaining unresolved paths being displayed. Next, check the TEE attributes **Architecture**, **Chip Kernel Library**, **CPU**, and **Device Port**. The latter is needed for the emulator only. For the emulator, check also the TEE attributes **KF Instruct**, **KF Voltage**, **KFD Code**, **KFD Speed**, **KFT Code**, and **Programmer Assistant**. On Windows 7 the **GDB Device Port** differs from the **Device Port** given to `ProgrammerAssistant.exe`. For example, `COM28` becomes `/dev/ttyS27`. On Windows 10 or higher **Device Port** and **GDB Device Port** are the same. So, **GDB Device Port** is set in TEE to `$(Device Port)` by default.

Collect your startup code files from the ChipON IDE project and copy the object files into a new folder of your TESSY project. Let TEE attribute **InitObjDir** point to this folder and make sure TEE attribute **Use Board Files** is set to `true`. Special compiler and linker settings can be adapted in variable `S_COMP_OPTIONS` respectively `S_LINK_OPTIONS` of the Makefile template given in TEE attribute **Makefile Template**.

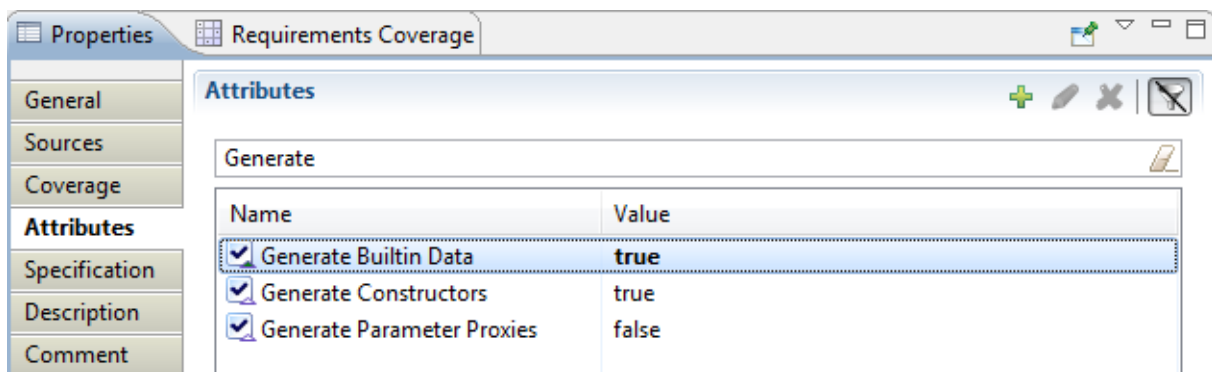
## 3 Parallel Built and Parallel Execution

The amount of test objects built in parallel is given by **Compiler Concurrency**, which is by default set to 4, while 20 source files per test object are compiled in parallel. So, no more than 80 processes run by default in parallel for the built. You can disable the parallel built by setting TEE attribute **Compiler Concurrency** to 1 and by clearing the TEE attribute **Make Options**. The parallel execution is set to 10 by default. Too many compiler or debugger processes in parallel may slow down the test execution. It depends on your system. So, feel free to test different values for **Compiler Concurrency** and **Target Concurrency**.

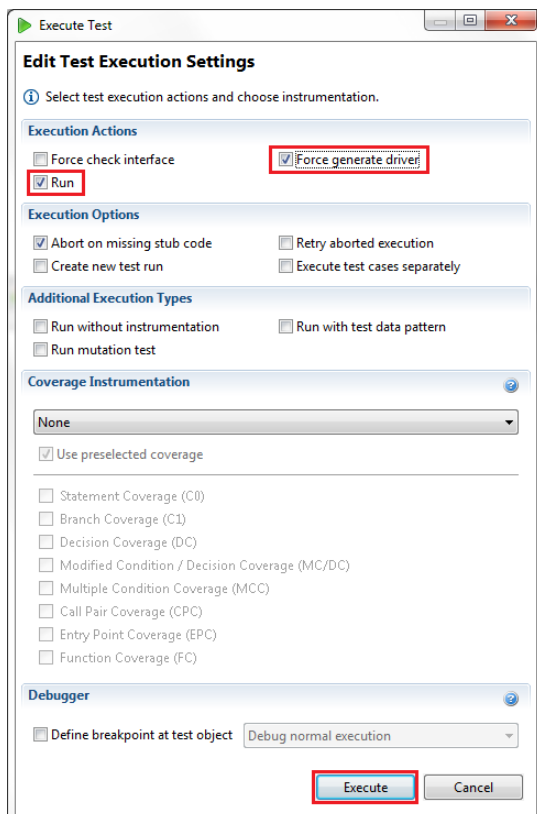
## 4 Interactive Debugging

In order to debug the test application interactively with the test case values provided from within TDE you need to rebuild the test application in a special mode, i.e. the input values will be compiled into the application.

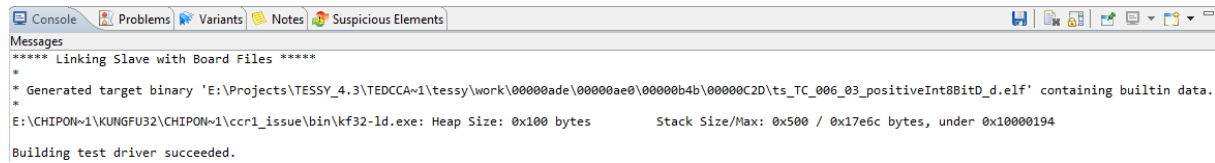
The TESSY KungFu GDB debugger adaption does not support interactive debugging during a test run. But it is possible to debug your test object interactively having the test data built-in which might be useful in case of errors during a test run. So, in order to debug the test object interactively TESSY provides the TEE attribute **Generate Builtin Data**. The attribute is of type Boolean and, if set to *true*, TESSY will rebuilt your target binary during the next test run having the selected test data built-in, i.e. TESSY will not actually perform the test run but instead create the target binary with test data built-in to it. To disable this feature, you have to set the attribute to *false*.



Open the **Execute Test** dialog and make sure **Force Generate Driver** is selected.



Now execute the test by pressing the **Execute** button. TESSY displays the path to the generated built-in target binary in the **Console** view.



```
Console Problems Variants Notes Suspicious Elements
Messages
***** Linking Slave with Board Files *****
*
* Generated target binary 'E:\Projects\TESSY_4.3\TEDCCA-1\tessy\work\00000ade\00000ae0\00000b4b\00000C2D\ts_TC_006_03_positiveInt8BitD_d.elf' containing builtin data.
*
E:\CHIPON-1\KUNGFU32\CHIPON-1\ccr1_issue\bin\kf32-ld.exe: Heap Size: 0x100 bytes      Stack Size/Max: 0x500 / 0x17e6c bytes, under 0x10000194
Building test driver succeeded.
```

You can copy the path of the generated target binary from TESSY's **Console** view. Unfortunately, it is not possible to load this generated binary into the ChipON IDE by now. So, you will have to utilize the `kf32-gdb.exe` from command line with the path to the generated binary as its only argument. *When using the emulator, i.e. target hardware, you will have to load the hex-file into the target device at first using the `ProgrammerAssistant.exe` tool.* Now, start the debugger and copy the commands from the GDB client script and paste them into the debugger's command line. Next, connect the target by entering `target` plus the contents of the TEE attribute **GDI Parameter**. Substitute beforehand the `$(Device File)` by the contents of the TEE attribute **Device File**. For example:

```
target kf32sim --arch="C:/ChipON IDE/KungFu32/ChipONCC32/ccr1_issue/scripting/KF32LS520MQV.def"
```

Enter `file` followed by the path to the elf-binary in order to load the symbols into the debugger. Now, enter `load` and set a breakpoint to your test object, for example

```
break foo
```

assuming that the test object is named `foo()`. Finally, enter `run` to start the program and to reach the test object under test.

## 5 Troubleshooting

Please contact [support@razorcat.com](mailto:support@razorcat.com) if you encounter any unsolvable problems.

### 5.1 The debugger crashes with signal SIGSEGV, Segmentation fault

Please, check your float and double values. Also, `static double` values may cause a crash. Make sure that there are no unresolved symbols and make sure the stack size (see `vector.c` of your ChipON IDE project) is large enough. The stack size should have at least 0x300 bytes.