GTM-RM

GTM Reference Model

Users Guide

Date: 10.04.2014

(Revision 1.5.5-01)

Robert Bosch GmbH
Automotive Electronics (AE)

LEGAL NOTICE

The content of this document is subject to continuous developments and improvements. All particulars and its use contained in this document are given by BOSCH in good faith.

NO WARRANTIES: TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON, EITHER EXPRESSLY OR IMPLICITLY, WARRANTS ANY ASPECT OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING ANY OUTPUT OR RESULTS OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO UNLESS AGREED TO IN WRITING. THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS BEING PROVIDED “AS IS”, WITHOUT ANY WARRANTY OF ANY TYPE OR NATURE, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY THAT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS FREE FROM DEFECTS.

ASSUMPTION OF RISK: THE RISK OF ANY AND ALL LOSS, DAMAGE, OR UNSATISFACTORY PERFORMANCE OF THIS SPECIFICATION (RESPECTIVELY THE PRODUCTS MAKING USE OF IT IN PART OR AS A WHOLE), SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO RESTS WITH YOU AS THE USER. TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON EITHER EXPRESSLY OR IMPLICITLY, MAKES ANY REPRESENTATION OR WARRANTY REGARDING THE APPROPRIATENESS OF THE USE, OUTPUT, OR RESULTS OF THE USE OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, BEING CURRENT OR OTHERWISE. NOR DO THEY HAVE ANY OBLIGATION TO CORRECT ERRORS, MAKE CHANGES, SUPPORT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, DISTRIBUTE UPDATES, OR PROVIDE NOTIFICATION OF ANY ERROR OR DEFECT, KNOWN OR UNKNOWN. IF YOU RELY UPON THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, YOU DO SO AT YOUR OWN RISK, AND YOU ASSUME THE RESPONSIBILITY FOR THE RESULTS. SHOULD THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL LOSSES, INCLUDING, BUT NOT LIMITED TO, ANY NECESSARY SERVICING, REPAIR OR CORRECTION OF ANY PROPERTY INVOLVED TO THE MAXIMUM EXTEND PERMITTED BY LAW.

Conventions

The following conventions are used within this document.

ARIAL BOLD CAPITALS	Names of signals
Arial bold	Names of files and directories
`Courier bold`	Command line entries
`Courier`	Extracts of files

References

This document refers to the following documents.

Ref	Authors(s)	Title
GTMSPEC	AE/EIN2	GTM-IP Specification
IFSDOC	AE/EIM3	IFS User's Guide March 2010
TBDOC	AE/EIN2	GTM-IP Testbench Users guide

Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
GTM	Generic Timer Module
PWM	Pulse Width Modulation
IFS	Interface Simulator

Table of Contents

1 Introduction

1.1 Overview

The Generic Timer Module reference model (GTM-RM) is SystemC based simulation model that is meant for verification, evaluation and demonstration purposes of the GTM functionality and behaviour. The GTM-RM is embedded into a simulation environment called IFS (Interface Simulator), which provides a configuration and status check interface (CPU abstraction level) as well as possibilities for measuring and visualizing of input and output signals. The activities of the IFS are specified in a so called IFS script using an easy to learn formal language. Moreover, the IFS environment also provides a software interface which can be used for software development in early stages of the design process where no real hardware is available.

The purpose of this document is to describe the usage of the GTM-RM within the IFS based simulation environment. A detailed functional description of the GTM-RM is described in the GTM specification [GTMSPEC]. A more detailed description of the IFS system can be found in [IFSDOC] and the test bench guide [TBDOC].

2 Reference Model Configuration

2.1 GTM-RM Device Configuration

2.1.1 GTM-RM Default Device Configuration

In the GTM-RM v1.5.5-01 the theoretical maximum number of GTM submodule instances is realized. The GTM-IP 104 configuration is a subset of the GTM-RM v1.5.5-01 configuration.
The following table lists the number of instances available in GTM-RM v1.5.5-01.

	Instances
BRC	1
PSM	2
TIM	7
TOM	6
ATOM	12
DTM	48
MCS	7
DPLL	1
SPE	4
CMP	1
MON	1

2.1.2 GTM-RM Command Line Configuration

By default the GTM-RM behaves like GTM-IP device 104. This means e.g. that the DPLL RAM2 size and also the DPLL PMT action number is limited like specified for device 104.

2.2 GTM-RM Configuration for Simulation

2.2.1 TRIGGER_DLY register

Address 280F0h
The value written at address 280F0h defines a delay of DPLL trigger input signal. The unit is 1 ns.

2.2.2 STATE_DLY register

Address 280F4h
The value written at address 280F4h defines a delay of DPLL state input signal. The unit is 1 ns.

2.3 GTM_REV Register

The reset value of the register **GTM_REV** is 0x10015501

2.4 Sub Module Base Addresses

The following Table 2.1 lists all sub modules of the GTM-RM v1.5.5-01 with its corresponding base addresses.

Table 2.1: Sub Module Base Addresses

Submodule	**Base address**
GTM Global	0x00000000
BRIDGE	0x00000030
MAP	0x00000F00
MCFG	0x00000F40
TBU	0x00000100
MON	0x00000180
CMP	0x00000200
ARU	0x00000280
CMU	0x00000300
BRC	0x00000400
ICM	0x00000600
SPE0	0x00000800
SPE1	0x00000880
SPE2	0x00000900
SPE3	0x00000980
TIM0	0x00001000
TIM1	0x00001800
TIM2	0x00002000
TIM3	0x00002800
TIM4	0x00003000
TIM5	0x00003800
TIM6	0x00004000
TOM0	0x00008000
TOM1	0x00008800
TOM2	0x00009000
TOM3	0x00009800
TOM4	0x0000A000
TOM5	0x0000A800
ATOM0	0x0000D000
ATOM1	0x0000D800
ATOM2	0x0000E000
ATOM3	0x0000E800
ATOM4	0x0000F000
ATOM5	0x0000F800

ATOM6	0x00010000
ATOM7	0x00010800
ATOM8	0x00011000
ATOM9	0x00011800
ATOM10	0x00012000
ATOM11	0x00012800
DTM0	0x00013000
DTM1	0x00013040
DTM2	0x00013080
DTM3	0x000130C0
DTM4	0x00013100
DTM5	0x00013140
DTM6	0x00013180
DTM7	0x000131C0
DTM8	0x00013200
DTM9	0x00013240
DTM10	0x00013280
DTM11	0x000132C0
DTM12	0x00013300
DTM13	0x00013340
DTM14	0x00013380
DTM15	0x000133C0
DTM16	0x00013400
DTM17	0x00013440
DTM18	0x00013480
DTM19	0x000134C0
DTM20	0x00013500
DTM21	0x00013540
DTM22	0x00013580
DTM23	0x000135C0
DTM24	0x00013600
DTM25	0x00013640
DTM26	0x00013680
DTM27	0x000136C0
DTM28	0x00013700
DTM29	0x00013740
DTM30	0x00013780
DTM31	0x000137C0
DTM32	0x00013800
DTM33	0x00013840
DTM34	0x00013880
DTM35	0x000138C0

DTM36	0x00013900
DTM37	0x00013940
DTM38	0x00013980
DTM39	0x000139C0
DTM40	0x00013A00
DTM41	0x00013A40
DTM42	0x00013A80
DTM43	0x00013AC0
DTM44	0x00013B00
DTM45	0x00013B40
DTM46	0x00013B80
DTM47	0x00013BC0
F2A0	0x00018000
AFD0	0x00018080
FIFO0	0x00018400
FIFO0_MEMORY	0x00019000
F2A1	0x0001C000
AFD1	0x0001C080
FIFO1	0x0001C400
FIFO1_MEMORY	0x0001D000
DPLL	0x00028000
DPLL_RAM1A	0x00028200
DPLL_RAM1BC	0x00028400
DPLL_RAM2	0x0002C000
MCS0	0x00030000
MCS0_MEMORY	0x00038000
MCS1	0x00031000
MCS1_MEMORY	0x00040000
MCS2	0x00032000
MCS2_MEMORY	0x00048000
MCS3	0x00033000
MCS3_MEMORY	0x00050000
MCS4	0x00034000
MCS4_MEMORY	0x00058000
MCS5	0x00035000
MCS5_MEMORY	0x00060000
MCS6	0x00036000
MCS6_MEMORY	0x00068000

2.5 Memory Address Ranges

The following table lists the address range of the Memories modelled in GTM-RM.

The address range is calculated

Table 2.2: Memory Address Ranges

Submodule	Memory Address Ranges
FIFO0_MEMORY	0x00019000 – 0x00019FFC
FIFO1_MEMORY	0x0001D000 – 0x0001DFFC
MCS0_MEMORY	0x00038000 – 0x000397FC
MCS1_MEMORY	0x00040000 – 0x000417FC
MCS2_MEMORY	0x00048000 – 0x000497FC
MCS3_MEMORY	0x00050000 – 0x000517FC
MCS4_MEMORY	0x00058000 – 0x000597FC
MCS5_MEMORY	0x00060000 – 0x000617FC
MCS6_MEMORY	0x00068000 – 0x000697FC
DPLL_RAM1A	0x00028200 – 0x000283FC
DPLL_RAM1B	0x00028400 – 0x000285FC
DPLL_RAM1C	0x00028600 – 0x000289FC
DPLL_RAM2	0x0002C000 – 0x0002D7FC (0x0002FFFC)

2.6 ARU Configuration

Table 2.3: ARU Write Addresses

Name	Address		Name	Address
ARU_ACCESS	0x000		**ATOM [0 .. 3]**	
TIM [0 .. 2]			ATOM0_WRADDR[0..7]	0x11F..0x126
TIM0_WRADDR[0..7]	0x001..0x008		ATOM1_WRADDR[0..7]	0x127..0x12E
TIM1_WRADDR[0..7]	0x009..0x010		ATOM2_WRADDR[0..7]	0x12F..0x136
TIM2_WRADDR[0..7]	0x011..0x018		ATOM3_WRADDR[0..7]	0x137..0x13E
TIM3_WRADDR[0..7]	0x019..0x020		ATOM4_WRADDR[0..7]	0x13F..0x146
TIM4_WRADDR[0..7]	0x021..0x028		ATOM5_WRADDR[0..7]	0x147..0x14E
TIM5_WRADDR[0..7]	0x029..0x030		ATOM6_WRADDR[0..7]	0x14F..0x156
TIM6_WRADDR[0..7]	0x031..0x038		ATOM7_WRADDR[0..7]	0x157..0x15E
misc			ATOM8_WRADDR[0..7]	0x15F..0x166
unused	0x039..0x050		ATOM9_WRADDR[0..7]	0x167..0x16E
F2A [0]			ATOM10_WRADDR[0..7]	0x16F..0x176
F2A0_WRADDR[0..7]	0x051..0x058		ATOM11_WRADDR[0..7]	0x177..0x17E
F2A1_WRADDR[0..7]	0x059..0x060		**DPLL**	
BRC			DPLL_WRADDR[0..31]	0x17F..0x19E
BRC_WRADDR[0..21]	0x061..0x076		**misc**	
MCS [0 .. 2]			unused	0x19F..0x1FD
MCS0_WRADDR[0..23]	0x077..0x08E		ARU_EMPTY_ADDR	0x1FE
MCS1_WRADDR[0..23]	0x08F..0x0A6		ARU_FULL_ADDR	0x1FF
MCS2_WRADDR[0..23]	0x0A7..0x0BE			
MCS3_WRADDR[0..23]	0x0BF..0x0D6			

MCS4_WRADDR[0..23]	0x0D7..0x0EE			
MCS5_WRADDR[0..23]	0x0EF..0x106			
MCS6_WRADDR[0..23]	0x107..0x11E			

The round trip time of ARU in GTM-RM v1.5.5-01 is 73 * SYS_CLK.
The SYS_CLK frequency can be controlled by command files,
e.g. the line
`CLK FREQ  100 MHz`
in file **`sw_tom_port_test.cmd`**
See chapter 4 and 5 for details.

3 Installation

3.1 System requirements

The GTM-RM model is delivered and tested for SystemC v2.3 reference implementation from OSCI. Compilation and simulation was done on the operating system Linux RedHat 5 and Windows 7. For Linux the gcc compiler version 4.1.1 was used. For Windows 7 the Visual C++ 2010 Express Edition was used for compilation.

The delivered executables are tested on following Hardware/Software Environment:

- Linux Distribution: RH5U8 RedHat 5.8
- Linux Hardware: x86_64
- Linux Kernel Version: 2.6.18-348.18.1.el5
- (Linux Modelsim Version: Modelsim v10.0d)

or

- Windows 7

3.2 Directory structure of deliverables

```
                                              entry point via environment
|                                             variable GTM_HOME
|
|----doc                                      documentation
|
|----design                                   GTM-IP / GTM-RM Design Data
|
|----install                                  installation files
|
|----libs                                     libraries of GTM-RM
|
|----simulation                               simulation of GTM-IP/GTM-RM
```

3.3 Environment installation

3.3.1 Installation on a Windows platform

To be able to compile a new executable (gtm_app.exe) for windows platform, the following environment variables have to be set properly in the windows control panel

GTM_HOME	This is the root path of the GTM-RM release package. The variable should point to directory `GTM_RM_104_1<xxyz>_WinApps`.
SYSTEMC_HOME	Has to point to the root directory of the OSCI SystemC reference implementation

	GTM_RM_104_1<xxyz>_WinApps/libs/systemc-2.2.0
SCV_HOME	Has to point to the root directory of the SCV library GTM_RM_104_1<xxyz>_WinApps/libs/scv-1.0p2-sysc2.2
IFS_HOME	Has to point to the root directory of the IFS test bench library GTM_RM_104_1<xxyz>_WinApps/libs/ifs-3.3
ASM_MCS_HOME	Has to point to the root directory of the assembler MCS library GTM_RM_104_1<xxyz>_WinApps/libs/asm_mcs

In peparation of a simulation on a Windows patform, the setup file setup.bat of directory `install` has to be adapted and then executed:

1. Modify the file `install/setup.bat`:
   Replace in the file the part <`path_to_GTM`> by your path to this deliverable package

2. Execute the setup file
   `> setup.bat`

3. To be able to run the simulation in batch mode
   change to directory `simulation/rm/msvc100`
   and replace in the file `simulate.bat` the part <`path_to_GTM`> by your path to this deliverable package.
   Execute the batch file by calling
   `> simulate.bat <path_to_cmd_file> <cmd_file_name>`
   (Note: <cmd_file_name> without file extension .cmd)

4 Simulation execution

4.1 IFS command file based simulation for Windows

4.1.1 Simulation of delivered test command files

Simulations controlled by ASCI-command files (`*.cmd`) should be performed in the directory `GTM_RM_104_1<xxyz>_WinApps/simulation/rm/msvc100`

In this directory the simulation is started by calling the executable file `gtm_app.exe` .
This executable can be called with the following arguments:

```
-c <name> : control command file name (e.g. –c control.cmd). If no argument –c <name> is passed, the default is a file named control.cmd.
-o <path> : VCD-trace file directory <path> (e.g. –o <path>/trace.vcd). Default is no VCD file generation. Note: if the argument –o is missing, no VCD file is written at all.
-afd      : trace additional signals of module AFD0
-atom     : trace additional signals of module ATOM0
-cmu      : trace additional signals of module CMU0
-dpll     : trace additional signals of module DPLL
-fifo     : trace additional signals of module FIFO0
-f2a      : trace additional signals of module F2A0
-mcs      : trace additional signals of module MCS0
-tbu      : trace additional signals of module TBU
-tim      : trace additional signals of module TIM0
-tom      : trace additional signals of module TOM0
```

Additionally, a seed number can be appended as an argument by adding

```
seed=<number>
          : seed number for a randomized test cases.
```

If no seed is specified, the test bench chooses one and prints the chosen value.
The output has the following style:

```
..
   --------------------------------------------
   INFO : no arguments specified…
   use following additional vsim arguments for defining a seed
   [seed=<global_seed> | seedfile=<seedfile> | makeseedfile]
   --------------------------------------------
   --------------------------------------------
   generating random global seed 531650471121000790
   --------------------------------------------
```

..
Append the printed seed number as an argument to re-simulate the randomized simulation.

Example: simulation of test case integration_tests\sw_tom_port_test.cmd

`> gtm_app.exe –c integration_tests\sw_tom_port_test.cmd`

If a command file makes usage of randomization expressions, the seed can be initialized with a value on the command line using the following argument:

    seed=<NUM>

whereas <NUM> is an arbitrary integer. If this argument is omitted a random seed is used.

The batch file `simulate.bat` can be used to simulate in batch mode. This speeds up the simulation on Windows by forwarding the output to a dedicated log file.
Please adapt the path `<path_to_GTM>` inside the script `simulate.bat` before calling.
Start the batch simulation by calling
`> simulate.bat <subsirectory> <cmd-file>`

e.g. to simulate the case integration_tests\sw_gtm_irq_test.cmd type
`> simulate.bat integration_tests sw_tom_port_test`

4.1.2 Creation of new test command files

To create your own test command file (e.g. `my_test.cmd`) consider using one of the delivered command files as a template. Save the new file in the directory
`GTM_RM_104_1<xxyz>_WinApps/simulation/rm/msvc100`
and use it as an argument.
Example: simulation of test case `my_test.cmd`
`> gtm_app.exe –c my_test.cmd`

5 C-Software via HAL

5.1.1 Simulation of delivered C-software tests

Simulations controlled by C-coded application files (`*.c`) should be performed for Windows in the directory
`GTM_RM_104_1<xxyz>_WinApps\simulation\rm\msvc100`

By choosing one of the special command files out of directory integration_tests as an argument the corresponding C-test is selected.

Example: simulation of software integration tests on Windows
```
> gtm_app.exe –c integration_tests\sw_tom_port_test.cmd
```

5.1.2 Creation of new C-software tests on a Windows platform

To create a new software application C-file please switch to directory
`GTM_RM_104_1<xxyz>_WinApps\design\gtm_sim\v1\software\integration_tests`

Perform now following steps:

1. In this directory create your own test file (`my_test.c`) with the function
   `signed char my_test()`
   being the entry point of your C-code.
2. Edit file `gtm_main_104.cpp`: add in this file your own entry
   DEFINE_TEST_FCT(my_test);
3. Create a new file `sw_my_test.cmd` similar to one of the delivered sw_<xyz>.cmd-files
   (directory
   `GTM_RM_104_1<xxyz>_WinApps\design\gtm_sim\v1\sim_data\ifscmd`):
   add in this file at least the line
   `SWI RUN_SW my_test`
4. Create new executable `gtm_app.exe`:
   a. Switch to directory
      `GTM_RM_104_1<xxyz>_WinApps\simulation\rm\msvc100`
   b. Start Visual C++ 2010 Express Edition and load the project file
      `gtm_app_v104.vcxproj`
   c. In Project Map Explorer of Visual C++: Add the file `my_test.c` to the list of source files.
   d. Set explicitly for the new file `my_test.c` the compile option 'compile as C++ code (/Tp)'

e. Build new executable `gtm_app_v104.exe` (Configuration Debug/Release)
f. Copy new executable `gtm_app_v104.exe` from sub-directory `Debug/Release` to directory `GTM_RM_104_1<xxyz>_WinApps\simulation\rm\msvc100`

The new software integration test is then simulated by executing `gtm_app.exe` with the argument `sw_my_test.cmd` in directory `GTM_RM_104_1<xxyz>_WinApps\simulation\rm\msvc100`.

Example: simulation of new software integration tests

```
> gtm_app.exe –c integration_tests\sw_my_test.cmd
```

6 Tools

6.1 Editor CREdit

To edit a simulation control file (*.cmd) and to start the simulation by calling the executable with the control file as an argument can easily be done with the editor CREdit. This tool is not part of the Release package but can be downloaded from following web page: http://www.praven3.com/credit/

In order to integrate the GTM into the editor CREdit, some adjustments have to be performed in the editor. After starting the CREdit.exe, click the menu *Options->Tools*. Click the *New* button In the *Configure Tools* dialog to add an entry into the Menu contents. Define the name *GTM* for the inserted item and click the generated item. Configure the dialog according to the settings mentioned in figure
Figure 7.1. The variable <PATH> refers the absolute path to the directory, where your file *gtm_app.exe* and the subdirectory *inc* with the delivered command files is located (e.g. `GTM_RM_104_1<xxyz>_WinApps\simulation\rm\msvc100\`). The optional command line arguments of the executable *gtm_app.exe* may be adapted to the desired needs by modifying the text field *Arguments*. Finally press the OK button.

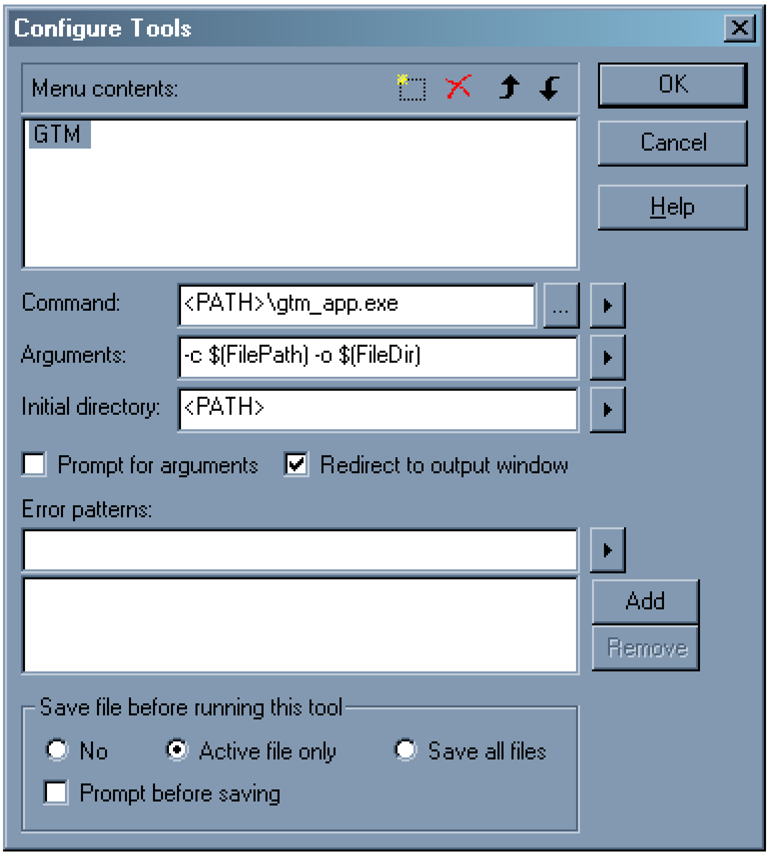

Figure 7.1: Dialog Configure Options.

In order to execute command files within the Editor, just open a command file in the editor and click the menu *Tools->GTM*. The trace.vcd file is generated in the same directory where the executed command file is located.

In order to enable command file specific syntax highlighting in the editor CREdit.exe click the menu *Options->Syntax Highlighting* to open the Syntax Highlighting Dialog. Click the button *Import* and open the delivered file *credit_syntax.ini*. In the Dialog Import of figure Figure 7.2 move the item *IFS Command File* from the list field *Source* to the list field *Target* by selecting the item and pressing the "->" button. Finally click the OK button. Now, all files with file extension *.ifs* and *.cmd*. are highlighted using the IFS Command File syntax.

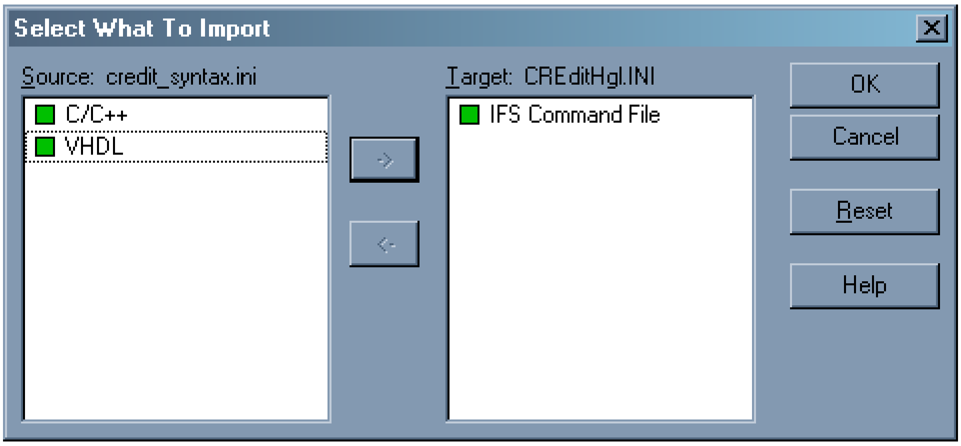

Figure 7.2: Dialog Import.

6.2 Wave-Viewer GTKWave

To view the waves stored in the file trace.vcd, the wave viewer GTKWave could be used. This tool is not part of the Release package but can be downloaded from following web page:
http://www.dspia.com/gtkwave.html

For the installation of GTKWave some steps have to be performed. Please read the following installation notes carefully:

Chapter 2.. Download the following files from the Website http://www.dspia.com/gtkwave.html and store them in a dedicated directory (e.g. c:\gtkwave_3-3)

- gtkwave.exe.gz
- lilbs.tar.gz
- tcl_libs.tar.gz (creates directories \bin and \lib

2. Unzip libs.tar.gz and tcl_libs.tar.gz
    - libs.tar.gz   -> contains some DLLs with naming lib*.dll
    - tcl_libs.tar.gz -> creates directories \bin and \lib

3) **Rename** gtkwave.exe.gz to gtkwave.exe (don't unzip it)

4) Copy the files tcl85.dll and tk85.dll from directory \bin one level higher
   (e.g. to C:\gtkwave_3-3)

5) Move complete directory \lib (with all subdirectories) one level higher
   (e.g. to C:)

6) The executable gtkwave.exe can be started from the command line together with the trace file (e.g. trace.vcd) as an argument
   (e.g. C:\gtkwave_3-3> gtkwave.exe trace.vcd)

Please see notes on CREdit above on how to start GTKWave out of this module.

Revision History

Issue	Date	Remark
0.1	10.04.2014	Initial version