Imperas Installation and Getting Started Guide

This document explains how to install and get started with
OVPsim Simulator and models and the Imperas professional products
under Linux and Windows.

Imperas Software Limited
Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	Imperas
Version:	1.4.26
Filename:	Imperas_Installation_and_Getting_Started.doc
Project:	Imperas / Open Virtual Platforms Installation and Getting Started
Last Saved:	Friday, 17 July 2015
Keywords:	Installation, Getting Started, Linux, Windows, Licensing

Table of Contents

1 Preface

This document describes how to install the Imperas Professional products, Advanced Multicore Software Development Kit (M*SDK) and Virtual Platform Development and Simulation (C*DEV, S*DEV and M*DEV), and the Open Virtual Platforms OVPsim simulator on both Windows and Linux computers. The installation includes models, documents and examples.

This document further introduces how to compile applications using the supplied example GNU cross compiler toolchains/tool kits and how to use this binary with a virtual platform.

Please see other documents that explain how to create platforms / harnesses, processor models, and peripheral / behavioral models.

2 Introduction

This installation guide is for the installation of the OVP or Imperas Professional products on a Linux or Windows host computer. We will need the installation files from of either OVPsim from www.OVPworld.org or the Imperas products from www.Imperas.com, and at least one example compiler toolchain, also from www.OVPworld.org.

OVP is provided as a single installation package ‘OVPsim’

Imperas products are provided as two installation packages ‘Imperas_SDK’ for M*SDK and ‘Imperas_DEV’ for C*DEV, S*DEV and M*DEV (configured on installation or post-install with configuration script).

In order to build native host executables and to cross compile applications under Windows, we suggest the installation of an MSYS/MinGW[1] environment www.mingw.org. See section 5 for instructions on setting up this environment. This will make compilation of Applications, Models and Platforms / Simulation harnesses simple.

Part of the installation includes examples files and these will be used to illustrate the processes and tools used to get your applications running on virtual simulation platforms. In this document, we will use the openCores openRISC OR1K as the target embedded processor.

[1] It is also possible to use a Cygwin / MinGW environment but care must be taken to ensure that the MINGW GNU toolset is used.

3 Hardware and Software Requirements

3.1 Operating System[2]

Product Build	Operating System
Linux 32-bit	Fedora Core 12
Linux 64-bit[3]	Fedora Core 18
Windows 32-bit	Windows 7 64-bit[4] Professional with SP1
Windows 64-bit[3]	Windows 7 64-bit Professional with SP1

3.2 Hardware

Supported Processors: x86 32-bit and 64-bit

Disk space requirements:

OVP or Imperas Installations:	140MB to 220MB
Toolchain Installations (each):	60MB to 300MB[5]

[2] These are the versions of the operating systems that has been fully verified and supported by Imperas, the product has been shown to operate correctly on others.
[3] This is only supported by the Imperas installations, Imperas_SDK and Imperas_DEV
[4] The Imperas 32-bit products are supported in emulation mode on 64-bit hosts
[5] The disk space indicated is per toolchain requirement. The actual disk space required is dependent upon the type and number of toolchains installed

4 Installation

The Imperas professional products and the OVPsim simulator may be installed on a Linux or Windows platform.

Section 4.1 provides specific information regarding the product packages. The details of the package installations are the same and are shown for Linux users in section 4.3, and Windows users in section 4.4.

Different major versions of the software should not be installed on top of each other. The new installation should either be made into a different directory or the old version deleted/uninstalled before the new version is added. For this reason you should avoid making changes to files in the installation directories, as those changes will be lost when the version is un-installed and a subsequent version is installed.
Different minor versions may be installed together, later minor versions typically provide updates to products.

4.1 Installation Packages

4.1.1 OVPsim

The OVPsim simulation package, ‘OVPsim’, contains

1. the OVPsim reference simulator
2. the OVP model library
3. examples and demonstrations

4.1.2 Imperas Developer

The Imperas Developer package, ‘Imperas_DEV, contains

1. a professional version of the simulator that can be licensed for different capabilities
    a. Controller (C*DEV) : single processor / multi-core processor
    b. Standard (S*DEV) : multiple processors / multi-core processors (homogeneous processor vendor)
    c. Multi (M*DEV) : multiple processors / multi-core processors (heterogeneous)
2. The Imperas Generation tool 'iGen' that increases productivity by simplifying and automating the generation of C, C++, SystemC TLM2.0 platforms files, peripheral and processor model templates

The Imperas Developer package once installed requires that the IMPERAS_PERSONALITY is set to reflect the level of the product that has been purchased. This is done during installation or after installation using the *selectSimulator* script.

4.1.3 Imperas Advanced Multicore Software Development Kit

The Imperas SDK package, 'Imperas_SDK', contains

1. a professional version of the simulator that is licensed for capability
   a. Multi : multiple processors or multi-core processors (heterogeneous)
2. The Imperas Generation tool 'iGen' that increases productivity by simplifying and automating the generation of C, C++, SystemC TLM2.0 platforms files, peripheral and processor model templates
3. The Imperas Verification, Analysis and Profiling (VAP) tools
4. The cpuGen processor model generator productivity tool

4.2 Download the Installation files

First you must download the installation files from either OVPworld.org or Imperas.com.

4.2.1 Login

The OVPworld download page can only be accessed when you are registered and logged into the sites forums. Please register (can take up to 24 hours to receive authorization email) and be logged in from the forums page before continuing.

Imperas customers should go to Imperas.com and select user login. This accepts the same username and password as the OVPworld.org site. This will take you to the Imperas User Home Page. Click on Imperas Product Downloads (either Beta or current, depending on what version you wish to download) and you will be requested to enter your Imperas user name and password, which should have been provided to you by Imperas.

4.2.2 Download the appropriate package

To download OVPsim after logging in to the OVPworld website go to the downloads page http://www.ovpworld.org/download.php, look on the right hand side list and select the appropriate link for the version you wish to download (Windows or Linux). This will download the Windows or Linux installer executable for OVPsim.

To download the Imperas professional tools or the Imperas development tools after logging into the Imperas Product Downloads area select the product you wish to download (Windows or Linux).

4.3 Installing Under Linux

This section tells you how to install the Linux version of the products. Windows users should refer to section 4.4.

The Linux versions are provided as pre-built binaries in a self extracting executable file, either the Imperas SDK as ***Imperas_SDK.<version number>.<dot***

release>.Linux32*[6]*.exe, the Imperas Developer as ***Imperas_DEV.<version number>.<dot release>. Linux 32.exe*** or OVPsim as ***OVPsim.<version number>.<dot release>. Linux 32.exe*** or ***OVPsim. Linux.exe (for current version from www.OVPworld.org)***

4.3.1 Installing

Execute the self extracting executable file to install. You may need to change the installer to be executable first:

```
> chmod +x Imperas_SDK.<major version>.<minor version>.Linux32.exe
```

The self extracting executable will install the files in the current directory or in a selected directory:

Then execute the self extracting installer to extract the files:

```
> ./Imperas_SDK.<major version>.<minor version>.Linux32.exe
```

Or

```
> ./OVPsim.<major version>.<minor version>.Linux32.exe
```

You will be asked to accept a license agreement. It should be read and, if acceptable, agreed to by typing yes at the prompt. For example:

```
Software License Agreement for Imperas Proprietary Software
Imperas Limited.
---------------------

<.. lines removed ..>

You are not allowed to use this software without a specific signed license
agreement.

Imperas_Software_License_Note (v.1.0)


Please indicate whether you accept the license agreement
yes/no >
```

Once the license agreement has been accepted the Imperas tools or OVPsim you will be prompted for the directory to install into:

```
Install into directory <current directory>/Imperas.<release major version>
yes/no >
```

[6] This shows the release for a Linux 32-bit host, Linux 64-bit hosts are supported and available from Imperas.

If you reply yes, the installation will carry on into the current directory. If you reply no, you will be prompted for a directory to install into:

```
Please provide an installation directory:
```

This will create `<installDir>/Imperas.<release major version> (for example /usr/Imperas.20130630)`, where `<installDir>` is the directory that is entered at the prompt. Since the tools are installed in a directory tree whose root contains the version, multiple versions may coexist side by side, or the old version's directory tree may be deleted.

The Linux binary executables and shared objects are provided in the directory:
`<installDir>/Imperas.<major version>/bin/Linux32`

The model libraries are provided, as shared objects, in the directory:
`<installDir>/Imperas.<major version>.<minor version>/lib/Linux32/ImperasLib`

4.3.2 Setting up the Linux Environment

4.3.2.1 Setup Script

A script is provided that provides a function to setup the environment in which to run the Imperas tools. The script is provided in the *bin* directory below the Imperas installation directory. This script should be sourced in a bourne shell to provide the function ***setupImperas***, which is then executed passing the path to the Imperas directory as the argument.

```
> source <installDir>/Imperas.<major version>/bin/setup.sh
> setupImperas <installDir>/Imperas.<major version>
```

One of the environment variables set is IMPERAS_RUNTIME which is used to select the simulator to be loaded, either ***CpuManager*** for the Imperas Professional simulator or ***OVPsim*** for the OVP simulator.
This environment variable may be set prior to running this function. If not set the *setupImperas* function will set the environment variable in accordance with the installed files i.e. if the Imperas simulator, ***CpuManager***, is available it will set the IMPERAS_RUNTIME to CpuManager; if not found it will be set to the OVPsim simulator, ***OVPsim***.

4.3.2.1.1 32-bit Product on 64-bit Host

The setup script also allows for the setup of a 32-bit product on a 64-bit host. This does require that the 64-bit host includes 32-bit compatibility libraries.

```
> source <installDir>/Imperas.<major version>/bin/setup.sh
> setupImperas <installDir>/Imperas.<major version> -m32
```

4.3.2.2 Environment

The following environment variables are required for the OVP simulator or Imperas professional tools:

Required Environment Variables

`IMPERAS_HOME`	Points to the root of the tools installation
`IMPERAS_UNAME`	Is set to the Host OS type, ***Linux***
`IMPERAS_ARCH`	Is set to the Host architecture, ***Linux32***
`IMPERAS_SHRSUF`	Is set to the suffix for shared libraries, ***so***
`IMPERAS_VLNV`	Points to the root of the compiled library in the Imperas installation
`IMPERAS_RUNTIME`	Specifies which simulator, Imperas (CpuManager) or OVPsim, to load at runtime

The **`PATH`** should include **`$IMPERAS_HOME/bin/$IMPERAS_ARCH`**

The **`LD_LIBRARY_PATH`** should include **`$IMPERAS_HOME/bin/$IMPERAS_ARCH`** and **`$IMPERAS_HOME/lib/$IMPERAS_ARCH/External/lib`**

Additional Environment Variables

`IMPERAS_PERSONALITY`	If this is set it can modify the features available from the Imperas simulator runtime. This also modifies the license features that will be required.
`IMPERAS_ISS`	If this is set it will select the Imperas Instruction Set Simulation executable version.

The following must be set manually, if required:

`IMPERAS_PROXY_SERVER`	Only required for Demo and Web licenses, if the internet is accessed through a proxy server, this should be set to the value <hostname>:<port>, e.g. 'myproxyserver.com:3128'

The following show example commands to set these variables.

Create an environment variable **`IMPERAS_HOME`** pointing to the root of the Imperas tree, for example:

```
> export IMPERAS_HOME=<installDir>/Imperas.20130630
```

Create an environment variable **`IMPERAS_ARCH`** for the current architecture, for example:

```
> export IMPERAS_ARCH=Linux32
```

Create an environment variable **`IMPERAS_SHRSUF`** for the current architecture, for example:

```
> export IMPERAS_SHRSUF=so
```

Create an environment variable **`IMPERAS_VLNV`** pointing to the root of the Imperas library, for example:

```
> export IMPERAS_VLNV=$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib
```

Create an environment variable **`IMPERAS_RUNTIME`** specifying which library to load at runtime. If you have a license for the Imperas tools this should be:

```
> export IMPERAS_RUNTIME=CpuManager
```

If you have a license for OVPsim then this may be left unset (as this is the default value) or set to:

```
> export IMPERAS_RUNTIME=OVPsim
```

Add the Imperas executables directory to the search path, for example:

```
> export PATH=${PATH}:$IMPERAS_HOME/bin/$IMPERAS_ARCH
```

Add Imperas shared library directories to the variable **`LD_LIBRARY_PATH`**, for example[7]:

```
> export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:$IMPERAS_HOME/bin/$IMPERAS_ARCH:\
$IMPERAS_HOME/lib/$IMPERAS_ARCH/External/lib
```

These commands should be added to a shell start up script, e.g. `~/.bashrc`, or must be executed some other way before the tools are used.

If you are installing one of the Developer products (C*DEV, S*DEV or M*DEV) from Imperas you will need to setup the environment variable **`IMPERAS_PERSONALITY`** for the correct type. This can be achieved by directly using the script ***IMPERAS_HOME/bin/selectSimulator.sh***.

Next, you must also set up licensing for the program. This is described for both Linux and Windows installations in section 4.5.

4.3.2.3 Internet Access Via a Proxy Server

If you are using a Demo license downloaded from the OVP website, this will require web access in order to run the simulator.
If this is the case and you use an Internet Proxy Server to access the web, you must set the environment variable, IMPERAS_PROXY_SERVER, in order to enable access to the internet.

For example if the Proxy Server is running on myproxyserver.com at port 3128

```
> export IMPERAS_PROXY_SERVER="http://myproxyserver.com:3128"
```

[7] This is all one command; note the use of the Linux line continuation character '\' at the end of the first 2 lines

A simple check to ensure this has worked correctly is as follows

```
> export IMPERAS_PROXY_SERVER="http://myproxyserver.com:3128"
> http_proxy=${IMPERAS_PROXY_SERVER wget http://www.ovpworld.org
```

This should cause the index.html to download if all is successful

4.4 Installing Under Windows

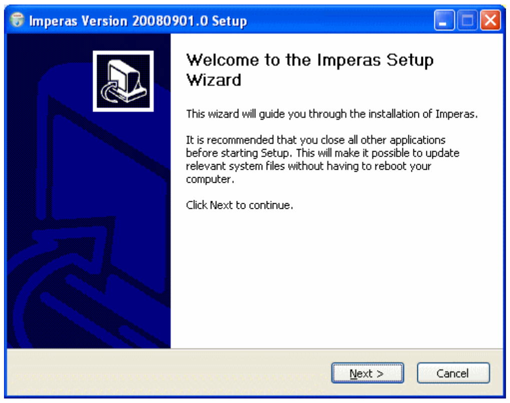
This section tells you how to install the Windows version of the tools. Linux users should refer to section 4.3.

The Windows versions are provided as executable installers, either the Imperas SDK as ***Imperas_SDK.<version number>.<dot release>.Windows32[8].exe***, the Imperas Developer as ***Imperas_DEV.<version number>.<dot release>.Windows32.exe*** or OVPsim as ***OVPsim.<version number>.<dot release>. Windows32.exe*** or ***OVPsim.Windows.exe (for current version from www.OVPworld.org)***

The installer, when executed, will extract the tools and setup the environment. Remember to either uninstall any existing versions before installing or to select a different directory to install to. An uninstall program, `uninstall.exe` is provided in the root install directory (i.e. <installDir>\uninstall.exe, e.g. `C:\Imperas\uninstall.exe`)

4.4.1 Installing

1. Execute the provided installer



Imperas Version 20080901.0 Setup

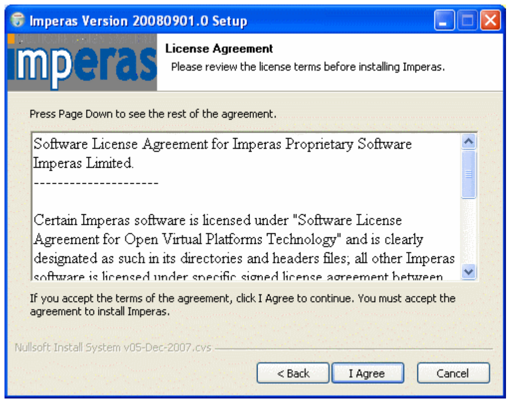
Welcome to the Imperas Setup Wizard

This wizard will guide you through the installation of Imperas.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

2. Read and click through the license agreement



Imperas Version 20080901.0 Setup

License Agreement

Please review the license terms before installing Imperas.

Press Page Down to see the rest of the agreement.

Software License Agreement for Imperas Proprietary Software
Imperas Limited.

Certain Imperas software is licensed under "Software License Agreement for Open Virtual Platforms Technology" and is clearly designated as such in its directories and headers files; all other Imperas software is licensed under specific signed license agreement between

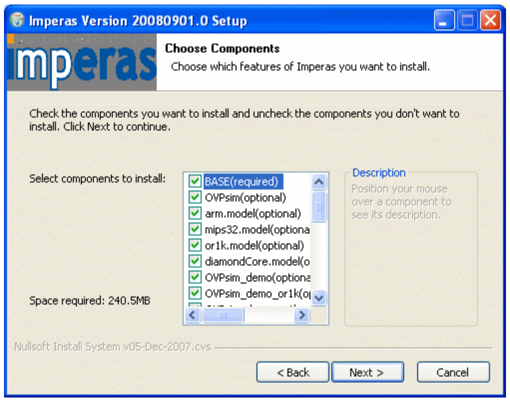
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Imperas.

Nullsoft Install System v05-Dec-2007.cvs

3. Unselect any options that are not required

4. Choose the location for installation

[8] This shows the release for a Windows 32-bit host, other Windows hosts are supported by Imperas.


Imperas Version 20080901.0 Setup

Choose Components
Choose which features of Imperas you want to install.

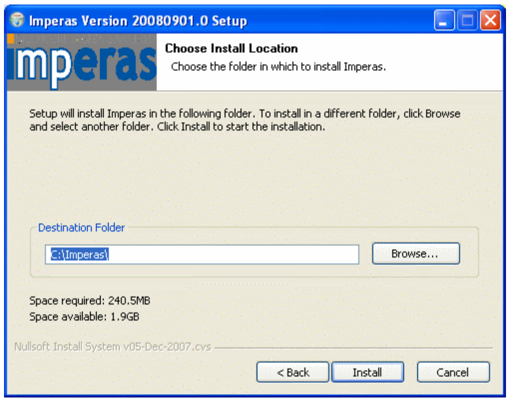
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- BASE(required)
- OVPsim(optional)
- arm.model(optional)
- mips32.model(optiona
- or1k.model(optional)
- diamondCore.model(o
- OVPsim_demo(optiona
- OVPsim_demo_or1k(o

Description
Position your mouse over a component to see its description.

Space required: 240.5MB

Nullsoft Install System v05-Dec-2007.cvs


Imperas Version 20080901.0 Setup

Choose Install Location
Choose the folder in which to install Imperas.

Setup will install Imperas in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

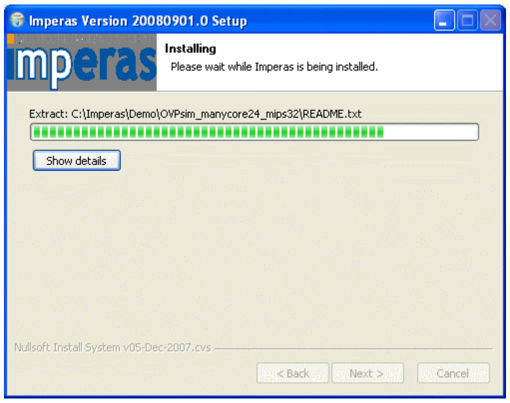
Destination Folder
C:\Imperas\

Space required: 240.5MB
Space available: 1.9GB

Nullsoft Install System v05-Dec-2007.cvs

5. Start the installation

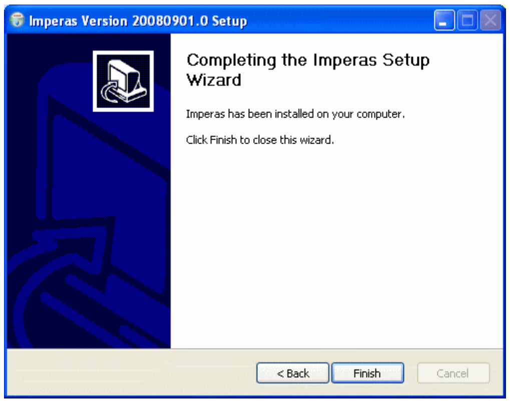
6. When completed click Finish


Imperas Version 20080901.0 Setup

Installing
Please wait while Imperas is being installed.

Extract: C:\Imperas\Demo\OVPsim_manycore24_mips32\README.txt

Nullsoft Install System v05-Dec-2007.cvs

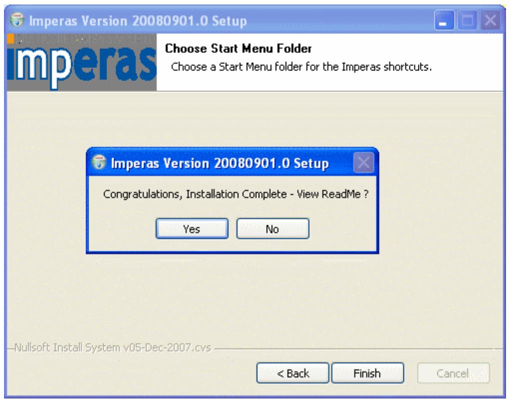

Imperas Version 20080901.0 Setup

Completing the Imperas Setup Wizard

Imperas has been installed on your computer.

Click Finish to close this wizard.

7. Finally, read the notes for this release.


Imperas Version 20080901.0 Setup

Choose Start Menu Folder
Choose a Start Menu folder for the Imperas shortcuts.

Imperas Version 20080901.0 Setup
Congratulations, Installation Complete - View ReadMe ?

Nullsoft Install System v05-Dec-2007.cvs

.

4.4.2 Setting up the Windows Environment

The following environment variables are automatically set by the installer:

Required Environment Variables

`IMPERAS_HOME`	Points to the root of the installation
`IMPERAS_VLNV`	Points to the compiled library
`IMPERAS_UNAME`	Set to the Host OS Type, ***Windows***
`IMPERAS_ARCH`	Set to the Host architecture, ***Windows32***
`IMPERAS_SHRSUF`	Is set to the suffix for shared libraries, ***dll***
`IMPERAS_RUNTIME`	Specifies which simulator, Imperas (CpuManager) or OVPsim, to load at runtime
`PATH`	modified to include ***`IMPERAS_HOME/bin/IMPERAS_ARCH`***
`LD_LIBRARY_PATH`	modified to include **`IMPERAS_HOME/bin/IMPERAS_ARCH`** and **`IMPERAS_HOME/lib/IMPERAS_ARCH/External/lib`**

Additional Environment Variables

`IMPERAS_PERSONALITY`	If this is set it can modify the features available from the Imperas simulator runtime. This also modifies the license features that will be required.
`IMPERAS_ISS`	If this is set it will select the Imperas Instruction Set Simulation executable version.

The following must be set manually, if required:

`IMPERAS_PROXY_SERVER`	Only required for Demo and Web licenses, this should be set to the value <hostname>:<port>, e.g. 'myproxyserver.com:3128'

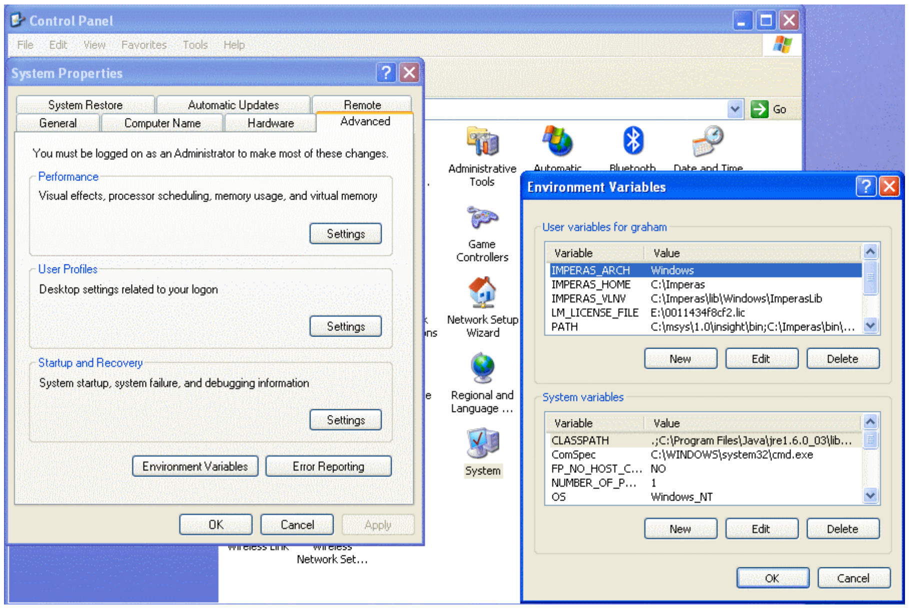
The environment variable **`IMPERAS_RUNTIME`** may be set. If the Imperas tools are installed then it will be set to **`CpuManager`** to select the Imperas simulator library to be loaded at runtime. For OVPsim this variable will be set to OVPsim. It may be left unset and the default value **`OVPsim`** will be used.

If you are installing one of the Developer products (C*DEV, S*DEV or M*DEV) from Imperas you will be asked during installation which product that you have had licenses provided for. This will set the environment variable **`IMPERAS_PERSONALITY`** for the correct product personality. This can be also be achieved after installation by directly using the script ***`IMPERAS_HOME`/bin/selectSimulator.bat*** or using the 'select product' link provided on the Program Start menu.

You must also set up licensing for the program. This is described for both Linux and Windows installations in section 4.5.

Environment variables on Windows XP can be modified by going to:

```
Start->Control Panel->System->Advanced->Environment Variables
```



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

User variables for graham

Variable	Value
IMPERAS_ARCH	Windows
IMPERAS_HOME	C:\Imperas
IMPERAS_VLNV	C:\Imperas\lib\Windows\ImperasLib
LM_LICENSE_FILE	E:\0011434f8cf2.lic
PATH	C:\msys\1.0\insight\bin;C:\Imperas\bin\...

System variables

Variable	Value
CLASSPATH	.;C:\Program Files\Java\jre1.6.0_03\lib...
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	1
OS	Windows_NT

4.5 Setting up Licensing (Both Windows and Linux)

The OVP and Imperas products are licensed using FLEXlm. In order to execute the tools you will need a license file. A license file is bound to an individual computer through that computer's host ID. To obtain a license file you must provide the host ID and host name of one of your computers that will host the license.

To determine the host ID of a computer use the `lmhostid` command of the `lmutil` utility program (`lmutil` on Linux or `lmutil.exe` on Windows). If you have already gone through the installation process then the `lmutil` utility will be in the directory `IMPERAS_HOME/bin/IMPERAS_ARCH` which will already be on your executable path.

From a Linux shell or a Windows Command Prompt/MSYS window do the following:

```
> lmutil lmhostid
lmhostid - Copyright (c) 1989-2006 Macrovision Europe Ltd. and/or Macrovision
Corporation. All Rights Reserved.
The FLEXlm host ID of this machine is "002486571114"
>
```

NOTE

If the returned host ID is "000000000000" then this may be because you are using a modern OS which uses Consistent Network Device Naming. Please see appendix E.5 for ways in which this problem can be resolved.

The host name of the computer can be obtained with the `hostname` command. From a Linux shell or a Windows Command Prompt/MSYS window do the following:

```
> hostname
Workstation123
>
```

Floating licenses require that a license server is executed on a machine that will be used to serve license to clients. The license server, lmgrd (lmgrd.exe on Windows) is provided in the binary directory of an installation.

4.5.1 OVPsim FLEXlm License keys

OVPsim FLEXlm license keys may be requested directly from http://www.ovpworld.org/licensekey.php by registered OVP users. Licenses for commercial use may be obtained from Imperas.
OVPsim licenses are uncounted node-locked licenses and as such do not require a license server or a license daemon.

OVPsim looks, by default, for its license file in:

```
$IMPERAS_HOME/OVPsim.lic
```

If the license file is installed there then there is no need to do anything else. If it is installed elsewhere then the ***`IMPERASD_LICENSE_FILE`*** environment variable[9] should be set to point to it.

OVPsim license key files are to a specific version (release) of OVPsim, with a license feature entry similar to that shown below.

```
FEATURE IMP_OVPSIM_20150205 imperasd 1.0 1-jul-2015 uncounted
HOSTID=... SIGN="...
```

You will need to obtain a new license file when you install a new version of the simulator. Since it is tied to the release, it is OK to place it in the release directory. License key files that are not tied to a specific release should not be placed in the install directory, as that directory will be replaced when a new release is installed.

When using an OVPsim license the product will also require access to the internet to allow checking for product updates.

[9] For information on setting the environment variable on Windows please see appendix

4.5.2 Imperas Licenses

The license to run the Imperas products (or OVPsim in some circumstances) is provided as a floating license where one computer on the network is the license server and the same or other computers on the network check out licenses from it.

The ***IMPERASD_LICENSE_FILE*** environment variable should be set to point to the license server.

4.5.3 Using Floating Licenses

Licenses for the Imperas tools generally will require a license daemon to be running on a license server. Instructions for getting a license daemon running on a license server can be found in APPENDIX D, Additional FLEXlm Licensing Information.

Once you have your license server running, the computer from which the tools are run needs the `IMPERASD_LICENSE_FILE` environment variable to be set to an '@' followed by the host name of the license server, for example:

```
IMPERASD_LICENSE_FILE=@serverhost
```

where **serverhost** is the host name of the license server. (See the respective installation instructions for Windows and Linux for examples of how to set an environment variable.) In this example we assume that the default port (27000-27009) is being used by the license server. If the license server was set up to use a different port for some reason it must be specified before the '@', for example:

```
IMPERASD_LICENSE_FILE=27000@serverhost
```

Here, **27000** is the port number that the license server is using. Consult your license administrator or see APPENDIX D, Additional FLEXlm Licensing Information, for more information.

4.5.4 Using Node Locked License

A license may be for a single node locked machine and as such held in a local file. In this case the same environment variable can be used but locates the file directly, for example

```
IMPERASD_LICENSE_FILE=/home/user/Imperas/Imperas.lic
```

4.5.5 License Queuing

The licensing software supports license queuing for server based license daemons, this allows waiting for licenses becoming available which is useful for batch oriented execution. In order to enable queuing, define the environment variable as shown

```
IMPERAS_QUEUE_LICENSE=1
```

5 Windows Development Environment

5.1 Introduction

The development of platforms, processor and peripheral models on the Windows operating system has been validated in an environment using MSYS and MINGW. A default build environment is provided with both the Imperas tools and OVPsim installations that will allow models and platforms to be built in this environment.

Linux users should skip ahead to Section 6.

5.2 MSYS / MinGW Environment

The MSYS / MinGW environment is a shell and GNU toolchain to use under Windows.

This should be installed on a Windows 32-bit or a Windows 64-bit host machine.

It is recommended that the latest version of MSYS be obtained from the mingw website and the MinGW cross compilers taken from the version saved to the OVP World website resources page.

5.2.1 Obtaining Latest MSYS

5.2.1.1 First, lets download

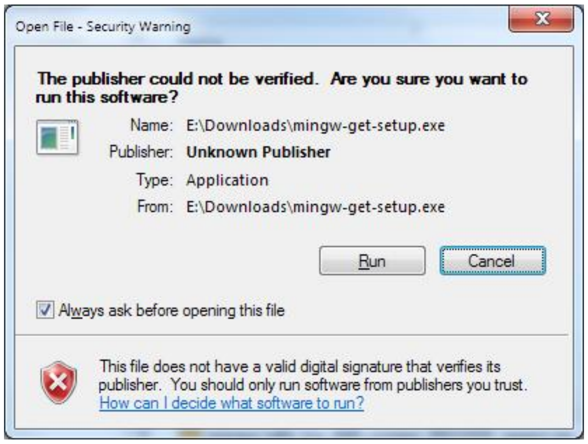
MSYS is provided as part of the MinGW installation provided by http://www.mingw.org/ and available to download from http://sourceforge.net/projects/mingw/files/MinGW/

Click on the link, **Download mingw-get-setup.exe (86.5 kB)**, after the text ‘looking for the latest version?’ to start the download of the installer.

This will download a small executable which we will then use to download and install the MSYS installation.

5.2.1.2 Download and Installation of MSYS

Execute the program that you downloaded.



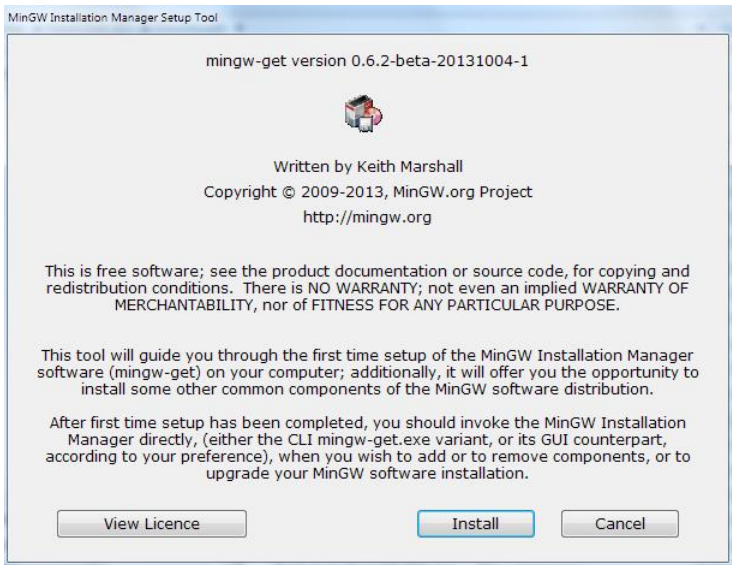
The publisher could not be verified. Are you sure you want to run this software?

Name: E:\Downloads\mingw-get-setup.exe
Publisher: **Unknown Publisher**
Type: Application
From: E:\Downloads\mingw-get-setup.exe

☑ Always ask before opening this file

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust.
How can I decide what software to run?

Select Run to execute the download and installation program



mingw-get version 0.6.2-beta-20131004-1

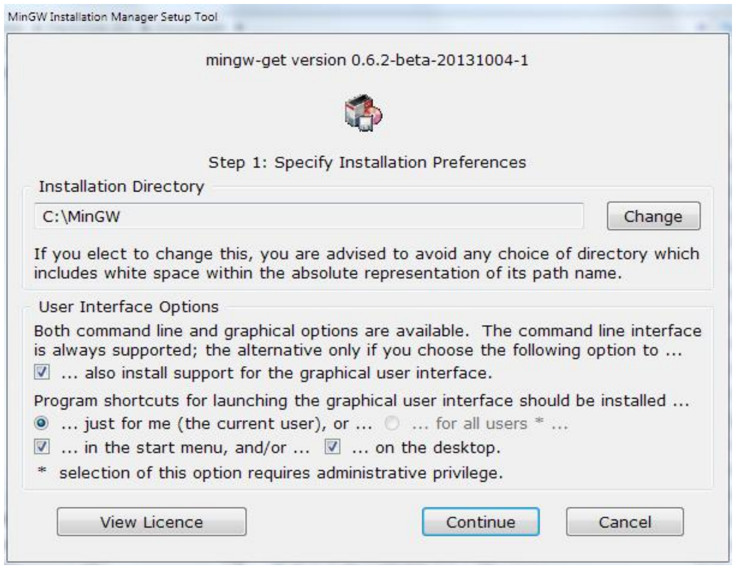
Written by Keith Marshall
Copyright © 2009-2013, MinGW.org Project
http://mingw.org

This is free software; see the product documentation or source code, for copying and redistribution conditions. There is NO WARRANTY; not even an implied WARRANTY OF MERCHANTABILITY, nor of FITNESS FOR ANY PARTICULAR PURPOSE.

This tool will guide you through the first time setup of the MinGW Installation Manager software (mingw-get) on your computer; additionally, it will offer you the opportunity to install some other common components of the MinGW software distribution.

After first time setup has been completed, you should invoke the MinGW Installation Manager directly, (either the CLI mingw-get.exe variant, or its GUI counterpart, according to your preference), when you wish to add or to remove components, or to upgrade your MinGW software installation.

View the License and if acceptable, click Install



MinGW Installation Manager Setup Tool

mingw-get version 0.6.2-beta-20131004-1

Step 1: Specify Installation Preferences

Installation Directory

C:\MinGW

If you elect to change this, you are advised to avoid any choice of directory which includes white space within the absolute representation of its path name.

User Interface Options

Both command line and graphical options are available. The command line interface is always supported; the alternative only if you choose the following option to …

☑ … also install support for the graphical user interface.

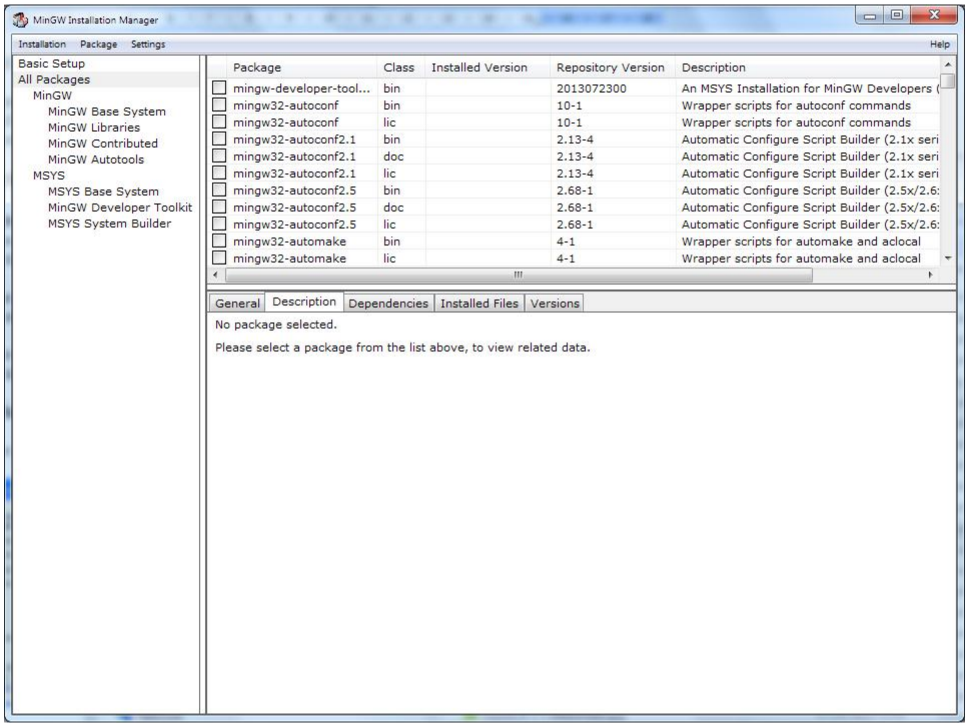
Program shortcuts for launching the graphical user interface should be installed …

● … just for me (the current user), or … ○ … for all users * …

☑ … in the start menu, and/or … ☑ … on the desktop.

* selection of this option requires administrative privilege.

Select the options and installation root directory, click Continue.

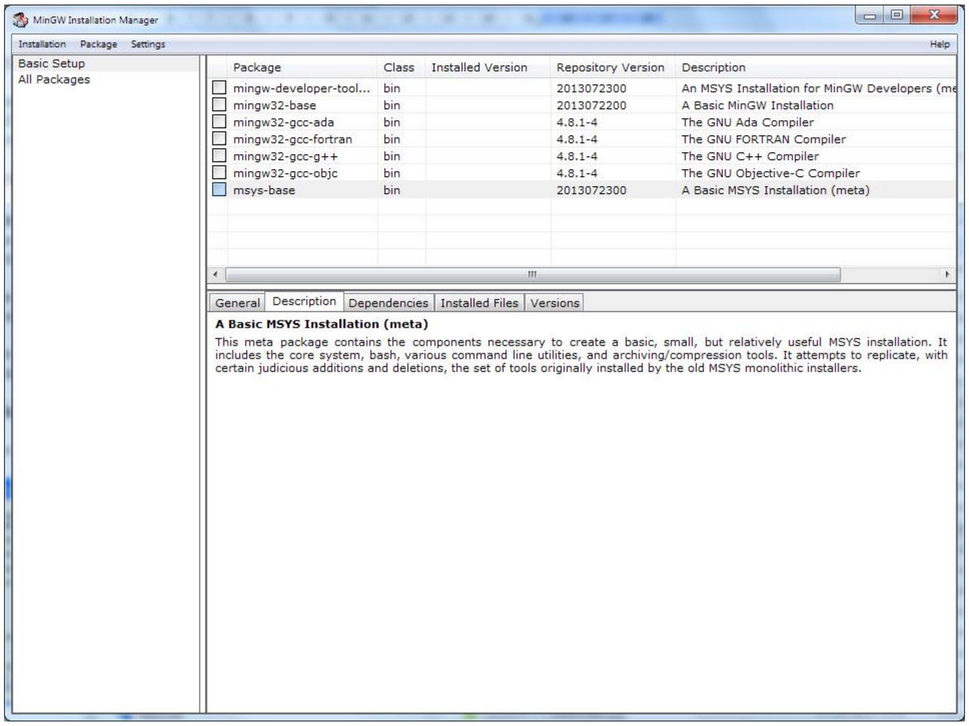


Package	Class	Installed Version	Repository Version	Description
mingw-developer-tool…	bin		2013072300	An MSYS Installation for MinGW Developers (
mingw32-autoconf	bin		10-1	Wrapper scripts for autoconf commands
mingw32-autoconf	lic		10-1	Wrapper scripts for autoconf commands
mingw32-autoconf2.1	bin		2.13-4	Automatic Configure Script Builder (2.1x seri
mingw32-autoconf2.1	doc		2.13-4	Automatic Configure Script Builder (2.1x seri
mingw32-autoconf2.1	lic		2.13-4	Automatic Configure Script Builder (2.1x seri
mingw32-autoconf2.5	bin		2.68-1	Automatic Configure Script Builder (2.5x/2.6:
mingw32-autoconf2.5	doc		2.68-1	Automatic Configure Script Builder (2.5x/2.6:
mingw32-autoconf2.5	lic		2.68-1	Automatic Configure Script Builder (2.5x/2.6:
mingw32-automake	bin		4-1	Wrapper scripts for automake and aclocal
mingw32-automake	lic		4-1	Wrapper scripts for automake and aclocal

No package selected.

Please select a package from the list above, to view related data.

The installer can install all of the MSYS and MinGW components, we want to install only the MSYS Base System. Select the "Basic Setup" option and select 'msys-base' for installation.

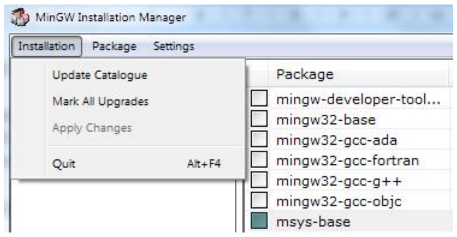


	Package	Class	Installed Version	Repository Version	Description
☐	mingw-developer-tool...	bin		2013072300	An MSYS Installation for MinGW Developers (me
☐	mingw32-base	bin		2013072200	A Basic MinGW Installation
☐	mingw32-gcc-ada	bin		4.8.1-4	The GNU Ada Compiler
☐	mingw32-gcc-fortran	bin		4.8.1-4	The GNU FORTRAN Compiler
☐	mingw32-gcc-g++	bin		4.8.1-4	The GNU C++ Compiler
☐	mingw32-gcc-objc	bin		4.8.1-4	The GNU Objective-C Compiler
☐	msys-base	bin		2013072300	A Basic MSYS Installation (meta)

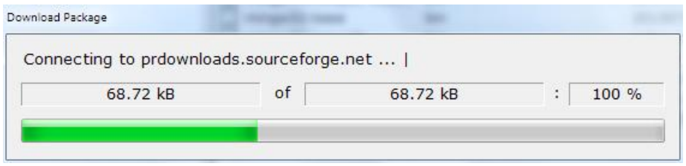
A Basic MSYS Installation (meta)

This meta package contains the components necessary to create a basic, small, but relatively useful MSYS installation. It includes the core system, bash, various command line utilities, and archiving/compression tools. It attempts to replicate, with certain judicious additions and deletions, the set of tools originally installed by the old MSYS monolithic installers.

Under the Installation menu select 'Apply Changes'.

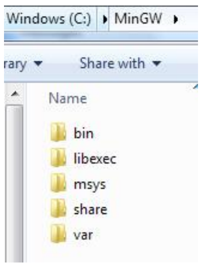


This will start the download and installation of the MSYS base system



```
Connecting to prdownloads.sourceforge.net ... |
68.72 kB   of   68.72 kB   :   100 %
```

When the installation has completed you will have a MinGW directory containing the basic MSYS installation:



5.2.2 Installation of MinGW 32-bit and 64-bit Toolchains

In the previous section we installed the latest MSYS environment, next we need to install the specific MinGW toolchains[10]. The version that has been verified with the Imperas/OVP build system may be obtained from the OVP World website (problems have been reported using later versions).

The files may be found on the OVPworld.org website under the Resources tab. There are two zip files containing the 32-bit and 64-bit toolchains respectively.

mingw-w32-bin_i686-mingw_20111031_sezero.zip

mingw-w64-bin_i686-mingw_20111031_sezero.zip

Download the appropriate one for your host (32-bit or 64-bit) and we recommend extracting into a directory, for example C:\MinGW_Toolchains, separate to the root of the MSYS/MinGW installation in the previous section, C:\MinGW. This will create mingw32 and mingw64 directories respectively.

5.2.2.1 Adding MinGW to the PATH

Select the Windows control panel and get to the advanced system setting and the environment setting within it.

To the PATH variable add
1. The MSYS installation binary directory
   C:\MinGW\msys\1.0\bin
2. The MinGW installation binary directory or directories.
   C:\MinGW_Toolchains\mingw32\bin
   C:\MinGW_Toolchains\mingw64\bin

[10] These versions of the toolchains are used within Imperas and so are know to work with the provided build environment.

5.2.2.2 MinGW 'make'

It is important that the correct version of make is used when using Makefiles in the Windows build environment.

The Windows bat files supplied by Imperas all invoke 'mingw32-make'

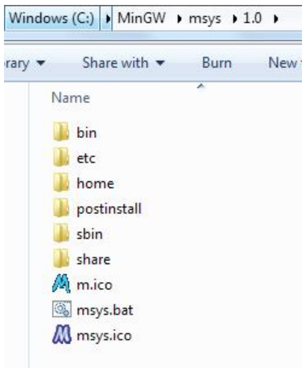
In the MSYS shell it is possible to invoke make, using one of the shell scripts supplied by Imperas in the Demo or Example directories or directly but this may not use the correct version. By default the Cross Compiler toolchains provide 'gmake.exe' and ' mingw32-make', they do not provide 'make.exe' so this is often either not found or the incorrect version found.

We recommend copying 'gmake.exe' to 'make.exe' in the Cross Compiler binary directories, for example in the directories C:\MinGW_Toolchains\mingw32\bin and C:\MinGW_Toolchains \mingw64\bin

5.2.2.3 Completion and test of MSYS/MinGW installation

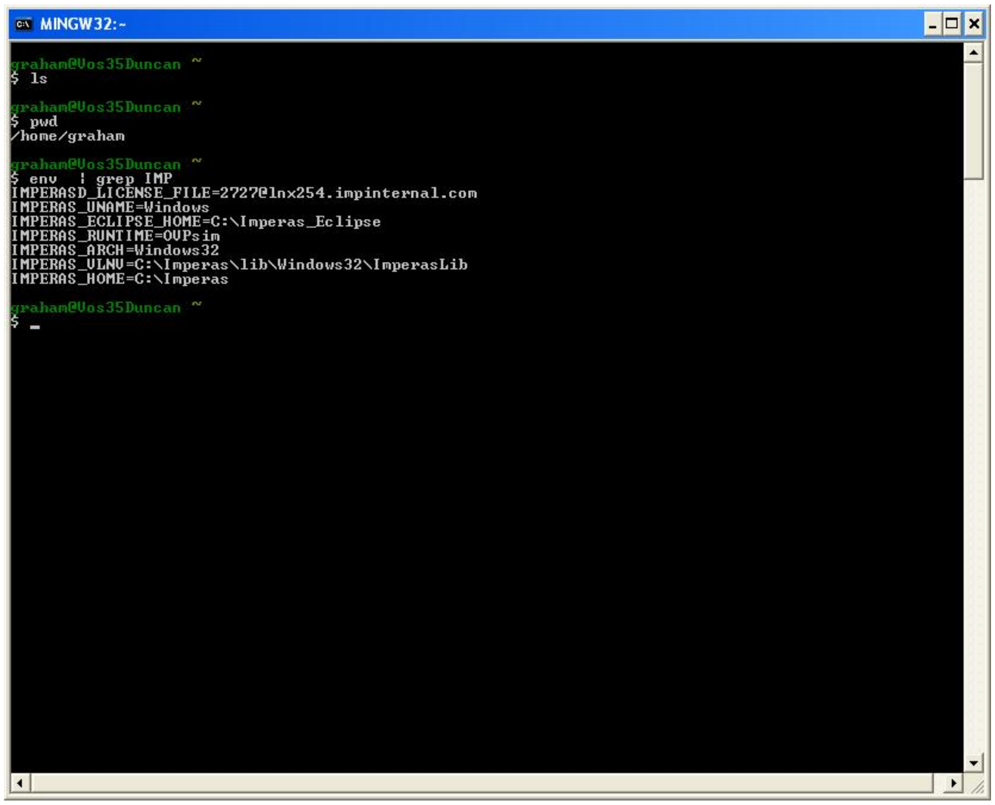
The installation process may have created a desktop shortcut

If not, create your own desktop shortcut by locating the msys.bat file in the installation at C:\MinGW\msys\1.0\msys.bat and create a shortcut and move to your desktop.



Start the MinGW shell and then type ‘ls’ and ‘pwd’ in it to check all is ok and working.

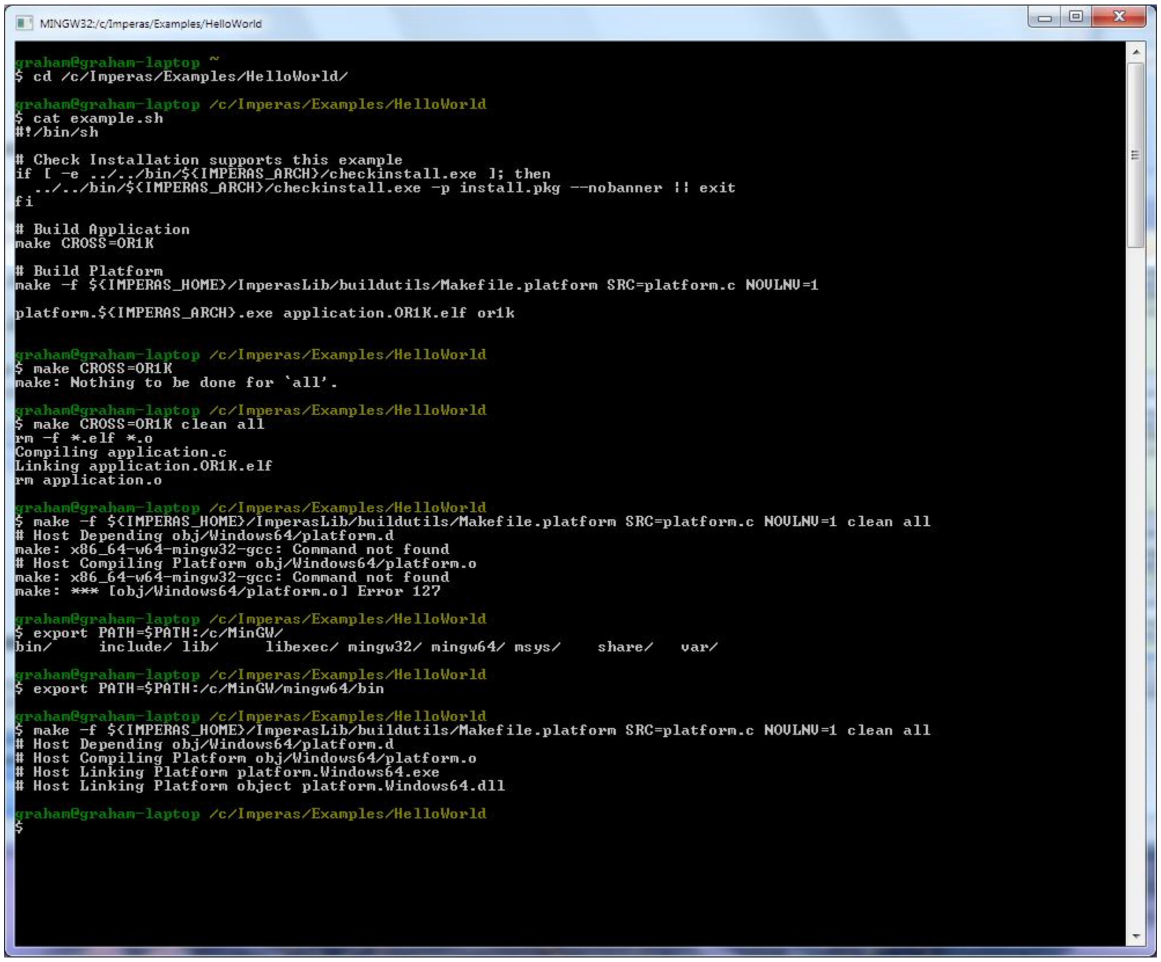
If you already have an OVPsim or Imperas installed try ‘env | grep IMP’ which should show that the Imperas environment has been configured.



```
graham@Vos35Duncan ~
$ ls

graham@Vos35Duncan ~
$ pwd
/home/graham

graham@Vos35Duncan ~
$ env  | grep IMP
IMPERASD_LICENSE_FILE=2727@lnx254.impinternal.com
IMPERAS_UNAME=Windows
IMPERAS_ECLIPSE_HOME=C:\Imperas_Eclipse
IMPERAS_RUNTIME=OVPsim
IMPERAS_ARCH=Windows32
IMPERAS_VLNV=C:\Imperas\lib\Windows32\ImperasLib
IMPERAS_HOME=C:\Imperas

graham@Vos35Duncan ~
$ _
```

With any of the OVPsim, Imperas_SDK or Imperas_DEV packages installed and the or1k.toolchain package installed you can try running the Example Imperas/Examples/HelloWorld that will ensure everything is available and correctly installed.

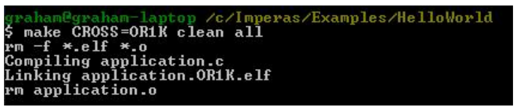
In the following screen shot we show the execution of the example using the script *example.sh*. Below we further describe each section separately



```
graham@graham-laptop ~
$ cd /c/Imperas/Examples/HelloWorld/

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ cat example.sh
#!/bin/sh

# Check Installation supports this example
if [ -e ../../bin/${IMPERAS_ARCH}/checkinstall.exe ]; then
  ../../bin/${IMPERAS_ARCH}/checkinstall.exe -p install.pkg --nobanner || exit
fi

# Build Application
make CROSS=OR1K

# Build Platform
make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform SRC=platform.c NOVLNV=1

platform.${IMPERAS_ARCH}.exe application.OR1K.elf or1k


graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make CROSS=OR1K
make: Nothing to be done for `all'.

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make CROSS=OR1K clean all
rm -f *.elf *.o
Compiling application.c
Linking application.OR1K.elf
rm application.o

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform SRC=platform.c NOVLNV=1 clean all
# Host Depending obj/Windows64/platform.d
make: x86_64-w64-mingw32-gcc: Command not found
# Host Compiling Platform obj/Windows64/platform.o
make: x86_64-w64-mingw32-gcc: Command not found
make: *** [obj/Windows64/platform.o] Error 127

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ export PATH=$PATH:/c/MinGW/
bin/      include/ lib/      libexec/ mingw32/ mingw64/ msys/     share/   var/

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ export PATH=$PATH:/c/MinGW/mingw64/bin

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform SRC=platform.c NOVLNV=1 clean all
# Host Depending obj/Windows64/platform.d
# Host Compiling Platform obj/Windows64/platform.o
# Host Linking Platform platform.Windows64.exe
# Host Linking Platform object platform.Windows64.dll

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$
```

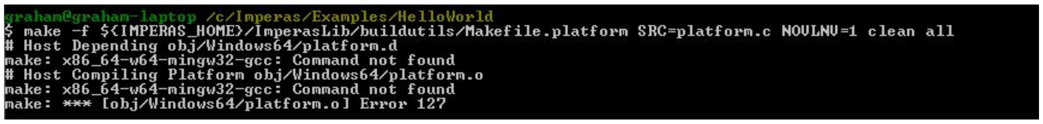
Build the application, setting CROSS to the Cross Compiler toolchain we are using. In this case we set CROSS=OR1K to utilize the installed cross compiler toolchain for the OR1K processor, OR1K.toolchain.

The command line to do this is *make CROSS=OR1K clean all*



```
graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make CROSS=OR1K clean all
rm -f *.elf *.o
Compiling application.c
Linking application.OR1K.elf
rm application.o
```

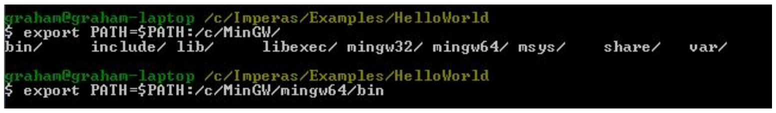
In this next screenshot we attempt to build the platform using the supplied Makefile, Makefile.platform. We specify that we will build locally and not use the VLNV library structure with NOVLNV=1.
The command line used is
*make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform \
NOVLNV=1 clean all*
As you can see the platform cannot be built because the host GCC (x86_64-mingw32-gcc as we are running on a 64-Bit Windows 7 host machine) cannot be found. This is because we have not setup the PATH environment variable.



```
graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform SRC=platform.c NOVLNV=1 clean all
# Host Depending obj/Windows64/platform.d
make: x86_64-w64-mingw32-gcc: Command not found
# Host Compiling Platform obj/Windows64/platform.o
make: x86_64-w64-mingw32-gcc: Command not found
make: *** [obj/Windows64/platform.o] Error 127
```

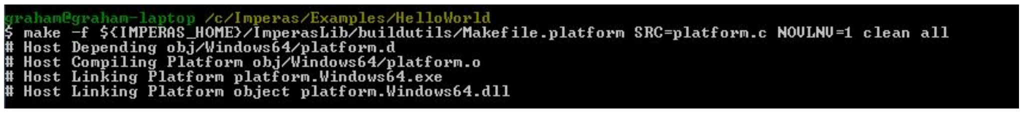
We can setup the PATH environment variable in the shell. Below we show the addition of the bin directory for the 64-bit Host GCC to the PATH.

export PATH=$PATH:/c/MinGW_Toolchains/mingw64/bin



```
graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ export PATH=$PATH:/c/MinGW/
bin/      include/ lib/      libexec/ mingw32/ mingw64/ msys/     share/   var/

graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ export PATH=$PATH:/c/MinGW/mingw64/bin
```

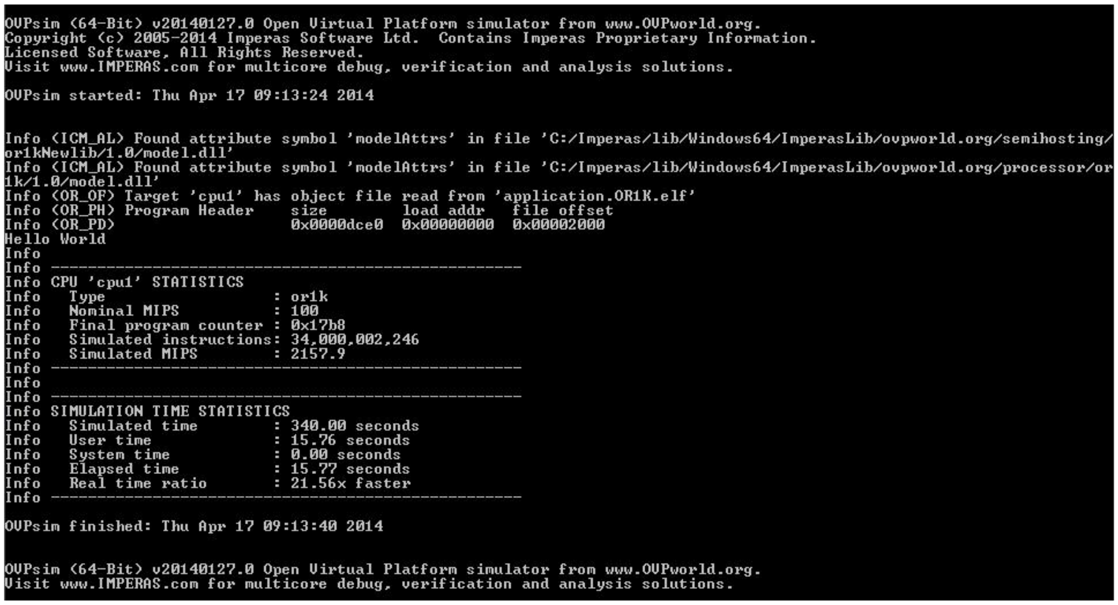
We are now able to re-invoke the same make command and build the platform



```
graham@graham-laptop /c/Imperas/Examples/HelloWorld
$ make -f ${IMPERAS_HOME}/ImperasLib/buildutils/Makefile.platform SRC=platform.c NOVLNV=1 clean all
# Host Depending obj/Windows64/platform.d
# Host Compiling Platform obj/Windows64/platform.o
# Host Linking Platform platform.Windows64.exe
# Host Linking Platform object platform.Windows64.dll
```

We can now run the OVP virtual platform. the simulator loads the OVP Fast processor model (and any other model or semihost libraries required), loads an application into memory and uses the processor model to execute the binary code.

The command line to run the simulation is
platform.${IMPERAS_ARCH}.exe application.OR1K.elf or1k


```
OVPsim (64-Bit) v20140127.0 Open Virtual Platform simulator from www.OVPworld.org.
Copyright (c) 2005-2014 Imperas Software Ltd.  Contains Imperas Proprietary Information.
Licensed Software, All Rights Reserved.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

OVPsim started: Thu Apr 17 09:13:24 2014


Info (ICM_AL) Found attribute symbol 'modelAttrs' in file 'C:/Imperas/lib/Windows64/ImperasLib/ovpworld.org/semihosting/
or1kNewlib/1.0/model.dll'
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file 'C:/Imperas/lib/Windows64/ImperasLib/ovpworld.org/processor/or
1k/1.0/model.dll'
Info (OR_OF) Target 'cpu1' has object file read from 'application.OR1K.elf'
Info (OR_PH) Program Header    size        load addr   file offset
Info (OR_PD)                   0x0000dce0  0x00000000  0x00002000
Hello World
Info
Info ---------------------------------------------------
Info CPU 'cpu1' STATISTICS
Info   Type                  : or1k
Info   Nominal MIPS          : 100
Info   Final program counter : 0x17b8
Info   Simulated instructions: 34,000,002,246
Info   Simulated MIPS        : 2157.9
Info ---------------------------------------------------
Info
Info ---------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 340.00 seconds
Info   User time             : 15.76 seconds
Info   System time           : 0.00 seconds
Info   Elapsed time          : 15.77 seconds
Info   Real time ratio       : 21.56x faster
Info ---------------------------------------------------

OVPsim finished: Thu Apr 17 09:13:40 2014


OVPsim (64-Bit) v20140127.0 Open Virtual Platform simulator from www.OVPworld.org.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.
```
[11]

5.2.3 Checking Imperas tools and OVPsim Installation

When the Imperas tools or OVPsim were installed, new environment variables would have been created, and the PATH environment variable would have been updated to include the 'bin' directory of the installation.

We can check this by starting an Msys shell.

In the MSYS shell, type the following

```
        $ env | grep IMPERAS
```

This should provide us with 4 environment variables shown here with their default values

```
        IMPERAS_HOME=C:\Imperas
        IMPERAS_UNAME=Windows
        IMPERAS_ARCH=Windows32[12]
        IMPERAS_SHRSUF=dll
        IMPERAS_RUNTIME=OVPsim
        IMPERAS_VLNV=C:\Imperas\lib\Windows32\ImperasLib
```

[11] The execution time was extended from the application provided so that statistics would be reported fully.
[12] This will be Windows64 if you are installing one of the Windows 64-bit products

The IMPERAS_RUNTIME will be set to either ***OVPsim*** or ***CpuManager*** dependent upon the installation of OVPsim or the Imperas tools respectively. The Imperas tool installation includes an OVPsim installation; by changing this variable you can choose to run using either of the simulators.

In addition, if we print the value of the PATH environment variable:-

```
$ echo $PATH
```

It should contain the following

```
/c/Imperas/bin/Windows32[13]
```

If you installed, OVPsim or the Imperas tools, in a different directory to the default, then these variables will be changed accordingly.

These environment variables are used by the Makefiles whilst compiling applications, models and platforms / simulation harnesses for the OVP and Imperas tools.

IMPORTANT

We have observed that when other packages are installed in the MSYS environment the msys.bat file used to start up the shell may be modified such that the PATH variable is restricted to the MINGW and MSYS paths. If you find that the PATH environment variable does not contain your full PATH, find the bat file executed from the msys link and if MSYSTEM is set to MINGW32 please remove.

[13] This will be Windows64 if you are installing one of the Windows 64-bit products

5.3 Using a Cygwin Environment

5.3.1 Use MINGW GNU Toolset

Although, the MSYS environment is recommended it is also possible to make use of the Cygwin environment.

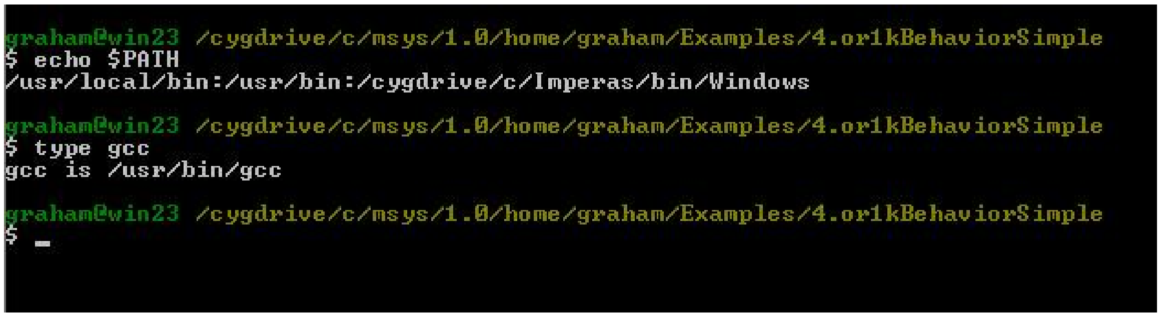
If an installation of cygwin is used as the environment it must be ensured that the MINGW GNU tools are used during the build of any application or DLL.

Using the MINGW GNU toolset provides a native Windows executable.

Using the Cygwin GNU toolset generates executables and DLLs that will reference and be reliant on the Cygwin DLLs. This will create executables and DLLs that are not portable and therefore not compatible in other environments. There are compatibility issues even between different versions of Cygwin.

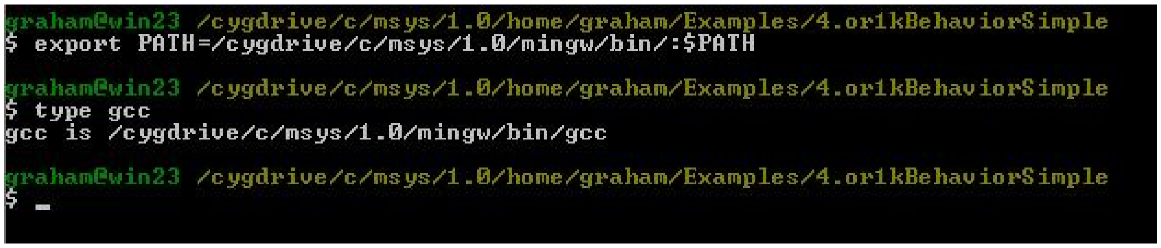
5.3.2 Verify use of MINGW GNU Toolset

To verify that the correct environment is being used check the version of gcc that is found on the path in a cygwin shell. The following illustrates that the incorrect version of gcc would be used is from the Cygwin install.



```
graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ echo $PATH
/usr/local/bin:/usr/bin:/cygdrive/c/Imperas/bin/Windows

graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ type gcc
gcc is /usr/bin/gcc

graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ _
```

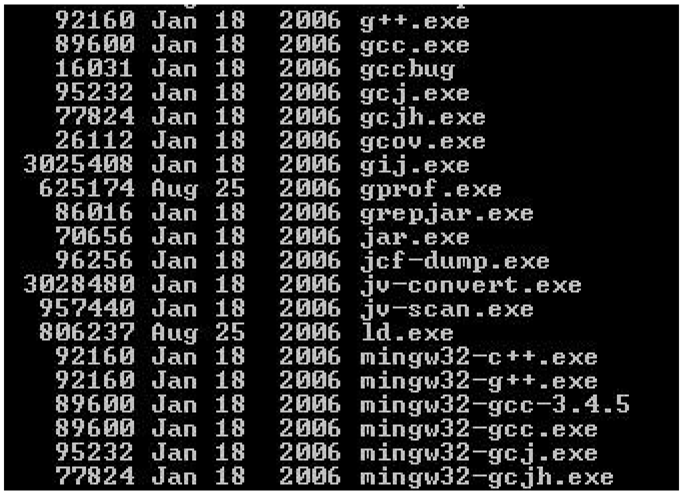
We should modify the path so that the MINGW GNU tools appear before the Cygwin tools.



```
graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ export PATH=/cygdrive/c/msys/1.0/mingw/bin/:$PATH

graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ type gcc
gcc is /cygdrive/c/msys/1.0/mingw/bin/gcc

graham@win23 /cygdrive/c/msys/1.0/home/graham/Examples/4.or1kBehaviorSimple
$ _
```

We can see now that the GNU tools are being found from the MINGW installation before the Cygwin.
Note that this installation of MINGW has been made under an MSYS installation.

Note that when creating Makefiles or scripts to build executables and DLLs it is often useful to use MINGW specific names of the GNU tool executables. These appear in the mingw/bin directory as, for example, mingw-gcc.exe. By doing this you will guarantee that you do not inadvertently pick up the Cygwin tools. There are mingw-* named copies of all the GNU tools.



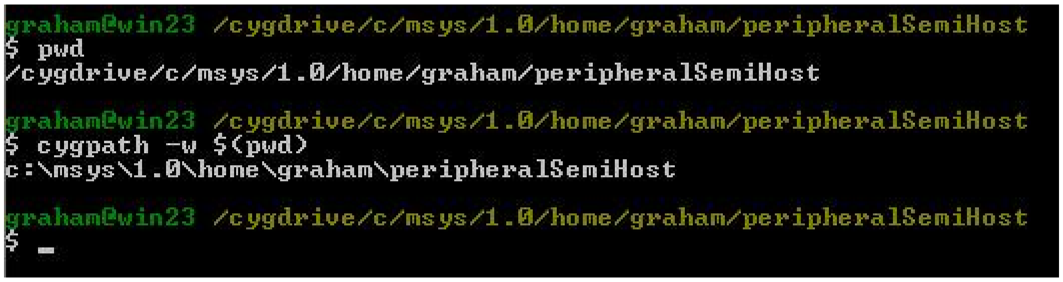
```
   92160 Jan 18  2006 g++.exe
   89600 Jan 18  2006 gcc.exe
   16031 Jan 18  2006 gccbug
   95232 Jan 18  2006 gcj.exe
   77824 Jan 18  2006 gcjh.exe
   26112 Jan 18  2006 gcov.exe
 3025408 Jan 18  2006 gij.exe
  625174 Aug 25  2006 gprof.exe
   86016 Jan 18  2006 grepjar.exe
   70656 Jan 18  2006 jar.exe
   96256 Jan 18  2006 jcf-dump.exe
 3028480 Jan 18  2006 jv-convert.exe
  957440 Jan 18  2006 jv-scan.exe
  806237 Aug 25  2006 ld.exe
   92160 Jan 18  2006 mingw32-c++.exe
   92160 Jan 18  2006 mingw32-g++.exe
   89600 Jan 18  2006 mingw32-gcc-3.4.5
   89600 Jan 18  2006 mingw32-gcc.exe
   95232 Jan 18  2006 mingw32-gcj.exe
   77824 Jan 18  2006 mingw32-gcjh.exe
```

5.3.3 Cannot Build OVPsim Examples

The Mingw gnu tools are native Windows tools and as such are not expecting Cygwin path names.

There is a utility ‘cygpath’ that will convert a Cygwin path into a native windows path. This may be used to give the correct path for the MINGW GNU tools.

cygpath –w $(pwd)



```
graham@win23 /cygdrive/c/msys/1.0/home/graham/peripheralSemiHost
$ pwd
/cygdrive/c/msys/1.0/home/graham/peripheralSemiHost

graham@win23 /cygdrive/c/msys/1.0/home/graham/peripheralSemiHost
$ cygpath -w $(pwd)
c:\msys\1.0\home\graham\peripheralSemiHost

graham@win23 /cygdrive/c/msys/1.0/home/graham/peripheralSemiHost
$ _
```

Unfortunately this cannot be used in conjunction with the Makefile system provided with the OVPsim examples because the Windows native path contains a ‘:’ which is a special character when used in Makefiles!
The OVPsim examples are all verified for use in the MSYS environment.

5.4 Building Virtual platforms with Microsoft MSVC

The Microsoft Visual C++ environment may be used to build the virtual platform executable for a Windows host.

An example is provided in the Platform Examples section to show the compilation of a virtual platform using MSVC.

5.4.1 Running the Example

Take a copy of the example:

```
> cp –r $IMPERAS_HOME/Examples/Platforms/simple_MSVC_compile .
```

5.4.1.1 Compiling the CpuManager Platform

The test platform can be compiled to produce an executable, `platform.${IMPERAS_ARCH}.exe`, using the MSVC environment.

To compile using MSVC the setup program must be invoked that is provided with MSVC to start a shell with an appropriate environment. This is typically called a ‘Visual Studio 2008 Command Prompt’
A script in the platform directory, compile.msvc.bat, contains the following command line that is used to build using nmake and the Makefile provided.

```
> nmake –nologo –s –f nmakefile
```

Note that for completeness this example can also be built in a Linux shell or in a MinGW shell on Windows by using the following command in the `platform` directory:

```
> make –C platform
```

5.4.1.2 Creating an Application Executable

A test case must be created using the processor tool chain. Because the OR1K processor is supported by Imperas tools and shipped as an example, there is already an encapsulated tool chain that you can use to compile test cases for it.

Within the `platform` directory is a simple assembler test, `application/asmtest.S`, which simply performs a few instructions and exits. The application can be compiled using the following command in the `platform` directory:

```
> make –C application
```

The result is an ELF format file for the OR1K called `asmtest.OR1K.elf`.

5.4.1.3 Running the Simulation

Having compiled the test platform and application, you are now ready to run a simulation. Do this by running the following in the `example` directory:

```
platform/platform.${IMPERAS_ARCH}.exe application/asmtest.OR1K.elf
```

You should see the following output:

```
Processor ‘cpu1’ terminated at ‘exit’, address 0x10000bc
```

This message is printed by the *imperasExit* semihosting library as the processor executes the first instruction at `exit` in the application.

6 Installing Additional Tools

Additional tools are provided in Linux or Windows installers. These are available from the OVP website, www.OVPworld.org

IMPORTANT

The Cross Compiler and Peripheral Simulation Engine toolchains are provided only as 32-bit native executables. This requires that a 64-bit host must provide a 32-bit compatibility mode in order that they can execute. The packages they are provided in are labeled as 64-bit packages, for example Linux64 only to indicate that they support a 64-bit Imperas or OVPsim installation.

6.1 Cross Compiler Toolchains

Imperas provides example pre-built toolchains for processors provided by OVP (the toolchains are available at www.OVPworld.org).

NOTE

The Cross Compiler toolchains are provided to allow a user to quickly start generating application binary code that can be executed on OVP processor models.
It is expected that after the initial experiments a cross compiler toolchain will be selected from a cross compiler supplier.
Some processors and processor variants are not supported by the cross compiler toolchains available on the OVPWorld website and alternatives are indicated to download and install.

Also as part of these cross compiler toolchains is a Makefile to provide a default build environment for an application onto a processor. The Makefiles are found in the directory ***`IMPERAS_HOME/lib/IMPERAS_ARCH/CrossCompiler`*** in the format ***`<Processor Type/Variant>.makefile.include`***, for example ***`MIPS32.Makefile.include`***.

Each cross compiler toolchain should be installed into the current installation of Imperas Professional or OVPsim.

6.2 Peripheral Simulation Engine Toolchain

A toolchain is provided to allow behavioral code to be compiled for a Peripheral Simulation Engine (PSE). This is the simulation engine that is used to provide peripheral models in a virtual platform simulation.

When installed this provides the toolchain ***pse-elf*** and a ***PSE.makefile.include***.

This toolchain should be installed into the current installation of Imperas tools or OVPsim.

7 Getting Started

The following should be carried out in a Linux shell or on Windows an MSYS shell.

7.1 Check Tool Installation

This section provides some checks that the tools have been correctly installed.

7.1.1 OVPsim and Imperas Simulator

To verify the installation after an OVPsim or an Imperas install one of the Demos that are downloaded with both installations may be executed. These Demos will work with the OVPsim and also the Imperas Simulator.

For example, execute the fibonacci benchmark on a single core OR1K platform.

```
> cd $IMPERAS_HOME/Demo/OVPsim_or1k
> ./OVPsim_or1k.<OS>.exe ./fibonacci.OR1K.elf
```

This should provide an output similar to that shown below:

```
OVPsim (32-Bit) v20130630.0 Open Virtual Platform simulator from www.IMPERAS.com.
Copyright (c) 2005-2012 Imperas Software Ltd.  Contains Imperas Proprietary Information.
Licensed Software, All Rights Reserved.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

OVPsim started: Tue Jan 29 12:16:59 2013


Info (ICM_AL) Found attribute symbol 'modelAttrs' in file
'C:/Imperas/lib/Windows32/ImperasLib/ovpworld.org/semihosting/or1kNewlib/1.0/model.dll'
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file
'C:/Imperas/lib/Windows32/ImperasLib/ovpworld.org/processor/or1k/1.0/model.dll'
Info (OR_OF) Target '/cpu0' has object file read from './fibonacci.OR1K.elf'
Info (OR_SH)     Section        flg  sect addr   size        load addr   file offset
Info (OR_SD)     .text          -ax  0x00000000  0x0000d0f4  0x00000000  0x00002000
Info (OR_SD)     .rodata        -a-  0x0000d0f4  0x0000045c  0x0000d0f4  0x0000f0f4
Info (OR_SD)     .data          wa-  0x0000d550  0x00000878  0x0000d550  0x0000f550
starting...
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
<snip>
fib(37) = 24157817
finishing...
Info
Info ---------------------------------------------------
Info CPU '/cpu0' STATISTICS
Info   Type                 : or1k
Info   Nominal MIPS         : 100
Info   Final program counter : 0x1884
Info   Simulated instructions: 4,093,555,347
Info   Simulated MIPS        : 494.3
Info ---------------------------------------------------
Info
Info ---------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 40.94 seconds
Info   User time             : 8.25 seconds
Info   System time           : 0.00 seconds
Info   Elapsed time          : 8.28 seconds
Info   Real time ratio       : 4.94x faster
Info ---------------------------------------------------

OVPsim finished: Tue Jan 29 12:17:07 2013
```

7.1.2 Imperas Tool Installation

There are two executables supplied with the Imperas tools installation, imperas.exe and igen.exe

```
> imperas.exe --version
```

This should provide an out put similar to that shown below:

```
(MS_VSN)
        IMPERAS SIMULATOR (32-Bit) version 20130630.0
        Copyright (c) 2005-2012 Imperas Software Ltd.
                     ALL RIGHTS RESERVED

This program is proprietary and confidential information of
Imperas Software Ltd. and may be used and disclosed only as authorized
in a license agreement controlling such use and disclosure.
```

```
> igen.exe --version
```

This should provide an out put similar to that shown below:

```
(MS_VSN)
        IMPERAS IGEN (32-Bit) version 20130630.0
        Copyright (c) 2005-2012 Imperas Software Ltd.
                     ALL RIGHTS RESERVED

This program is proprietary and confidential information of
Imperas Software Ltd. and may be used and disclosed only as authorized
in a license agreement controlling such use and disclosure.
```

7.2 Cross Compiler Toolchain

For this example the OR1K toolchain will be used to compile an application to execute on the OR1K processor model contained in a simple platform.

7.2.1 Download OR1K toolchain

Go to the OVPworld.org downloads page http://www.ovpworld.org/download.php and then go to the OR1K menu tab to get to the OR1K downloads page: http://www.ovpworld.org/download_OR1K.php. Look on the right hand side list to find the appropriate (Windows or Linux) installer for the OR1K Toolchains. This will download either an installer for Windows (or1k.toolchain.<major version>.<dot release>.Windows32.exe) or an installer for Linux (or1k.toolchain.<major version>.<dot release>.Linux32.exe)

7.2.2 Install OR1K toolchains

The toolchains are provided in the same way as the Imperas tools and OVPsim installations; that is as a Windows or Linux installation executable.

The version of the toolchain available is the same as the version of the Imperas tools or OVPsim available. When extracted they should be installed into the same directory. For example, into the directory <installDir>/Imperas.<major version> on Linux.

7.3 Compiling a first application

There are many different directories of examples in the directories `$IMPERAS_HOME/Examples` and `$IMPERAS_HOME/Demo`.

For our first application compilation let’s copy some examples to our working area.

```
$ cp –r “$IMPERAS_HOME/Examples/HelloWorld” .
$ cd HelloWorld
$ ls
```

In this directory you should see a platform description file platform.c, an application application.c, which can run on this platform, and a Makefile for building the application.

A Makefile for building the platform is provided in the ImperasLib library infrastructure at $IMPERAS_HOME/ImperasLib/buildutils/Makefile.platform

Firstly let’s try compiling the Platform (note this example shows the Linux ‘\’ line continuation character)

```
$ make –f $IMPERAS_HOME/ImperasLib/buildutils/Makefile.platform \
        SRC=platform.c \
        NOVLNV=1
```

This should produce a file called ‘platform.<OS>.exe’; where OS could be Windows32 or Linux32, depending upon your host platform.
Now let’s compile the Application

```
$ make CROSS=OR1K
```

This should cross compile our application for the OR1K processor, producing a file called ‘application.OR1k.elf’

We can now run our application on our platform – for example on a Windows host

```
$ platform.Windows32.exe application.OR1K.elf or1k
```

This should provide an output similar to that shown below

```
OVPsim v20130630.0 Open Virtual Platform simulator from www.OVPworld.org.
Copyright (c) 2005-2011 Imperas Software Ltd.  Contains Imperas Proprietary
Information.
Licensed Software, All Rights Reserved.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

OVPsim started: Wed Jan 04 13:19:11 2012


Warning (LIC_EXP) License feature 'IMP_OVPSIM_20130630' will expire in 26 days
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file
'C:/Imperas/lib/Windows32/ImperasLib/ovpworld.org/semihosting/or1kNewlib/1.0/mod
el.dll'
Info (ICM_AL) Found attribute symbol 'modelAttrs' in file
'C:/Imperas/lib/Windows32/ImperasLib/ovpworld.org/processor/or1k/1.0/model.dll'
Info (OR_OF) Target '/cpu1' has object file read from 'application.OR1K.elf'
Info (OR_SH)     Section        flg  sect addr   size        load addr   file offset
Info (OR_SD)     .text          -ax  0x00000000  0x0000cf90  0x00000000  0x00002000
Info (OR_SD)     .rodata        -a-  0x0000cf90  0x00000440  0x0000cf90  0x0000ef90
Info (OR_SD)     .data          wa-  0x0000d3d0  0x00000878  0x0000d3d0  0x0000f3d0
Hello World
Info
Info ---------------------------------------------------
Info CPU '/cpu1' STATISTICS
Info   Type                  : or1k
Info   Nominal MIPS          : 100
Info   Final program counter : 0x1720
Info   Simulated instructions: 2,226
Info   Simulated MIPS        : run too short for meaningful result
Info ---------------------------------------------------
Info
Info ---------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 0.00 seconds
Info   User time             : 0.02 seconds
Info   System time           : 0.00 seconds
Info   Elapsed time          : 0.02 seconds
Info ---------------------------------------------------

OVPsim finished: Wed Jan 04 13:19:11 2012


OVPsim v20130630.0 Open Virtual Platform simulator from www.OVPworld.org.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.
```

NOTE

The ‘Simulated MIPS’ figure provides an indication of how many millions of instructions the simulator was able to run on your host machine per second. You will get the message ‘run too short for meaningful result’ if the simulated time is less than a second.

7.4 Semihosting Support

7.4.1 In Imperas and OVP simulations

The term "semihosting" is over used and can mean several things. In the case of OVP simulations the term semihosting refers to specific behavior that is provided by the Host

system rather than the simulation platform or an operating system running on the simulation platform.
For example, in an embedded system to get output on a terminal the platform would typically include a hardware device, such as a UART, and the application software would provide a driver to initialize the device and provide low level functions that can be used by library functions, for example printf, to output characters to the terminal. This adds complexity to the software and the platform that is not required if we simply wanted to cross compile and run a bare metal C application on a specific processor variant.
In this example semihosting would be used to provide behavior for the low level functions without having to add anything to the platform or application code. A semihost library replaces behavior of the low level function in the cross compiled application with a native host function provided within a semihosting intercept library.
The semihost library is native code that is loaded as a shared object by the simulator. It is defined in the instantiation of a processor instance in a platform.

7.4.2 Replacing function and/or instruction behavior

Different mechanisms may be used to support semihosting. This may be replacing a functions behavior or using one or more[14] specific instructions to control semihost operations.

A semihost library may replace a function in its entirety to provide that function itself, for example the function _write, which sends a character from a buffer to a hardware device output register, may be intercepted and replaced by a semihost function that takes the character from the buffer and displays it on the host stdout.

A semihost library may replace a specific instruction, typically a software interrupt instruction, that is a way of the application program requesting an action from a Host operating system, but that in this case is intercepted and the function performed

7.4.3 Specific to a Cross Compiler and C Library

The way a semihost library is intended to work is specific to a processor type and also to the cross compiler toolchain and C library used during the application compilation and linkage. It is, therefore, important that the correct semihost library is included with the processor instantiation in a platform for the cross compiler and C library used to build the application that will execute on the processor.

Example mips32 CodeSourcery cross compiler toolchain uses mips32Newlib semihost library.
The C library used in the application compilation in this case is specified by a linker script, *mipssim-hosted.ld*. This controls the C library included. The semihosting is performed at the low level function level, as defined in the *.intercept* table of the semihost file mips32Newlib.c in the VLNV library.

Example ARM Linaro cross compiler toolchain used armAngel semihost library.

[14] The actual number of instruction types that can be used is limited in a semihost library

The C library used in the application compilation in this case is specified by a specs file, *aprofile-ve.specs*. This controls the C library included. The semihosting is performed at the instruction level, as defined in the *armOSOperation* function of the semihost file armAngel.c in the VLNV library.

7.4.4 Used to terminate the simulation

The semihost library may also detect other events in the execution of application software, for example a call to *exit* or a jump-to-self.
An embedded processor would never stop executing instructions, unless put into a specific mode. In order to stop an application 'running off the end' designers typically put spin loops in areas of code that are used to capture unwanted events from which the program cannot recover including the 'exit' function code.
The execution of a spin loop could cause the application to run forever and make the simulation appear to lock-up so the semihost library is used to detect spin loops or calls to exit and terminate the simulation.

7.4.5 Caution using with EPKs and non-baremetal platforms

The semihost library replaces functions and/or instructions behavior so it is not the case that a semihost library should always be used. In an EPK or other non-baremetal platform care should be taken to avoid the situation that code is not executed when expected because a symbol has happened to match one defined in the semihost library and has caused the functions normal behavior to be replaced by that of the semihost library.

7.5 Simulation Time Statistics

At the end of simulation, if the verbose flag has been used, some statistics will be reported by the simulator. This provides information to indicate how many instructions were executed on the OVP Processor model and the equivalent simulated Millions of Instructions Per Second (MIPS) that were executed. This can be used as an indication of how well the simulator performed on the host machine running the particular application.

The typical simulation statistics is shown below. Most entries are self explanatory but are described in the following sections

```
Info ---------------------------------------------------
Info CPU 'platform/cpu0' STATISTICS
Info   Type                  : arm (Cortex-A9UP)
Info   Nominal MIPS          : 100
Info   Final program counter : 0x4b8
Info   Simulated instructions: 22,400,008,761
Info   Simulated MIPS        : 5720.7
Info ---------------------------------------------------
Info
Info ---------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time        : 224.00 seconds
Info   User time             : 3.92 seconds
Info   System time           : 0.00 seconds
Info   Elapsed time          : 3.93 seconds
Info   Real time ratio       : 56.98x faster
Info ---------------------------------------------------
```

OVP Fast processor model configuration information

```
Info   Type                   : arm (Cortex-A9UP)
Info   Nominal MIPS           : 100
```

The model type and, if selected, the configuration variant are displayed. The nominal MIPS for the processor model, by default, is 100 MIPS. This has an effect when running with 'wallclock' enabled or when in a multi-processor system. With 'wallclock' enabled it is used to control the maximum execution performance i.e. only 100 million instructions will be executed in a real time 1 second. In a multi-processor system (without wallclock enabled) it provides an execution ration between processors e.g. a 200 MIPS processor will execute twice the instructions as a 100 MIPS processor in a simulation unit of time.

Program execution information

```
Info   Final program counter : 0x4b8
Info   Simulated instructions: 22,400,008,761
```

The 'Simulated instructions' will vary depending upon the application being executed, this count indicates the number of simulated processor instructions for the processors in the platform. The final program counter can be useful as an easy way of showing the application program executes in a deterministic way over a number of simulation runs.

Simulated MIPS

```
Info   Simulated MIPS         : 5720.7
```

The 'Simulated MIPS' will be a measure of the number of 'Simulated instructions' over the host elapsed time and is an indication of the performance of the simulator.

There is an overhead at the start of simulation with Just-In-Time code morphing simulators and so the Simulated MIPS are not calculated if the simulated run time is below a threshold. If you see the following output you are typically running too few instructions i.e. your application is too short.

```
Info   Simulated MIPS         : run too short for meaningful result
```

Simulation time statistics

The simulated time is the time it would have taken to run the application based upon the MIPS configuration of the processor and the number of instructions the application took to complete.

```
Info   Simulated time         : 224.00 seconds
```

The time fields are real time information from the native Host machine relating to the simulator execution.

```
Info   User time            : 3.92 seconds
Info   System time          : 0.00 seconds
Info   Elapsed time         : 3.93 seconds
```

The "real time ratio" is an indication of how much quicker the simulator was able to execute the processor instructions than the real silicon based upon the OVP Fast processor model MIPS rate configuration and the time the simulator took to execute the instructions on the native host machine.

```
Info   Real time ratio       : 56.98x faster
```

8 Understanding the operation of a code morphing simulator

8.1 Instruction Fetch

A code morphing simulator operates in two phases to provide the behavior of a target processor executing a sequence of instructions.

1. Morphing cross compiled target instructions (for example ARM Cortex-A9, MIPS 1074Kc, V850) to a sequence of native host instructions
2. Executing the sequence of native host instructions to provide the behavior expected of the target

In 1 we are executing in 'morph time' using the translations specified by the OVP processor model. During the translation process, each instruction is read from the virtual platform memory using a debug (artifact) transaction. This access must have no effect on the system and must be implemented as a 'back door' access that bypasses any behavior of the system; including timing or delays that may be in a SystemC TLM2 virtual platform implementation.
The code morpher will continue reading instructions and morphing code until it has constructed an entire 'code block'. This may be a few or many instructions.
Once a code block is created, the block is executed. We are now in 2, in 'run time', for each target instruction behavior, a) the instruction is reread from memory - but this time as a 'real' transaction that will have an effect on the system that the fetch would normally have and b) the native code is executed to create the behavior.

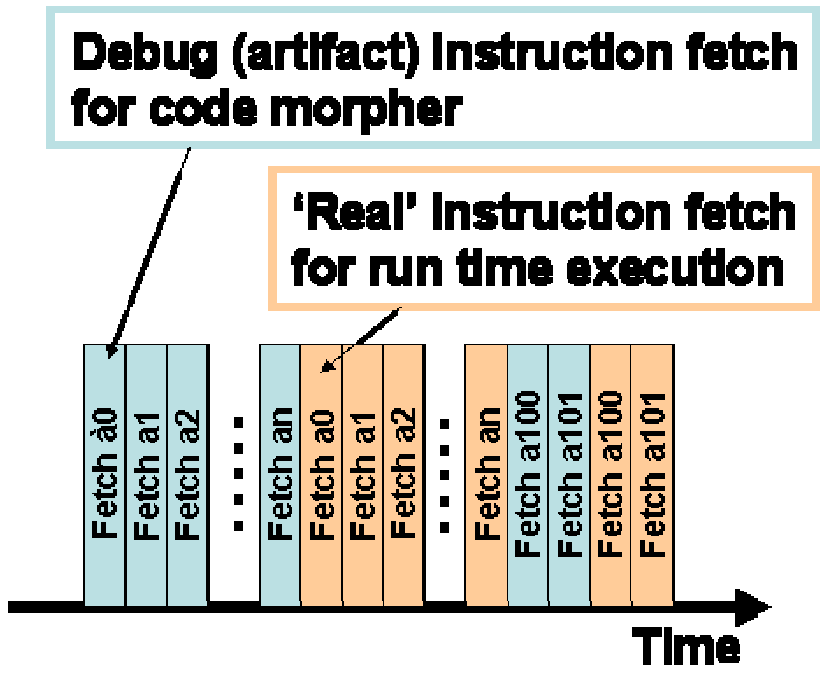


Figure 1: Real and Artifact accesses over time

8.2 SystemC Interface Transaction Types

In SystemC TLM2 interface code is generated for each OVP fast processor model.

There is also generic OVP processor interface code that is provided in the installed component source library.
The component source library is found at IMPERAS_HOME/ImperasLib/source, with the V/L/N/V of the TLM2.0 support models ovpworld.org/modelSupport/tlmProcessor/1.0 containing the interface file tlm2.0/tlmProcessor.cpp

In this file you will find the function icmCpuMasterPort::readUpCall that will perform a TLM2.0 read (fetch) transaction by calling either socket->b_transport() for a 'real' access or socket->transport_dbg() for a 'debug' (simulation artifact) access.

The SystemC virtual platform must provide both debug and real interface connections.

APPENDIX A Installation Packages, Products and Licensing Features

A.1 Installation Packages

There are three main product installation packages available to download via the Imperas or OVP World websites.

1. Imperas_SDK
2. Imperas_DEV
3. OVPsim

In addition to the product installation packages above are a number of installers providing specific demonstrations and example toolchains to be used in compilation of peripheral models and cross compilation of processor application code.

A.1.1 Imperas_SDK package

The Imperas_SDK package is available from the Imperas website and provides the M*SDK product features.

These include

1. Verification, Analysis and Profiling (VAP) tools
2. Multicore debugger for debugging of applications running on processor models and peripheral models in a unified environment
3. M*SIM professional simulator
4. Imperas Generator tool for generating virtual platforms and model (processor and peripheral) templates
5. Examples and Demonstrations

A.1.2 Imperas_DEV package

The Imperas_DEV package is available from the Imperas website and provides the Developer installations; C*DEV, S*DEV[15] and M*DEV which provide different simulator features. The package provides the DEV product features.

These include

1. M*SIM professional simulator
2. Imperas Generator tool for generating virtual platforms and model templates
3. Examples and Demonstrations

The setting of the IMPERAS_PERSONALITY environment variable is used to select the simulator features, between those provided by Controller (C*DEV), Standard (S*DEV) and Multi (M*DEV)

[15] The ‘Standard’ Developer simulator configuration provides similar capabilities to the OVPsim simulator.

A.1.3 OVPsim package

The OVPsim package is available from the OVP World website. The package provides the product features.

These include

1. OVPsim reference simulator
2. Examples and Demonstrations

A.2 License Features

A.2.1 Executable Programs

The following table shows the executable programs that are provided in the product packages.

Product Name	Package	License Feature	Simulator / Personality
Imperas.exe	Imperas_SDK	IMP_MPD	CpuManager / CPUMAN_MULTI
mpd.exe	Imperas_SDK	IMP_MPD	Not required
Igen.exe	Imperas_SDK/Imperas_DEV	IMP_IGEN	Any / Any
iPost.exe	Imperas_SDK	IMP_IPOST	Not required
cpuGen	Imperas_SDK	IMP_IGEN + IMP_CPUGEN	Any / Any

A.2.2 CpuManager Simulator and its Personalities

The CpuManager simulator runtime can be configured to provide different levels of simulation support. The configuration of the simulator is controlled with the IMPERAS_PERSONALITY environment variable. The simulator has a default personality equivalent to setting IMPERAS_PERSONALITY to CPUMAN_MULTI. The license features that the simulator will checkout are controlled by the personality.

Simulator Personality	Product	License Feature
CPUMAN_CONTROLLER	C*DEV	IMP_CPUMAN_CONTROLLER
CPUMAN_STANDARD	S*DEV	IMP_CPUMAN_STANDARD
CPUMAN_MULTI (default)	M*SDK and M*DEV	IMP_CPUMAN_MULTI
OVPSIM	OVPsim	IMP_OVPSIM

A.2.3 OVPsim Simulator

The OVPsim simulator does not use the IMPERAS_PERSONALITY environment variable and provides fixed simulation support, equivalent of the Developer S*DEV product.

APPENDIX B Obtaining an Imperas License

To obtain an

Imperas license	send an email request to **license_support@imperas.com**,
OVPsim license	visit the OVPworld.org license page, or send an email to license@ovpworld.org

The email sent must contain the following information:

Name	of the person responsible for the Imperas/OVP Tools
e-mail address	
Phone number	
Company Address	
Host id[16]	of the machine running the license server
Host Name	
Type (Linux, Windows)	

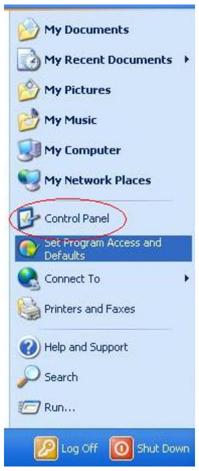
[16] This is the FlexLM information obtained using the lmutil(.exe) that is installed with all packages. See section 4.5 Setting up Licensing (Both Windows and Linux)

APPENDIX C Setting Environment Variables

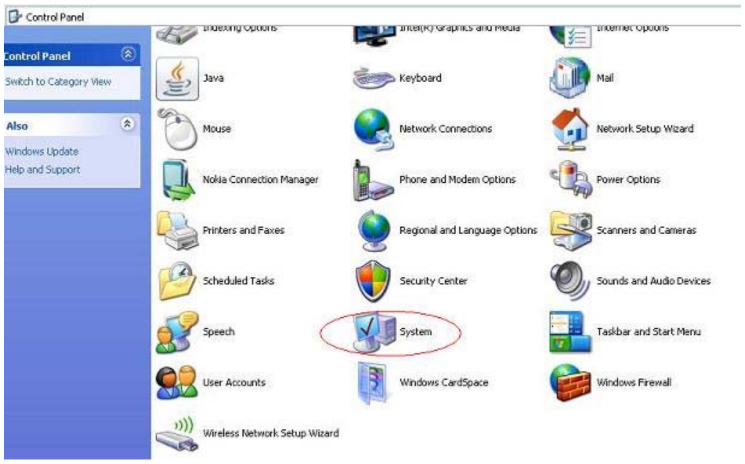
C.1 Opening System properties on Windows XP

In order to set an environment variable using Windows XP follow these steps

Select 'Control Panel'

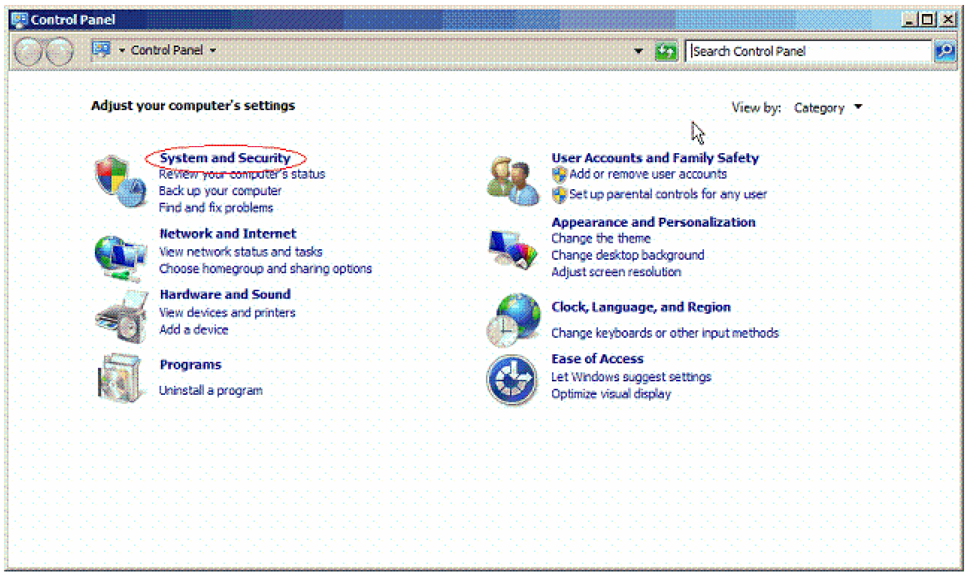


Select 'System' to bring up the 'System properties' dialogue box.

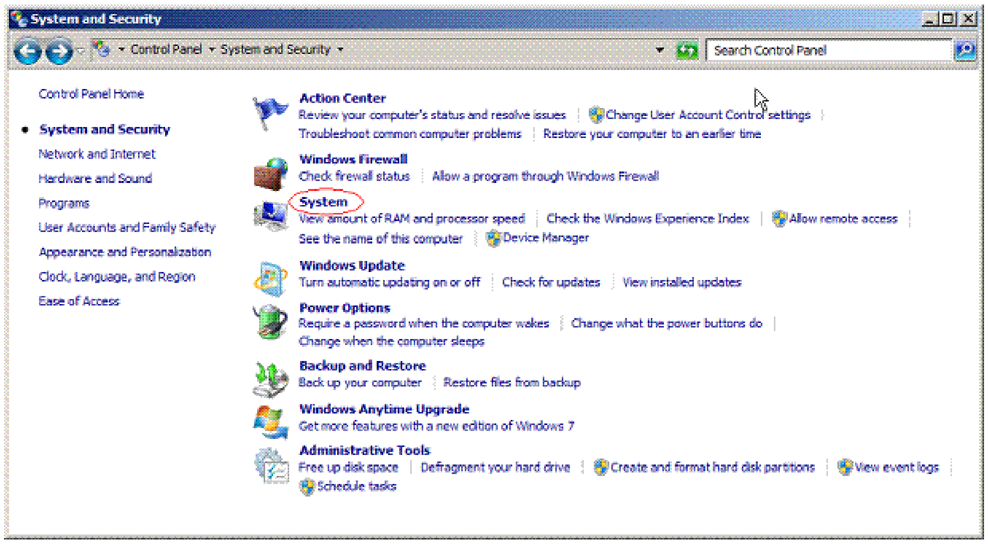


C.2 Opening System properties on Windows 7

Select 'Control Panel' and then select 'System and Security'

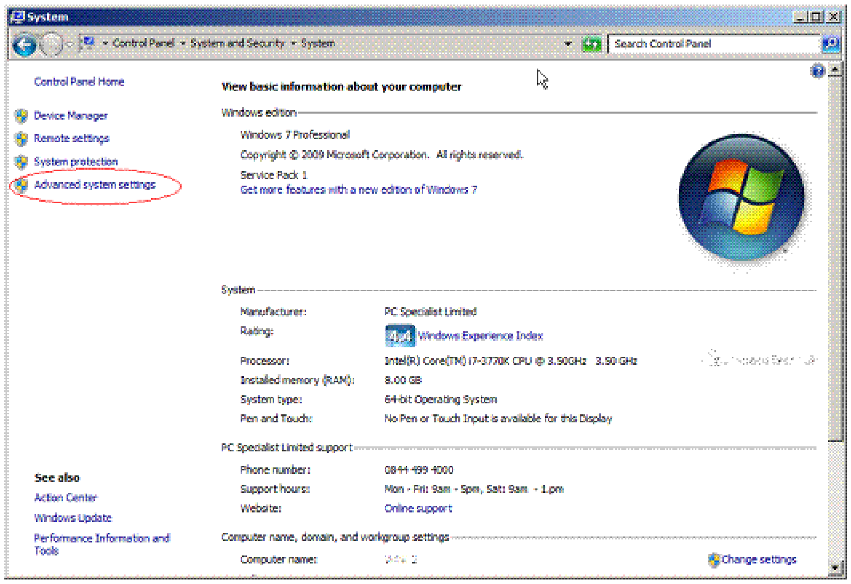

Control Panel
Control Panel
Search Control Panel
Adjust your computer's settings
View by: Category
System and Security
Review your computer's status
Back up your computer
Find and fix problems
Network and Internet
View network status and tasks
Choose homegroup and sharing options
Hardware and Sound
View devices and printers
Add a device
Programs
Uninstall a program
User Accounts and Family Safety
Add or remove user accounts
Set up parental controls for any user
Appearance and Personalization
Change the theme
Change desktop background
Adjust screen resolution
Clock, Language, and Region
Change keyboards or other input methods
Ease of Access
Let Windows suggest settings
Optimize visual display

Now select 'System'


System and Security
Control Panel ▸ System and Security
Search Control Panel
Control Panel Home
System and Security
Network and Internet
Hardware and Sound
Programs
User Accounts and Family Safety
Appearance and Personalization
Clock, Language, and Region
Ease of Access
Action Center
Review your computer's status and resolve issues
Change User Account Control settings
Troubleshoot common computer problems
Restore your computer to an earlier time
Windows Firewall
Check firewall status
Allow a program through Windows Firewall
System
View amount of RAM and processor speed
Check the Windows Experience Index
Allow remote access
See the name of this computer
Device Manager
Windows Update
Turn automatic updating on or off
Check for updates
View installed updates
Power Options
Require a password when the computer wakes
Change what the power buttons do
Change when the computer sleeps
Backup and Restore
Back up your computer
Restore files from backup
Windows Anytime Upgrade
Get more features with a new edition of Windows 7
Administrative Tools
Free up disk space
Defragment your hard drive
Create and format hard disk partitions
View event logs
Schedule tasks

.

Finally select ‘Advanced system settings’


System

Control Panel ▸ System and Security ▸ System

Search Control Panel

Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings

See also
Action Center
Windows Update
Performance Information and Tools

View basic information about your computer

Windows edition
Windows 7 Professional
Copyright © 2009 Microsoft Corporation. All rights reserved.
Service Pack 1
Get more features with a new edition of Windows 7

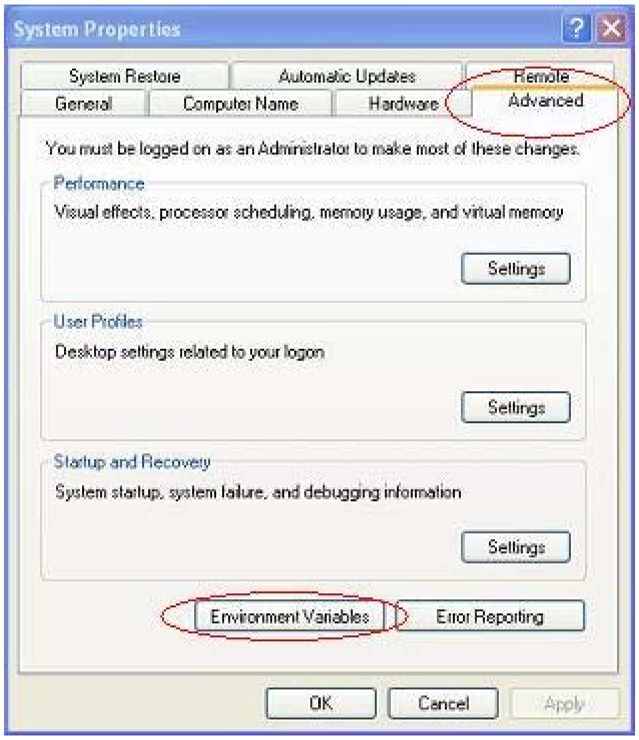
System
Manufacturer: PC Specialist Limited
Rating: 4.4 Windows Experience Index
Processor: Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz 3.50 GHz
Installed memory (RAM): 8.00 GB
System type: 64-bit Operating System
Pen and Touch: No Pen or Touch Input is available for this Display

PC Specialist Limited support
Phone number: 0844 499 4000
Support hours: Mon - Fri: 9am - 5pm, Sat: 9am - 1pm
Website: Online support

Computer name, domain, and workgroup settings
Computer name: [illegible]
Change settings

C.3 Modifying Environment Variables in Windows

Select ‘Advanced’ tab and then ‘Environment Variable’


System Properties

System Restore | Automatic Updates | Remote
General | Computer Name | Hardware | Advanced

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings

User Profiles
Desktop settings related to your logon
Settings

Startup and Recovery
System startup, system failure, and debugging information
Settings

Environment Variables | Error Reporting

OK | Cancel | Apply

.

APPENDIX D Additional FLEXlm Licensing Information

D.1 Locating the license server software

The license server and daemon are provided as part of the OVPsim or Imperas releases or may be downloaded from the Imperas and OVPWorld websites.

After installing the OVPsim or Imperas packages the license server, lmgrd, and the Imperas license daemon, imperasd, can be found in the binary installation directory.

On a Windows machine the server will be found

```
$IMPERAS_HOME/bin/$IMPERAS_ARCH/lmgrd.exe
```

On a Linux machine the server will be found

```
$IMPERAS_HOME/bin/$IMPERAS_ARCH/lmgrd
```

The deamon, imperasd, is found in the same directory.

D.2 Types of Licenses

There are several different types of license files. A license can be node-locked, which means the tools can only run on a single computer, or it can be *floating* in which case one computer acts as a license server, which runs a license daemon, and the tools may run on any computer that can contact the server over the local area network to check out a license.

In the case of a node-locked license, you will send Imperas/OVPworld the host name and host ID of the computer that you will run the tools on. In the case of a floating license you will need to choose one computer to be the license server and send the host name and host ID of that computer to Imperas/OVPworld to obtain your license. Note the license server may be the same computer you run the tools on.

In addition a license may be *uncounted*, which means that any number of copies of the program may run simultaneously, or it may be *counted* which means that only a specified number of instances of the program may be active at one time.

D.2.1 Uncounted Node-locked Licenses

An uncounted node-locked license is the simplest sort of license. It does not require a license server or a license daemon to be running. You can tell your license is node-locked if it has only uncounted FEATURE lines and comment lines (which start with a #) in it. For example:

```
#### OVPsim non-commercial license keys
#### Generated on 2011-04-30
#### Hostname1 PC1
FEATURE IMP_OVPSIM_20130630 imperasd 1.0 25-aug-2013 uncounted \
        HOSTID=002119426114 SIGN="0874 1401 CA64 337B FC3B 9CE6 9D29 \
        CFD3 F89E F4C1 D090 9084 CE9A E2A8 D8BA 0F6F BEF9 CC19 ACF4 \
        A7F9 6834 145A 0B11 0BED AD50 6672 47B2 1758 7B24 65D2"
```

You can tell if a license feature is uncounted by looking for the word *uncounted* following the expiration date of the feature.

If a license contains only node-locked, uncounted features then you only need to set the environment variable *IMPERASD_LICENSE_FILE* to point at it, e.g.:

```
IMPERASD_LICENSE_FILE=~/Imperas/license.lic
```

See the respective installation instructions for Windows and Linux for examples of how to set an environment variable.

D.2.2 Floating Licenses

If your license is a floating license then you will need a license daemon program running on a license server. With a floating license any computer that can find the license server on the local area network can run the tools.

An example of a floating license looks like:

```
SERVER server1 00F346829930
VENDOR imperasd /path-to-deamon/imperasd
USE_SERVER
################## Feature List ####################
FEATURE IMP_SIMULATOR imperasd 1.0 25-aug-2013 5 TS_OK SIGN="5763\
        07C3 196A 858A 5455 A9EF 541E D2D4 6A75 1ED5 B3BF AE94 F141 \
        1A7A BE50 CD76 3466 A503 3035 4464 9788 4911 AEC5 F73B 18E7 \
        92AC 27CF B8A4 9FF3 0DE4 "
FEATURE IMP_CPUMANAGER imperasd 1.0 25-aug-2013 5 TS_OK SIGN="10D9 \
        9297 4903 02EE 90C9 95BC 3371 9B72 11D4 F9E2 056F BBF2 AA5F \
        4DBE 681B 0193 A355 B1B2 1638 6BBF 01CD A006 5E73 56C1 FD5D \
        EAB6 B2E0 70C3 893F 9755"
FEATURE IMP_IGEN imperasd 1.0 25-aug-2013 5 TS_OK SIGN="0D60 4B55 \
        ABB6 7914 4031 B18D F7D0 30E7 216A ACE8 B0A2 9C8C A456 55CF \
        EC6D 134B 0A01 069D CB14 268E 1822 FAD7 CA44 CBAA 193F 63D4 \
        C8B9 D7C9 3227 68EA"
```

Here we see SERVER, VENDOR and USE_SERVER lines in addition to the FEATURE lines we saw in an uncounted node-locked license. Also, after the expiration date on the FEATURE lines we see ‘*5*’ instead of ‘*uncounted’* indicating this is a counted license. When either of these is true a license server will be required.

The format of the SERVER line is as follows:

```
SERVER <host name> <host ID> {<port>}
```

`<host name>` should be the host name of the license server. This can be edited if it is not

correct.
`<host ID>` is the host id of the license server. This cannot be changed without invalidating the license file. The license server must be the computer with this host ID.
`{<port>}` is an optional integer value which specifies the port the server communicates through. It should usually be left blank, in which case a port in the default range of 27000 to 27009 will be used. If it is specified in the license file then the same value must be specified on `the IMPERASD_LICENSE_FILE` environment variable on the computer running the target program.

The format of the VENDOR line is as follows:

```
VENDOR <vendor_daemon_name> <path to vendor daemon file>
```

`<vendor_daemon_name>` should be **`imperasd. <path to vendor daemon file>`** should be edited to be the path of the *imperasd* (*imperasd.exe* on Windows) file located in the directory `$IMPERAS_HOME/bin/$IMPERAS_ARCH` for example:

```
VENDOR imperasd /usr/Imperas/bin/Linux32/imperasd
```
or
```
VENDOR imperasd C:\Imperas\bin\Windows32\imperasd.exe
```

D.3 Configuring the License Server

The Imperas license daemon (imperasd) must be run on the license server using the FLEXlm tools. The FLEXlm `lmutil` tool and the Imperas daemon are provided in the `$IMPERAS_HOME/bin/$IMPERAS_ARCH` directory. The daemon can be started using the following command on the license server:

On a Windows machine in an MSYS shell

```
> $IMPERAS_HOME/bin/$IMPERAS_ARCH/lmgrd.exe –c license.lic –l license.log
```

On a Linux machine in a shell

```
> $IMPERAS_HOME/bin/$IMPERAS_ARCH/lmgrd –c license.lic –l license.log
```

Where `license.lic` is the name of the license file which must be able to be read from the license server and `license.log` is the name of a file where the license manager will write informational and debug messages.

The license server may also be started in a foreground mode by adding the **-z** switch

```
> $IMPERAS_HOME/bin/$IMPERAS_ARCH/lmgrd –z –c license.lic
```

If the license server is a different computer than you installed the tools on, you may want to copy the license manager tools and the daemon file from

`$IMPERAS_HOME/bin/$IMPERAS_ARCH` to the license server. The license tools all start with "`lm`" and the daemon file is named either `imperasd` or `imperasd.exe`.

The *lmgrd* command will need to be re-run every time the license server is rebooted.

On Linux it can be placed in a boot script file (e.g. /etc/rc.local).

On Windows it can be started as a service each time the computer is rebooted. The command `lmtools` will start a GUI program where you can configure the windows lmgrd service from the `Config Services` tab.

D.4 Configuring the Host Computer

On the computer that will be running the tools you simply have to set the IMPERASD_LICENSE_FILE environment variable to point to the license server,

```
IMPERASD_LICENSE_FILE=@serverhost
```

where **`serverhost`** is the hostname of the license server. The command *hostname* may be used on both Windows and Linux to find out the host name of a particular computer. This example assumes that the default port (27000-27009) is being used by the license server. If the license server was set up to use a different port it must be specified before the '@', for example:

```
IMPERASD_LICENSE_FILE=9999@serverhost
```

Here, **`9999`** is the port number that the license server is using.

See the respective installation instructions for Windows and Linux for examples of how to set an environment variable.

D.5 Other License File Configurations

FLEXlm has many different ways to configure it. You may already have other FLEXlm licenses running on your computers. By default we have configured things to be as independent of other installations as possible, but your particular situation may require alternative configurations.

Also, FLEXlm supports many different options to control how licenses are managed than are discussed her. If you need to use any of those additional capabilities please consult the FLEXlm End Users Manual.

Searching the internet for "FlexLM End Users Manual" should locate a copy of this manual.

D.6 The daemon options file

Use of the licenses can be customized by using a daemon options file.

To make FLEXlm use an options file, either specify the full path to the file in the license file or name the file `'imperasd.opt'` and place it in the same directory as the license file.

Each line of the options file controls one option. The most commonly used options are:

EXCLUDE feature[:keyword=value] type {name | group_name}

Exclude a user or host from using a license feature.

where featurename is the name of the feature as specified in the license file, type is normally USER, HOST or GROUP, and name is the name of the user or group to exclude.

INCLUDE feature[:keyword=value] type {name | group_name}

This option is the same as EXCLUDE but allows a user to use a license feature, but note, if any INCLUDE lines are used then one must be provided for each user who will use the license.

GROUP group_name user_list

Defines a name (group_name) for a group of users, so it can be used in other options like include & exclude. username list is just a list of names separated by a space.

RESERVE num_lic feature[:keyword=value] type {name | group_name}

Reserve licenses for a specific user, group or host.

where `num_lic` is the number of licenses to be reserved, and the other variables are the same as for EXCLUDE.

For a list of other options please consult the FLEXlm End Users Manual.

Searching the internet for “FlexLM End Users Manual” should locate a copy of this manual.

D.7 License Administration tools

There are several utilities that aid administration of the licenses and this section briefly describes the most frequently used.

`lmutil lmstat` - allows users to see:-

- which daemons are running
- which users are using which features

Options:
`-a` - Display all information.
`-A` - List all active licenses.
`-c` - Use the given license file.
`-f` - List all users of featurename.
`-S` - List all users of a vendor's features (for example, `arcd`)
`-s` - Display status of clients running on given host.

`lmutil lmdown` - gracefully shutdown all daemons.

Use -c to shutdown a specific license. Providing no arguments shuts down the license(s) specified by the environment variable `LM_LICENSE_FILE`. Use of `lmdown` should be protected to prevent unintentional loss of licenses.

`lmutil lmremove` - most commonly used to remove licenses that are still checked out after a node crashes.

`lmremove` will remove all instances of the specified feature by the given user on host.

`lmutil lmreread` - make the license daemon reread the given license file.

This will start any new daemons that have been added and cause currently running daemons to read their license files again.

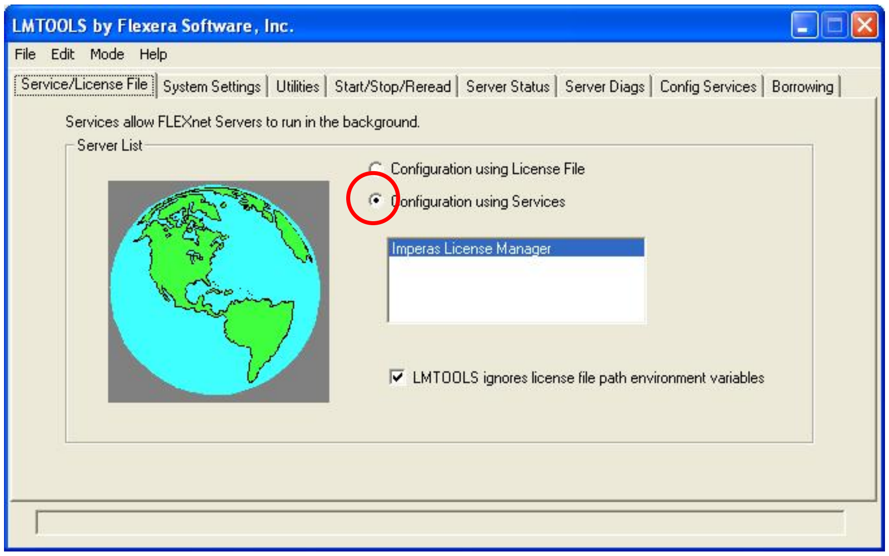
Use either `lmutil lmreread imperasd` or `lmutil lmreread –c <licensefile>`.

D.8 Configuring the License Server Manager as a Windows Service using LMTOOLS

To configure a license server manager (lmgrd) as a service, you must have Administrator privileges. The service will run under the LocalSystem account. This account is required to run this utility as a service.

To configure a license server as a service:
1. Run the **lmtools** utility, this is found in Imperas/bin/Windows32

2. Click the **Configuration using Services** button, and then click the **Config Services** tab.



LMTOOLS by Flexera Software, Inc.

File Edit Mode Help

Service/License File | System Settings | Utilities | Start/Stop/Reread | Server Status | Server Diags | Config Services | Borrowing

Services allow FLEXnet Servers to run in the background.

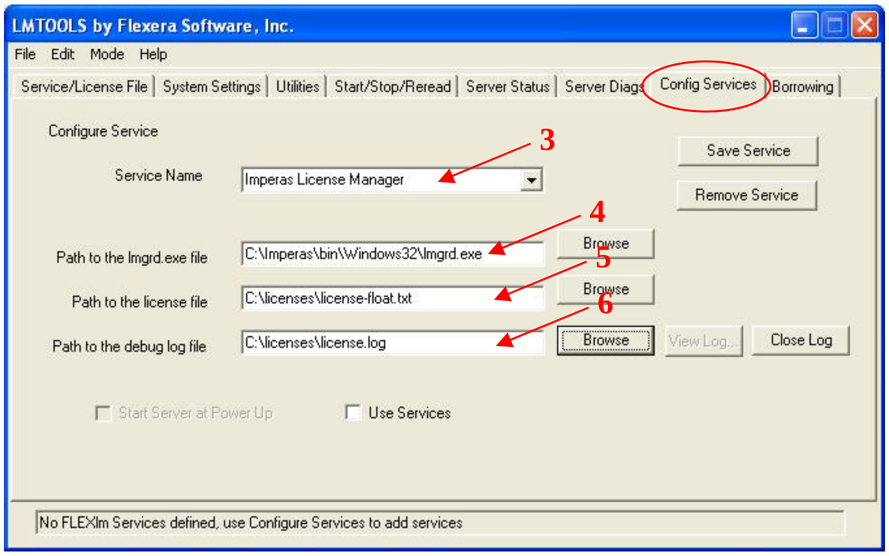
Server List

Configuration using License File [✓ Configuration using Services]

Imperas License Manager

☑ LMTOOLS ignores license file path environment variables

3. In the **Service Name**, type the name of the service that you want to define, for example, *Imperas License Manager*. If you leave this field blank, the service will be given a default name.
4. In the **Path to the lmgrd.exe** file field, enter or browse to lmgrd.exe for this license server.
5. In the **Path to the license file** field, enter or browse to the license file for this license server.
6. In the **Path to the debug log file**, enter or browse to the debug log file that this license server writes.



LMTOOLS by Flexera Software, Inc.

File Edit Mode Help

Service/License File | System Settings | Utilities | Start/Stop/Reread | Server Status | Server Diags | Config Services | Borrowing

Configure Service

Service Name: Imperas License Manager

Path to the lmgrd.exe file: C:\Imperas\bin\Windows32\lmgrd.exe

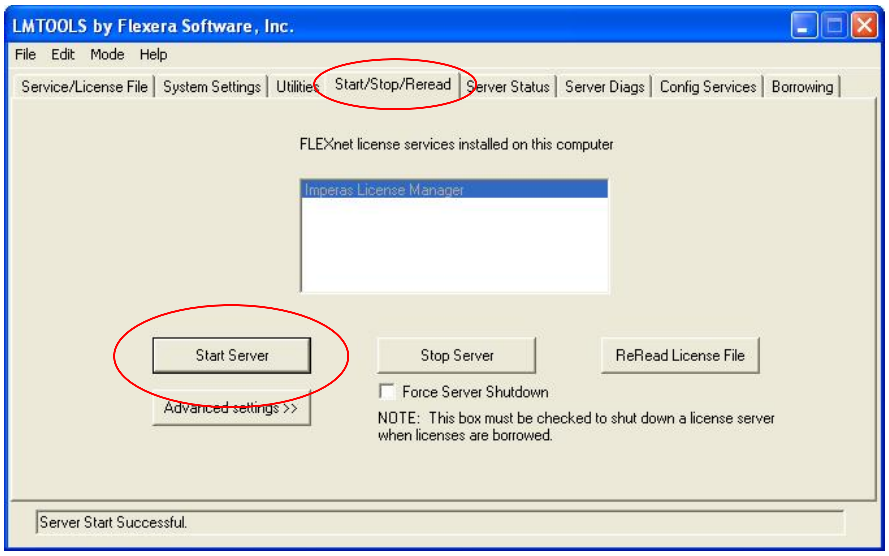
Path to the license file: C:\licenses\license-float.txt

Path to the debug log file: C:\licenses\license.log

☐ Start Server at Power Up ☐ Use Services

No FLEXlm Services defined, use Configure Services to add services

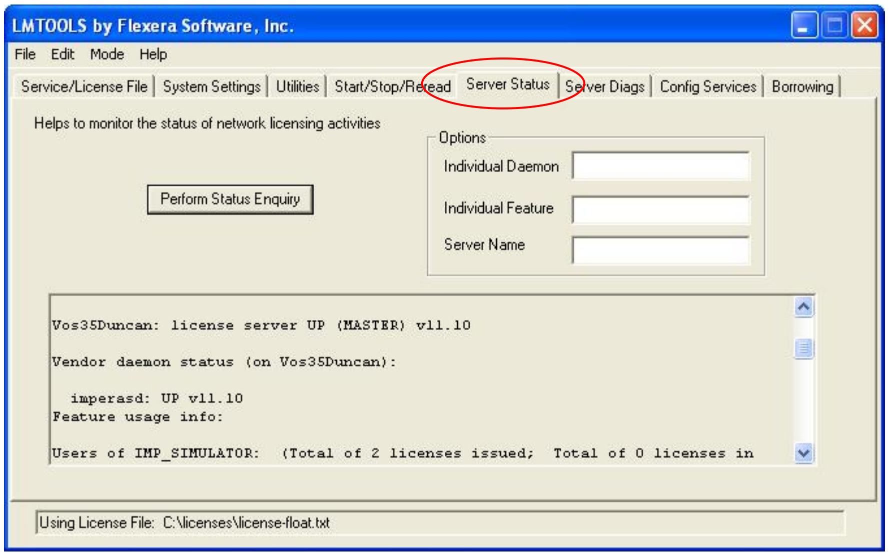
7. Start the license server


LMTOOLS by Flexera Software, Inc.
File Edit Mode Help
Service/License File | System Settings | Utilities | Start/Stop/Reread | Server Status | Server Diags | Config Services | Borrowing
FLEXnet license services installed on this computer
Imperas License Manager
Start Server
Stop Server
ReRead License File
Advanced settings >>
Force Server Shutdown

NOTE: This box must be checked to shut down a license server when licenses are borrowed.

Server Start Successful.

8. Check the license server status


LMTOOLS by Flexera Software, Inc.
File Edit Mode Help
Service/License File | System Settings | Utilities | Start/Stop/Reread | Server Status | Server Diags | Config Services | Borrowing
Helps to monitor the status of network licensing activities
Perform Status Enquiry
Options
Individual Daemon
Individual Feature
Server Name

```
Vos35Duncan: license server UP (MASTER) v11.10

Vendor daemon status (on Vos35Duncan):

  imperasd: UP v11.10
Feature usage info:

Users of IMP_SIMULATOR:  (Total of 2 licenses issued;  Total of 0 licenses in
```

Using License File: C:\licenses\license-float.txt

APPENDIX E Some Common Problems

E.1 Segmentation Fault when Native Debug of an OVP Platform

When an ICM C platform is being debugged the simulation may stop with the message

"Program received signal SIGSEGV, Segmentation fault."

This will have occurred in the call to icmInitplatform() when the licensing is initialized. The latest version of the FlexLM licensing code utilized by the OVP and Imperas products contains a feature that causes a segmentation fault when attempting to run under a debugger.

This has no effect on the execution of the OVP virtual platform and it is possible to continue the simulation and debugging the platform beyond this point. Simply continue the simulation.

The following is the contents of an example .gdbinint script that may be used to ignore this issue

```
handle SIGSEGV nostop
handle SIGSEGV noprint
```

The OVP and Imperas simulators also may use of the floating point exceptions during simulation, these may be ignored when debugging by adding the following

```
handle SIGFPE nostop
handle SIGFPE noprint
```

Finally, additional options that we have found useful are

```
set break pending on

set backtrace limit 100
```

E.2 Simulator Reports Internal Abort (ASRT)

The simulator will only report an assertion when it has detected that the internal state is no longer valid. This can be caused by a bug in the simulator, which should be reported to

Imperas including a test case[17] that can be used to reproduce the problem at the Imperas site.

However, often these problems are caused when the simulator is being used incorrectly.

Asynchronous Calls to a running simulator
One such way is by making asynchronous calls to a running simulator. There must be no asynchronous calls made to the running simulator. If the simulation is to be interrupted the procedure that is described in the 'OVPsim and CpuManager User Guide' must be followed.

If this procedure is not followed the asynchronous calls can cause the simulator to be interrupted during a system state update and hence the state can become corrupted.

When this occurs it is typical to see the following assertions reported by the simulator and for the simulation to terminate.

```
> Internal Abort (ASRT)
>
/home/release/build/20130630.6/Imperas/SimCommon/source/morph/codeBlock.c:1407
> : Assertion failure : bad native address bounds

> Internal Abort (ASRT)
> /home/release/build/20130630.6/Imperas/SimCommon/source/morph/xRef.c:690
> : Assertion failure : resolved cross-reference found
```

E.3 Building on Windows using mingw32-make

E.3.1 Error in Makefile.pse when building peripherals

There is a known bug in the version of GNU make 3.80. This version is sometimes installed as part of the MinGW installation as mingw32-make and will result in the error shown below when attempting to build a peripheral model

```
/c/Imperas/Examples/Models/Peripherals/creatingDMAC/1.registers
$ mingw32-make NOVLNV=1
C:/Imperas/ImperasLib/buildutils/Makefile.pse:39: *** missing `endif'.  Stop.

/c/Imperas/Examples/Models/Peripherals/creatingDMAC/1.registers
$
```

You can check the version of mingw32-make that you have installed with the command "mingw32-make --version"

[17] A test case should include the virtual platform source code, the application code being executed and clear instructions of how to reproduce the problem. All these files should be sent as a zip or tar file. It is useful to extract the files and check in a clean directory that the instructions sent can re-produce the problem.

```
$ mingw32-make --version
GNU Make 3.80
Copyright (C) 2002  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

$
```

E.4 Licensing

E.4.1 feature not supported

If you get an error something like the following:

```
Error (LIC_NE) License not available. FLEXLM reports:

License server system does not support this feature.
Feature:       IMP_OVPSIM
License path:  C:\Imperas\bin\Windows\OVPsim.lic;
FLEXnet Licensing error:-18,147
For further information, refer to the FLEXnet Licensing End User Guide,
available at "www.macrovision.com".


License server system does not support this feature.
Feature:       IMP_OVPSIM_20130630
License path:  C:\Imperas\bin\Windows\OVPsim.lic;
FLEXnet Licensing error:-18,147
For further information, refer to the FLEXnet Licensing End User Guide,
available at "www.macrovision.com".

Fatal (LIC_FE) If you have a license from OVPworld, put it in
$IMPERAS_HOME/OVPsim.lic
Alternatively, set IMPERASD_LICENSE_FILE environment variable to your
license file or server location.
If you don't have a key, click on download at www.OVPworld.org
Info Exiting
```

You have not got the correct license FEATURE available in your license file.

This could be caused by one of the following:

1. The license file has not been obtained
    a. Check the license file does include this FEATURE
2. The incorrect license file is being used or the license file has been corrupted
    a. Check the IMPERASD_LICENSE_FILE variable is pointing to the correct license file or license server for floating licenses
    b. Check the license file has not been corrupted. Use a hex editor to ensure only valid ASCII characters and <CR>, <LF> are present
3. The incorrect simulator is being used
    a. The incorrect IMPERAS_RUNTIME has been set.
       If your license(s) are supplied by Imperas for the Imperas tools, the

IMPERAS_RUNTIME should be set to ***CpuManager*** so that the correct Imperas licenses are used.

4. The simulator personality is not set correctly
    a. The correct runtime and IMPERAS_PERSONALITY has been set. The CpuManager simulator can be set to support different feature sets using the IMPERAS_PERSONALITY environment variable. This also means that different license features are required. See section

E.4.2 cannot access date server

If you get an error something like this:

```
Error (LIC_SRVNS) License not available. FLEXLM reports:
Bad Status connecting to internet date server.
Feature: IMP_OVPSIM_20120313
FLEXnet Licensing error:+97,210
Fatal (LIC_SRVNS) Exiting:
Info Exiting
```

For version locked licenses the simulator will access a date server. This error will occur if the simulator is not able to connect to the internet.

You may get this if you are using an internet proxy server. In this case you need to inform the Imperas products of the proxy server using the IMPERAS_PROXY_SERVER environment variable. This is set to an <address>:<port> pair for the proxy server access.

For example

```
export IMPERAS_PROXY_SERVER=192.168.2.1:1234
```

E.5 Ethernet Adapter Naming/HostId is zero

If you get an error something like this:

```
Invalid host.
The hostid of this system does not match the hostid
specified in the license file.
Feature: IMP_OVPSIM_20100528.0
Hostid: 112233445566
License path: /home/Imperas.20130630/OVPsim.lic
FLEXnet Licensing error:-9,57. System Error: 19 "(null)"
For further information, refer to the FLEXnet Licensing End User Guide,
available at "www.macrovision.com".
```

The problem is caused by the FlexLM licensing used obtaining the MAC address, against which the product is licensed, from the eth0 network interface.

If the network interface is enabled on a different device, e.g. eth1 the FlexLM software does not read a valid MAC address and a NULL host Id is reported. This may be caused because on modern operating systems Consistent Network Device Naming may be used, see the following link for further information
http://fedoraproject.org/wiki/Features/ConsistentNetworkDeviceNaming

The solution is to enable the network interface on eth0 or to rename the network interface to be eth0.

There are a number of ways this can be achieved, the following was taken from
http://www.science.uva.nl/research/air/wiki/LogicalInterfaceNames

E.5.1 How to change the default 'ens33' network device to old 'eth0'

The following URL explains the full procedure

http://unix.stackexchange.com/questions/81834/how-can-i-change-the-default-ens33-network-device-to-old-eth0-on-fedora-19

The minimum set of stages that were required when performed are:

1. Edit **/etc/default/grub**. Note you need root privileges to edit this file.
2. At the end of GRUB_CMDLINE_LINUX line append "**net.ifnames=0 biosdevname=0**"
3. Save the file
4. type the command "**grub2-mkconfig -o /boot/grub2/grub.cfg**"
5. type the command "**reboot**"

E.5.2 How to reorder or rename logical interface names in Linux

One of the problems of Linux is that the order of the network interfaces is unpredictable.

Between reboots it usually stays the same, but often after an upgrade to a new kernel or the addition or replacement of a network card (NIC) the order of all network interfaces changes. For example, what used to be eth0 now becomes eth1 or eth2 or visa versa.

Obviously there is some logic to which network interface gets which name, but Linux documentation states that this may change and no user or program should ever assume anything about this. This is annoying, in particular if your management interface is at eth1 at one node in a cluster and at eth2 in another node of the same cluster (which we have experienced). I personally like to have my (primary) management interface always to be eth0.

Thankfully, there are ways to achieve this. They can be divided in four methods:
1. Order the network interfaces based on physical properties of the NIC. (e.g. the physical location in the machine)
2. Order the network interfaces based on the MAC address of the NIC.
3. Order the network interfaces based on the driver of the NIC.
4. Order the network interfaces based on the physical location of the NIC in the computer

So you have to pick a method that suits you. I recommend either to use ifrename (based on physical properties, especially useful if you often change network cards in your hosts) or writing a udev rule (based on the MAC address). However, I listed the other methods as well. Be aware that the last two methods mentioned in this article are only for the masochistic (you will scream and shoot to get those to work).

Note: Linux kernels up to 2.4 did only probe for the first Ethernet card, ignoring other NICs. We assume you use a 2.6 or higher kernel or already fixed this, for example by specifying ether=0,0,eth1 as kernel parameter∞.

E.5.3 Based on the physical properties

Perhaps the most elegant way to name the Ethernet NIC is to do so based on their physical properties, like link speed and port type.

E.5.3.1 Using the ifrename tool

Ifrename is a tool specifically designed to (re)name network interfaces based on characteristics like MAC address (wildcards supported), bus information, and hardware settings. It uses a control file (/etc/iftab) to specify rules about how the interfaces will be named. (thanks to Matt Baron for this tip.)

```
# Example /etc/iftab file
eth2          mac 08:00:09:DE:82:0E
```

```
eth3           driver wavelan interrupt 15 baseaddress 0x390
eth4           driver pcnet32 businfo 0000:02:05.0
# wildcard name: pick the lowest available name of air0,
air1, air2, etc.
air*           mac 00:07:0E:* arp 1
```

E.5.3.2 Using the ethtool and ip programs

It is possible to check the NIC properties using the ethtool program, and to change the name using the ip program (thanks to Jollynn Schmidt for this tip):

```
if ethtool eth0 | grep -q "Port: FIBRE"; then
        ip link set dev eth0 name not_eth0
        ip link set dev eth1 name eth0
        ip link set dev not_eth0 name eth1
fi
```

The disadvantage of ethtool is that it can only be run by root, even when you're only using it to query for information. Though this is a minor annoyance of ethtool, it does not matter in this case, since you want to set a device name and thus need to be root anyway.

E.5.4 Based on the MAC address

Secondly, it is also possible to name the network interface based on the MAC address of each NIC. The advantage is that it is possible to use this method if you have two NICs which use the same driver (unlike the next method: based on driver).

First, you must determine the MAC address of your interfaces. You can do this locally on a machine running

```
ifconfig -a
```

The MAC address is listed as "hwaddr" (hardware address). Alternatively, you can even determine MAC addresses remotely using ping and /sbin/arp.

There are three ways to map the MAC address to the logical interface name. Either by using the udev rules, with the get-mac-address.sh script, or by using the nameif program.

The udev method should work on all recent Linux distributions, and is recommended. The get-mac-address.sh script and the nameif program are know to work with Debian, while on Red Hat, you can change the interface configuration file.

E.5.4.1 Using udev rules

udev replaced devfs in Linux 2.6. First make sure that your Linux system has udev installed, rather then devfs. If you have a /etc/udev directory, but not /etc/devfs directory, you are probably fine. If not, be aware that changing your kernel from devfs to udev is possible, but is not just a matter of adding a new module. Perhaps for now, another method is easier for you.

Now that you have udev, it is rather simple. You only need to create a udev rule mapping the MAC address to the interface name. Store this in a file inside the/etc/udev/rules.d/ directory:

```
KERNEL=="eth?", SYSFS{address}=="00:37:e9:17:64:af",
NAME="eth0"  # MAC of first NIC in lowercase
KERNEL=="eth?", SYSFS{address}=="00:21:e9:17:64:b5",
NAME="eth1"  # MAC of second NIC in lowercase
```

Most distributions already come with an example config file for you.
E.g. /etc/udev/rules.d/network-devices.rules or /etc/udev/rules.d/010_netinterfaces.rules.
More information can be found
at http://www.reactivated.net/writing_udev_rules.html∞ or http://www.debianhelp.co.uk/udev.htm∞. (Thanks to Casey Scott and Ron Hermsen for the pointers.)

Another rule I've seen is:

```
SUBSYSTEM=="net", DRIVERS=="?*",
ATTRS{address}=="00:16:3e:00:02:00", NAME="eth0"
SUBSYSTEM=="net", DRIVERS=="?*",
ATTRS{address}=="00:16:3e:00:02:01", NAME="eth1"
```

I'm not sure about the difference between these rules. Information is welcome (please leave a comment below)

E.5.4.2 Using the interface configuration file

If you run a Red-Hat-based distribution, you can simply add the MAC address in the interface configuration file /etc/sysconfig/network-scripts/ifcfg-eth0:

```
DEVICE=eth0
HWADDR=00:37:e9:17:64:af
```

You can give it any DEVICE name you want, like DEVICE=ethmgmt, as long as you remember to rename the config file:
/etc/sysconfig/network-scripts/ifcfg-ethmgmt

Source: http://forums.serverwatch.com/showthread.php?t=18476∞

E.5.4.3 Using the get-mac-address.sh script

Another solution is to use the get-mac-address.sh script to map interface names by MAC address. On Debian, this script is distributed as part of the ifupdown package∞(in /usr/share/doc/ifupdown/examples/get-mac-address.sh). Copy this script to a saner place (e.g. /usr/local/bin), and you can setup /etc/network/interfaces in this manner:

```
auto lo eth0 eth1

iface lo inet loopback

mapping eth0 eth1
         script /usr/local/bin/get-mac-address.sh
         map 00:37:E9:17:64:AF netA
         map 00:21:E9:17:64:B5 netB

iface netA inet static
         address etc...

iface netB inet static
         address etc...
```

Source: https://www.gelato.unsw.edu.au/archives/gelato-technical/2004-February/000334.html∞

The disadvantage of this method is that defines a mapping, rather then changing the actual logical interface name.

E.5.4.4 Using the nameif program

Alternative to the get-mac-address.sh script, you can also use the slightly more convenient nameif program, which is distributed as part of the net-tools package∞ on Debian.

The advantage of nameif is that you can specify the interface names in the /etc/mactab file:

```
ethmgnt 00:37:E9:17:64:AF
ethwireless 00:21:E9:17:64:B5
```

It is not possible to rename an interface to a name of an existing interface. So you can't rename eth1 to eth0 as long as eth0 still exists. It is possible to still swap the names eth0 and eth1 by using a temporary name (e.g. first rename eth1 to ethfoo, then eth0 to eth1 and finally ethfoo to eth0). Note that this method may lead to problems if you use common names such as eth0 and eth1. If you upgrade a kernel, the names may be different than you expected, and you may rename a NIC to eth0 while eth0 still exists, leading to name collisions. Therefore, it is recommended to use other names like "ethmgmnt", "ethwired", "ethwireless" and "eth10ge", as shown in the example above.

E.5.5 Based on the driver

Warning: This only works if the driver is available as a loadable module. Not if it is built into the kernel.

This is a relative easy method, since it does not rely on external scripts. The idea is to just load the kernel module for your eth0 interface before the modules for other network cards.

First of all, you must determine which driver is used for each network card. Thankfully Linux does have a system to load the appropriate driver automatically, based on the PCI ID of the network card. Unfortunately, there is no single command to simply get the driver (and other information like the link speed) based on just the interface name in Linux. Your best bet is to look for kernel messages:

```
dmesg | grep eth
```

This should give you a good estimate of the driver name. You can verify if the name indeed does exist and is loaded:

```
lsmod
```

Note:
lsmod gave:

```
e1000           84868  0
tg3             70816  0
```

However, the 0 indicates that these drivers are not controlling any device! That is strange, since modprobe -r tg3 and modprobe -r e1000 do disable the network cards. Apparently, this is a flaw in lsmod.

Note that running modprobe tg3 en then modprobe e1000 does bring them up in the correct order, with the correct interface names. This is a good check if this approach (using the driver to decide the interface name) can work.

E.5.5.5 Red Hat

In Red Hat, if the driver is called "tg3" (the Tigon driver), you simply specify the network name by adding this entry in /etc/modules.conf:

```
~alias eth0 tg3
```

E.5.5.1 Debian

On a Debian system, /etc/modules.conf is generated automatically and should not be edited directly. Instead, create a file in the subdirectory /etc/modules/ (do not use /etc/modprobe.d/, that seems out-of-date). For example, create the file /etc/modutils/interfaces and add the appropriate modules. For example:

```
alias eth0 tg3
alias eth1 e1000
```

Next, update /etc/modules.conf by running:

```
update-modules
```

Alternative method: I have encountered scenario's where the kernel did already load the modules for the drivers, even before /etc/modules.conf was read. The result was that in effect, the specification in /etc/modules.conf was ignored, and this method did not work. As an alternative, it is possible to also list the drivers, in the appropriate order, in /etc/modules (thus not /etc/modules.conf):

```
tg3
e1000
```

The result will be that the tg3 driver is loaded before the e1000 kernel.
Since /etc/modules only exists for Debian, this trick will most likely not work for other distributions.

E.5.6 Based on the physical location in the computer

Warning: This only works if the driver is built into the kernel, not as a loadable module.

Note: It is relatively hard to get this to work, and we encountered problems with it. The other methods are recommended.

It is possible to name the network interfaces based on the interrupt (IRQ) and memory address. This should work if you have network cards in PCI busses, and it involves appending the proper parameters to the "ether=" or "netdev=" kernel parameters.

First of all, you can detect the PCI slot of the devices using

```
lspci -v
```

This is reported to fail sometimes for certain cards. Now, write down the IRQ and IO (memory) address of each network card, and use this information to specify the interface name in your LILO or GRUB configuration file.

For LILO, you need to add an add line to the appropriate boot configuration. For example:

```
append="netdev=irq=21,io=0x2040,name=eth0

netdev=irq=20,io=0x3000,name=eth1
netdev=irq18,io=0x2000,name=eth2"
```

Under grub, it can just be listed as parameter. e.g.:

```
kernel /boot/vmlinuz netdev=irq=24,name=eth0
```

E.5.7 Ubuntu directly changing logical names

The following information is taken from the page found at
https://help.ubuntu.com/10.04/serverguide/network-configuration.html

The Ethernet interface logical names are configured in the file

```
/etc/udev/rules.d/70-persistent-net.rules
```

If you would like control which interface receives a particular logical name, find the line matching the interfaces physical MAC address and modify the value of NAME=ethX to the desired logical name. Reboot the system to commit your changes.

E.5.8 More Information

http://www.tldp.org/HOWTO/Ethernet-HOWTO-8.html∞

http://www.tldp.org/HOWTO/BootPrompt-HOWTO-11.html∞

Written by Freek Dijkstra. Licensed under public domain. (That is, feel free to modify, redistribute, cripple, or even sell it as your own work, and there is no need to mention the source, even though you are of course welcome to do so.)

E.6 Cannot run license server (lmgrd) on Ubuntu

When you try and run the FlexLM license on Ubuntu you get an error

```
./lmgrd: No such file or directory
```

The FlexLM license manager daemon is not officially supported on Ubuntu, it is supported on the RedHat Enterprise Linux distribution. It can, however, be used on the Ubuntu Linux operating system.

For Ubuntu the LSB support package is required, if it is not already present.

This may be obtained using the command

```
sudo apt-get install lsb
```

APPENDIX F Information on Installation and Makefiles

F.1 Introduction

Windows and Unix present a number of challenges to engineers in the different ways file structure paths are referenced, in particular the '\' (backslash) versus '/' (slash) differences, and the differences regarding the use of a ' ' (space) in a name, in Unix a space is generally treated as a separator, whereas on Windows it can form part of a filename without the need to be escaped.
In order to help overcome these issues a number of utilities, and included Makefiles have been created, which will help in the building of Platforms, Processors, Peripherals and Applications.

Take a look at the file Makefile.app, in the previous example 'Examples/HelloWorld'. The first line of the Makefile.app is thus :-

```
IMPERAS_HOME :=$(shell getpath.exe “$(IMPERAS_HOME)”)
```

This line will modify the windows long filename format, to the short filename format, so that in the Makefile, the default value of IMPERAS_HOME is modified from :-

```
IMPERAS_HOME = C:\Program Files\Imperas
```
To
```
IMPERAS_HOME = C:/PROGRA~1/Imperas
```

Thus removing the space separator, which would cause a problem for the 'make' program.

F.2 $(IMPERAS_HOME)/bin/Makefile.include

The second line of Makefile.app, includes some pre-defined variables, from an included makefile :-

```
include $(IMPERAS_HOME)/bin/Makefile.include
```
This included makefile, will provide the following variables to 'make' :-

```
IMPERAS_LIB
```
This variable will point to the Imperas installation lib directory

```
IMPERAS_BIN
```
This variable will point to the Imperas installation bin directory

```
IMPERAS_VMISTUBS
```
This variable will point to a link library, required whilst producing a shared library for a processor/semihosting library model.

```
IMPERAS_VMIINC
```
This variable will point to the ICM header files.

```
SIM_CFLAGS
```
This variable will contain a set of useful flags for compiling host specific code, including setting paths for the VMI include files

```
SIM_LDFLAGS
```
This variable will contain a set of useful flags for linking host specific code, including setting paths to runtime libraries (e.g., libOVPsim)

F.3 CrossCompiler Makefile includes

Each of the CrossCompilers which are shipped as packages, will install a file which can be included into a Makefile intended to help cross compile applications to the targeted processor.

Take a look at the file Makefile.app, in the previous example 'Examples/HelloWorld'. The fifth & sixth lines of the Makefile.app is thus :-

```
CROSS?=OR1K
-include $(IMPERAS_LIB)/CrossCompiler/$(CROSS).makefile.include
```

This inclusion provides a number of variables including IMPERAS_CC, IMPERAS_LD, which can be seen in usage in the file Makefile.app.

F.4 Component Library Makefiles

There are a number of Makefiles provided in the directory ImperasLib/buildutils to aid in the building of components in the VLNV library.

For individual components, these are:

Makefile.platform	: for building an executable from an ICM platform.c file
Makefile.pse	: for building a 'pse.pse' peripheral model shared object
Makefile.host	: for building native host model shared objects , such as processor models or intercept libraries.
Makefile.module	: for building hierarchical components as native host shared objects. These could be combinations of processor and peripheral models and include other modules.

For building the complete library of components, these are:

Makefile.library	: this may be used to build the entire VLNV component library
Makefile.igen	: uses the Imperas productivity tool 'igen' to generate peripheral model source code. Used by Makefile.library.

F.4.1 Makefile.platform

Include this in the Makefile found in a platform source directory

```
IMPERAS_HOME := $(shell getpath.exe "$(IMPERAS_HOME)")
include $(IMPERAS_HOME)/ImperasLib/buildutils/Makefile.platform
```

F.4.2 Makefile.pse

Include this in the Makefile found in a peripheral model pse directory

```
IMPERAS_HOME := $(shell getpath.exe "$(IMPERAS_HOME)")
include $(IMPERAS_HOME)/ImperasLib/buildutils/Makefile.pse
```

F.4.3 Makefile.host

Include this in the Makefile found in a model directory and used for compiling models, intercept (including semihosting) libraries and MMCs to native host code shared objects.

```
IMPERAS_HOME := $(shell getpath.exe "$(IMPERAS_HOME)")
include $(IMPERAS_HOME)/ImperasLib/buildutils/Makefile.host
```

F.4.4 Makefile.module

Include this in the Makefile found in a module directory and used for compiling modules to native host code shared objects.

```
IMPERAS_HOME := $(shell getpath.exe "$(IMPERAS_HOME)")
include $(IMPERAS_HOME)/ImperasLib/buildutils/Makefile.module
```

F.4.5 Makefile.library

Used with VLNVSRC and VLNVROOT to specify the root of the source directory and the root of the output binary VLNV library directory respectively.

```
> make -f ImperasLib/buildutils/Makefile.library \
         VLNVSRC=$IMPERAS_HOME/ImperasLib/source \
         VLNVROOT=$IMPERAS_VLNV
```

It is recommended to run this from the VLNVSRC directory, this can be achieved using the following

```
> make -C $IMPERAS_HOME/ImperasLib/source \
       -f ImperasLib/buildutils/Makefile.library \
         VLNVSRC=$IMPERAS_HOME/ImperasLib/source \
         VLNVROOT=$IMPERAS_VLNV
```

F.4.6 SystemC TLM2 Platforms Makefile.TLM.platform

Used within a SystemC TLM platform source directory to invoke the building of a local platform cpp file.

```
> make -f $IMPERAS_HOME/ImperasLib/buildutils/Makefile.TLM.platform \
        PLATFORM=ArmIntegratorCP_tlm2.0
```

F.5 Selecting An Alternative Component Library

By default the Imperas VLNV library is selected using the IMPERAS_VLNV environment variable, such as shown below

```
> export IMPERAS_VLNV=$IMPERAS_HOME/lib/$IMPERAS_ARCH/ImperasLib
```

This allows all components within this library to be found and used in platforms.

However, when you are creating custom components, they may be created using separate source and binary libraries which are built using the Makefile described in the previous section.

The IMPERAS_VLNV environment variable is a list of libraries to be searched. The list uses a colon (':') as the separator on Linux and a semi-colon (';') as the separator on Windows.

Thus if a new library, '$THISHOME/myLib', is created with custom components we can add this to the IMPERAS_VLNV search path,

For Linux, in a shell or shell script

```
> export IMPERAS_VLNV=$IMPERAS_VLNV:$THISHOME/myLib
```

And, for Windows, in a batch file

```
> set IMPERAS_VLNV=%IMPERAS_VLNV%;%THISHOME%/myLib
```

And, for Windows, in an MSYS/MINGW shell
Note that a semi-colon is used as the separator but it is also required to quote the complete string so that it is not pre-processed by Windows but passed complete into the OVPsim or Imperas product.

```
> export IMPERAS_VLNV=”$IMPERAS_VLNV;$THISHOME/myLib”
```

APPENDIX G Abort Detected in Program

The Imperas and OVP products contain a function to trap aborts. This may trap aborts caused anywhere in the execution flow i.e. it may not be specifically within the Imperas product but in an application using or a user shared object loaded by the Imperas product.

As the Imperas/OVP simulators load shared objects for processor models, peripheral models and intercept (including semihost) libraries it is possible that a segmentation fault or abort can be caused in the loaded code.

This may appear to be a fault of the simulator but this is unlikely. It is more likely to be a fault in one of the models or platforms that are loaded by the simulator. The following can be used to help you find the problem in the software that you have created.

An abort will be reported in the following manner. For an example, an abort has been caused in the virtual platform when run using the command line

```
OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe dhrystone.ARM_CORTEX_A15.elf
```

The abort is reported during the run as follows

```
             SystemC 2.3.0-ASI --- Jun  6 2013 16:33:50
        Copyright (c) 1996-2012 by all Contributors,
        ALL RIGHTS RESERVED



CpuManager (32-Bit) v99999999 Open Virtual Platform simulator from
www.IMPERAS.com.
Copyright (c) 2005-2013 Imperas Software Ltd.  Contains Imperas Proprietary
Information.
Licensed Software, All Rights Reserved.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

CpuManager started: Mon Jun 24 16:28:41 2013

<snip>

Internal Abort (ABRT) Imperas/Common/source/targetOS/tosSignal.c:656 : Abort
reached.
Uncaught Exception (SIGSEGV) at 0x5b5d30

This could be due to an error in your native code (for example the platform,
semihost library etc.) or an error in the simulator.
Use the following approaches to try to find the source of the error:
1. Rerun the simulation under a debugger.
2. Set environment variable IMPERAS_BACKTRACE=1 to generate a backtrace (Linux
only).
3. Set environment variable IMPERAS_LOOP_ON_EXCEPTION=1 to cause the simulator
to enter a wait loop on exception, so that a debugger can be attached.
If you believe the error originates within the simulator, please contact Imperas
support (support@imperas.com)
Info Exiting
```

To aid in tracking down the cause of the abort there are two features, explained in the abort output

G.1 Obtaining Backtrace

A backtrace can be generated, on Linux only, which may help you determine if the error originates in the code that you have added. The following shows the same platform run with the IMPERAS_BACKTRACE=1 set on the command line, i.e.

```
IMPERAS_BACKTRACE=1 OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe
dhrystone.ARM_CORTEX_A15.elf
```

And this results in the following extra information

```
*** backtrace 19 stack frames:
-------------------------------
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x8923b) [0x2e523b]
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x892e5) [0x2e52e5]
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x8932a) [0x2e532a]
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x8930c) [0x2e530c]
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x1db4eb) [0x4374eb]
/home/graham/Imperas/bin/Linux32/libCpuManager.so(+0x1db64b) [0x43764b]
[0xf1e40c]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(bindingEv+0x49) [0x18ff19]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(elab_doneEv+0x1f) [0x19012f]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(elabEv+0x14c) [0x174bfc]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(initEb+0x30) [0x176b60]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(_7scimeE+0x2d) [0x176c2d]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(20sc_stvpolE+0xdc) [0x1778ec]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(_startEv+0x73) [0x177b83]
OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe(sc_main+0x1d5) [0x805109f]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(sc_elab+0xa4) [0x163b94]
/user/systemc-2.3.0/lib-linux/libsystemc-2.3.0.so(main+0x32) [0x163a02]
/lib/libc.so.6(__libc_start_main+0xe6) [0xad1bb6]
OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe(__gxx_person_v0+0x159) [0x8050841]

Internal Abort (ABRT) Imperas/Common/source/targetOS/tosSignal.c:630 : Abort
reached.
Uncaught Exception (SIGSEGV) at 0x18fd30

Info Exiting
```

G.2 Connecting a Debugger to a running simulation

The simulator can be forced to wait at exit for a debugger to be connected.

The following shows the same platform run with the environment variable IMPERAS_LOOP_ON_EXCEPTION=1 set on the command line, i.e.

```
IMPERAS_LOOP_ON_EXCEPTION=1 OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe
dhrystone.ARM_CORTEX_A15.elf
```

This simulation will not finish. In a separate shell you can now start a suitable host debugger and examine the running processes on the host for the one running your platform executable, for example `OVPsim_single_arm_Cortex-A_tlm2.0.Linux32.exe` in this case.

##