Imperas Guide to using Virtual Platforms

Platform / Module Specific Information for atmel.ovpworld.org / AtmelAT91SAM7

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

Author	Imperas Software Limited
Version	20211118.0
Filename	Imperas_Platform_User_Guide_AtmelAT91SAM7.pdf
Created	31 December 2021
Status	OVP Standard Release

Copyright (c) 2021 Imperas Software Limited

Copyright Notice

Copyright 2021 Imperas Software Limited. All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the readers responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Model Release Status

This model is released as part of OVP releases and is included in OVPworld packages. Please visit OVPworld.org.

Table Of Contents

Copyright (c) 2021 Imperas Software Limited

Copyright (c) 2021 Imperas Software Limited

1.0 Platform / Module: AtmelAT91SAM7

This document provides the details of the usage of an Imperas OVP Virtual Platform / Module. The first half of the document covers specifics of this particular component. For more information about Imperas OVP virtual platforms, how they are built and used, please see the later sections in this document.

1.1 Virtual Platform / Module Type

Hardware described using OVP can either be a platform, module, processor, or peripheral.
This component has a purpose specified as being part of an Extendable Platform Kit (EPK). This is typically a platform that is part of a package that includes not only the platform but also software to run on the platform and scripts to control it.

1.2 Description

Simple platform of Atmel AT91 ARM7TDMI based system

1.3 Limitations

This platform is sufficient to boot Linux

1.4 Reference

Rev. 1354D ARM08/02

1.5 Licensing

Open Source Apache 2.0

1.6 Location

The AtmelAT91SAM7 virtual platform / module is located in an Imperas/OVP installation at the VLNV: atmel.ovpworld.org / platform / AtmelAT91SAM7 / 1.0.

2.0 Command Line Control of the Platform

2.1 Built-in Arguments

Table 1. Platform Built-in Arguments

Attribute	Value	Description
allargs	allargs	The Command line parser will accept the complete imperas argument set. Note that this option is ignored in some Imperas products

When running a platform in a Windows or Linux shell several command arguments can be specified.
Typically there is a '-help' command which lists the commands available in the platforms. For example:

```
myplatform.exe -help
```

Some command line arguments require a value to be provided. For example:

myplatform.exe -program myimagefile.elf

2.2 Platform Specific Command Line Arguments

Table 2. Platform Command Line Arguments

Name	Type	Description
kernel	stringvar	The Linux Kernel image e.g. linux
uartport	uns64var	Set the port number to open on the UART and wait for connection, by default a console will be opened.

3.0 Processor [arm.ovpworld.org/processor/arm/1.0] instance: ARM7TDMICore

3.1 Processor model type: 'arm' variant 'ARM7TDMI' definition

Imperas OVP processor models support multiple variants and details of the variants implemented in this model can be found in:
- the Imperas installation located at ImperasLib/source/arm.ovpworld.org/processor/arm/1.0/doc
- the OVP website: OVP_Model_Specific_Information_arm_ARM7TDMI.pdf

3.1.1 Description

ARM Processor Model

3.1.2 Licensing

Usage of binary model under license governing simulator usage.
Note that for models of ARM CPUs the license includes the following terms:
Licensee is granted a non-exclusive, worldwide, non-transferable, revocable licence to:
If no source is being provided to the Licensee: use and copy only (no modifications rights are granted) the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.
If source code is being provided to the Licensee: use, copy and modify the model for the sole purpose of designing, developing, analyzing, debugging, testing, verifying, validating and optimizing software which: (a) (i) is for ARM based systems; and (ii) does not incorporate the ARM Models or any part thereof; and (b) such ARM Models may not be used to emulate an ARM based system to run application software in a production or live environment.
In the case of any Licensee who is either or both an academic or educational institution the purposes shall be limited to internal use.
Except to the extent that such activity is permitted by applicable law, Licensee shall not reverse engineer, decompile, or disassemble this model. If this model was provided to Licensee in Europe, Licensee shall not reverse engineer, decompile or disassemble the Model for the purposes of error correction.
The License agreement does not entitle Licensee to manufacture in silicon any product based on this model.
The License agreement does not entitle Licensee to use this model for evaluating the validity of any ARM patent.

Copyright (c) 2021 Imperas Software Limited

Source of model available under separate Imperas Software License Agreement.

3.1.3 Limitations

Instruction pipelines are not modeled in any way. All instructions are assumed to complete immediately. This means that instruction barrier instructions (e.g. ISB, CP15ISB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. The model does not implement speculative fetch behavior. The branch cache is not modeled.
Caches and write buffers are not modeled in any way. All loads, fetches and stores complete immediately and in order, and are fully synchronous (as if the memory was of Strongly Ordered or Device-nGnRnE type). Data barrier instructions (e.g. DSB, CP15DSB) are treated as NOPs, with the exception of any undefined instruction behavior, which is modeled. Cache manipulation instructions are implemented as NOPs, with the exception of any undefined instruction behavior, which is modeled.
Real-world timing effects are not modeled: all instructions are assumed to complete in a single cycle.

3.1.4 Verification

Models have been extensively tested by Imperas. ARM7TDMI models have been successfully used by customers to simulate ucLinux on Atmel virtual platforms.

3.1.5 Core Features

Thumb instructions are supported.

3.1.6 Debug Mask

It is possible to enable model debug features in various categories. This can be done statically using the "override_debugMask" parameter, or dynamically using the "debugflags" command. Enabled debug features are specified using a bitmask value, as follows:
Value 0x080: enable debugging of all system register accesses.
Value 0x100: enable debugging of all traps of system register accesses.
Value 0x200: enable verbose debugging of other miscellaneous behavior (for example, the reason why a particular instruction is undefined).
All other bits in the debug bitmask are reserved and must not be set to non-zero values.

3.1.7 AArch32 Unpredictable Behavior

Many AArch32 instruction behaviors are described in the ARM ARM as CONSTRAINED UNPREDICTABLE. This section describes how such situations are handled by this model.

3.1.8 Equal Target Registers

Some instructions allow the specification of two target registers (for example, double-width SMULL, or some VMOV variants), and such instructions are CONSTRAINED UNPREDICTABLE if the same target register is specified in both positions. In this model, such instructions are treated as UNDEFINED.

3.1.9 Floating Point Load/Store Multiple Lists

Instructions that load or store a list of floating point registers (e.g. VSTM, VLDM, VPUSH, VPOP) are CONSTRAINED UNPREDICTABLE if either the uppermost register in the specified range is greater than

Copyright (c) 2021 Imperas Software Limited

32 or (for 64-bit registers) if more than 16 registers are specified. In this model, such instructions are treated as UNDEFINED.

3.1.10 Floating Point VLD[2-4]/VST[2-4] Range Overflow

Instructions that load or store a fixed number of floating point registers (e.g. VST2, VLD2) are CONSTRAINED UNPREDICTABLE if the upper register bound exceeds the number of implemented floating point registers. In this model, these instructions load and store using modulo 32 indexing (consistent with AArch64 instructions with similar behavior).

3.1.11 If-Then (IT) Block Constraints

Where the behavior of an instruction in an if-then (IT) block is described as CONSTRAINED UNPREDICTABLE, this model treats that instruction as UNDEFINED.

3.1.12 Use of R13

In architecture variants before ARMv8, use of R13 was described as CONSTRAINED UNPREDICTABLE in many circumstances. From ARMv8, most of these situations are no longer considered unpredictable. This model allows R13 to be used like any other GPR, consistent with the ARMv8 specification.

3.1.13 Use of R15

Use of R15 is described as CONSTRAINED UNPREDICTABLE in many circumstances. This model allows such use to be configured using the parameter "unpredictableR15" as follows:
Value "undefined": any reference to R15 in such a situation is treated as UNDEFINED;
Value "nop": any reference to R15 in such a situation causes the instruction to be treated as a NOP;
Value "raz_wi": any reference to R15 in such a situation causes the instruction to be treated as a RAZ/WI (that is, R15 is read as zero and write-ignored);
Value "execute": any reference to R15 in such a situation is executed using the current value of R15 on read, and writes to R15 are allowed (but are not interworking).
Value "assert": any reference to R15 in such a situation causes the simulation to halt with an assertion message (allowing any such unpredictable uses to be easily identified).
In this variant, the default value of "unpredictableR15" is "execute".

3.1.14 Unpredictable Instructions in Some Modes

Some instructions are described as CONSTRAINED UNPREDICTABLE in some modes only (for example, MSR accessing SPSR is CONSTRAINED UNPREDICTABLE in User and System modes). This model allows such use to be configured using the parameter "unpredictableModal", which can have values "undefined" or "nop". See the previous section for more information about the meaning of these values.
In this variant, the default value of "unpredictableModal" is "nop".

3.1.15 Integration Support

This model implements a number of non-architectural pseudo-registers and other features to facilitate integration.

3.1.16 Halt Reason Introspection

Copyright (c) 2021 Imperas Software Limited

An artifact register HaltReason can be read to determine the reason or reasons that a processor is halted. This register is a bitfield, with the following encoding: bit 0 indicates the processor has executed a wait-for-event (WFE) instruction; bit 1 indicates the processor has executed a wait-for-interrupt (WFI) instruction; and bit 2 indicates the processor is held in reset.

3.1.17 System Register Access Monitor

If parameter "enableSystemMonitorBus" is True, an artifact 32-bit bus "SystemMonitor" is enabled for each PE. Every system register read or write by that PE is then visible as a read or write on this artifact bus, and can therefore be monitored using callbacks installed in the client environment (use opBusReadMonitorAdd/opBusWriteMonitorAdd or icmAddBusReadCallback/icmAddBusWriteCallback, depending on the client API). The format of the address on the bus is as follows:
bits 31:26 - zero
bit 25 - 1 if AArch64 access, 0 if AArch32 access
bit 24 - 1 if non-secure access, 0 if secure access
bits 23:20 - CRm value
bits 19:16 - CRn value
bits 15:12 - op2 value
bits 11:8 - op1 value
bits 7:4 - op0 value (AArch64) or coprocessor number (AArch32)
bits 3:0 - zero
As an example, to view non-secure writes to writes to CNTFRQ_EL0 in AArch64 state, install a write monitor on address range 0x020e0330:0x020e0333.

3.1.18 System Register Implementation

If parameter "enableSystemBus" is True, an artifact 32-bit bus "System" is enabled for each PE. Slave callbacks installed on this bus can be used to implement modified system register behavior (use opBusSlaveNew or icmMapExternalMemory, depending on the client API). The format of the address on the bus is the same as for the system monitor bus, described above.

3.2 Instance Parameters

Several parameters can be specified when a processor is instanced in a platform. For this processor instance 'ARM7TDMICore' it has been instanced with the following parameters:

Table 3. Processor Instance 'ARM7TDMICore' Parameters (Configurations)

Parameter	Value	Description
endian	little	Select processor endian (big or little)
simulateexceptions	simulateexceptions	Causes the processor simulate exceptions instead of halting
mips	100	The nominal MIPS for the processor
semihostvendor	atmel.ovpworld.org	The VLNV vendor name of a Semihost library
semihostlibrary	intercept	The VLNV library name of a Semihost library
semihostname	loader	The VLNV name of a Semihost library

Table 4. Processor Instance 'ARM7TDMICore' Parameters (Attributes)

Copyright (c) 2021 Imperas Software Limited

Parameter Name	Value	Type
variant	ARM7TDMI	enum

3.3 Memory Map for processor 'ARM7TDMICore' bus: 'mainbus'

Processor instance 'ARM7TDMICore' is connected to bus 'mainbus' using master port 'INSTRUCTION'.
Processor instance 'ARM7TDMICore' is connected to bus 'mainbus' using master port 'DATA'.

Table 5. Memory Map ('ARM7TDMICore' / 'mainbus' [width: 32])

Lo Address	Hi Address	Instance	Component
0x0	0xFFFFF	ram0	ram
0x400000	0xFFBFFFFF	extDev	ram
0xFFF00000	0xFFF03FFF	SF	SpecialFunction
0xFFFCC000	0xFFFCFFFF	USART1	UsartInterface
0xFFFD0000	0xFFFD3FFF	USART0	UsartInterface
0xFFFE0000	0xFFFE3FFF	TC	TimerCounter
0xFFFF0000	0xFFFF3FFF	PIO	ParallelIOController
0xFFFF4000	0xFFFF7FFF	PS	PowerSaving
0xFFFF8000	0xFFFFBFFF	WD	WatchdogTimer
0xFFFFF000	0xFFFFFFFF	AIC	AdvancedInterruptController

3.4 Net Connections to processor: 'ARM7TDMICore'

Table 6. Processor Net Connections ('ARM7TDMICore')

Net Port	Net	Instance	Component
fiq	NFIQ	AIC	AdvancedInterruptController
irq	NIRQ	AIC	AdvancedInterruptController

4.0 Peripheral Instances

4.1 Peripheral [atmel.ovpworld.org/peripheral/AdvancedInterruptController/1.0] instance: AIC

4.1.1 Description

AIC: Advanced Interrupt Controller This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux The Advanced Interrupt Controller has an 8-level priority, individually maskable, vectored interrupt controller, and drives the NIRQ and NFIQ pins of the ARM7TDMI from: The external fast interrupt line (FIQ) The three external interrupt request lines (IRQ0 - IRQ2) The interrupt signals from the on-chip peripherals The AIC is extensively programmable offering maximum flexibility, and its vectoring features reduce the real-time overhead in handling interrupts. The AIC also features a spurious vector detection feature, which reduces spurious interrupt handling to a minimum, and a protect mode that facilitates the debug capabilities.

4.1.2 Licensing

Open Source Apache 2.0

Copyright (c) 2021 Imperas Software Limited

4.1.3 Limitations
This model is sufficient to boot Linux

4.1.4 Reference
Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

4.2 Peripheral [atmel.ovpworld.org/peripheral/WatchdogTimer/1.0] instance: WD

4.2.1 Description
WD: Watchdog The Watchdog is built around a 16-bit counter and is used to prevent system lock-up if the software becomes trapped in a deadlock. It can generate an internal reset or interrupt, or assert an active level on the dedicated pin NWDOVF. All programming registers are password-protected to prevent unintentional programming. for more information visit http://www.atmel.com/products/at91

4.2.2 Licensing
Open Source Apache 2.0

4.2.3 Limitations
This model is sufficient to boot Linux

4.2.4 Reference
Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

4.3 Peripheral [atmel.ovpworld.org/peripheral/PowerSaving/1.0] instance: PS

4.3.1 Description
This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux for more information visit http://www.atmel.com/products/at91

4.3.2 Licensing
Open Source Apache 2.0

4.3.3 Limitations
This model is sufficient to boot Linux

4.3.4 Reference
Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

Copyright (c) 2021 Imperas Software Limited

4.4 Peripheral [atmel.ovpworld.org/peripheral/ParallelIOController/1.0] instance: PIO

4.4.1 Description

PIO: Parallel I/O Controller This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux The Parallel I/O Controller has 32 programmable I/O lines. Six pins are dedicated as general-purpose I/O pins. Other I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins. The PIO controller enables generation of an interrupt on input change and insertion of a simple input glitch filter on any of the PIO pins. for more information visit http://www.atmel.com/products/at91

4.4.2 Licensing

Open Source Apache 2.0

4.4.3 Limitations

This model is sufficient to boot Linux

4.4.4 Reference

Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

4.5 Peripheral [atmel.ovpworld.org/peripheral/TimerCounter/1.0] instance: TC

4.5.1 Description

TC: Timer Counter This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux The Timer Counter block includes three identical 16-bit timer counter channels. Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation. The Timer Counter can be used in Capture or Waveform mode, and all three counter channels can be started simultaneously and chained together. for more information visit http://www.atmel.com/products/at91

4.5.2 Licensing

Open Source Apache 2.0

4.5.3 Limitations

This model is sufficient to boot Linux

4.5.4 Reference

Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

Copyright (c) 2021 Imperas Software Limited

4.6 Peripheral [atmel.ovpworld.org/peripheral/UsartInterface/1.0] instance: USART0

4.6.1 Description

USART: Universal Synchronous/Asynchronous Receiver Transmitter This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux. The USART has its own baud rate generator, and two dedicated Peripheral Data Controller. channels. The data format includes a start bit, up to 8 data bits, an optional programmable parity bit and up to 2 stop bits. The USART also features a Receiver Timeout register, facilitating variable length frame support when it is working with the PDC, and a Time-guard register, used when interfacing with slow remote equipment. for more information visit http://www.atmel.com/products/at91

4.6.2 Licensing

Open Source Apache 2.0

4.6.3 Limitations

This model is sufficient to boot Linux

4.6.4 Reference

Rev. 1354D ARM08/02

Table 7. Configuration options (attributes) set for instance 'USART0'

Attribute	Value	Type	Expression
finishOnDisconnect	1	bool	

4.7 Peripheral [atmel.ovpworld.org/peripheral/UsartInterface/1.0] instance: USART1

4.7.1 Description

USART: Universal Synchronous/Asynchronous Receiver Transmitter This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux. The USART has its own baud rate generator, and two dedicated Peripheral Data Controller. channels. The data format includes a start bit, up to 8 data bits, an optional programmable parity bit and up to 2 stop bits. The USART also features a Receiver Timeout register, facilitating variable length frame support when it is working with the PDC, and a Time-guard register, used when interfacing with slow remote equipment. for more information visit http://www.atmel.com/products/at91

4.7.2 Licensing

Open Source Apache 2.0

4.7.3 Limitations

This model is sufficient to boot Linux

4.7.4 Reference

Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

4.8 Peripheral [atmel.ovpworld.org/peripheral/SpecialFunction/1.0] instance: SF

4.8.1 Description

This model contains an accurate Register set interface. The functionality has only been implemented to sufficiently boot uClinux The AT91FR40162SB provides registers that implement the following special functions. Chip Identification RESET Status Protect Mode for more information visit http://www.atmel.com/products/at91

4.8.2 Licensing

Open Source Apache 2.0

4.8.3 Limitations

This model is sufficient to boot Linux

4.8.4 Reference

Rev. 1354D ARM08/02

There are no configuration options set for this peripheral instance.

5.0 Overview of Imperas OVP Virtual Platforms

This document provides the details of the usage of an Imperas OVP Virtual Platform / Module. The first half of the document covers specifics of this particular virtual platform / module.

This second part of the document, includes information about Imperas OVP virtual platforms and modules, how they are built and used.

The Imperas virtual platforms are designed to provide a base for you to run high-speed software simulations of CPU-based SoCs and platforms on any suitable PC. They are typically based on the functionality of vendors fixed or evaluation platforms, enabling you to simulate software on these reference platforms. Typically virtual platforms are fixed and require the vendor to modify or extend them. Imperas virtual platforms are different in that they enable you to extend the functionality of the virtual platform, to closer reflect your own platform, by adding more component models, running different operating systems or adding additional applications.

Imperas virtual platforms are created using the Imperas iGen technology, allowing them to be used with Imperas OVP based simulators and also with Accellera/OSCI compliant SystemC simulators and commercial EDA System Design environments that use SystemC.

Virtual platforms include simulation models of the target devices, including the processor model(s) for the target device plus enough peripheral models to boot an operating system or run bare metal applications. The platform and the peripheral models used in most of the virtual platforms are open source, so that you can easily add new models to the platform as well as modify the existing models. Some models are only provided as binary, normally because the IP owner has restricted the release of the model source. In this case, please contact Imperas for more information.

There are typically several generic flavors of the virtual platforms for specific processor families, some targeting full operating systems, such as Linux, and some which focus on Real Time Operating Systems (RTOS) such as Mentor Nucleus or freeRTOS. OVP models of the processor cores are included in the virtual platforms, and for those processors which support mulitple cores SMP Linux is often supported for that virtual platform. For all of these virtual platforms, many of the peripheral components of the platform are modeled, often including the Ethernet and USB components. The semi-hosting capability of the Imperas virtual platform simulator products enables connection via the Ethernet and USB components from the virtual platform to the real world via the x86 host machine.

The Imperas OVP CPU models are written using the OVP Virtual Machine Interface (VMI) API that defines the behavior of the processor. The VMI API makes a clear line between model and simulator allowing very good optimization and world class high speed performance. The processor models are Instruction Accurate and do not model the detailed cycle timing of the processor and they implement functionality at the level of a Programmers View of the processor and peripherals and the software running on them does not know it is not running on hardware. Many models are provided as a binary shared object

Copyright (c) 2021 Imperas Software Limited

and also as source. This allows the download and use of the model binary or the use of the source to explore and modify the model. The models are run through an extensive QA and regression testing process and most processor model families are validated using technology provided by the processor IP owners. All the models in this platform are developed with the Open Virtual Platforms APIs and are implemented in C. A platform can be modeled as different levels of hierarchy using separately describable and compilable modules.

More information on modeling and APIs can be found on the www.OVPworld.org site.

6.0 Getting Started with Imperas OVP Virtual Platforms

Virtual platforms are downloadable from the OVPworld website OVPworld.org/downloads. You need to browse and look for '<platform processor name> Examples'. You do need to be registered and logged in on the OVP site to download. OVPworld currently provides 32 bit host versions of packages containing virtual platforms.

When downloading, choose, Linux or Windows host. 32 bit packages can be installed and executed on 32 bit or 64 bit hosts. If you require a 64 bit host version please contact Imperas.

For example, for the ARM Versatile Express platform booting Linux on Cortex-A15MP Single, Dual, and Quad core procesors, you would want the download package: 'OVPsim_demo_Linux_ArmVersatileExpress_arm_Cortex-A15MP'.

Most virtual platform packages contain the platform and all the processor and peripheral models needed. You will need to download a simulator to run the platform. You can use OVPsim, downloadable from OVPworld.org/downloads, or you can use one of the Imperas simulators (imperas.com/products) available commercially from Imperas.

7.0 Simulating Software

7.1 Getting a license key to run

After you have downloaded you will need a runtime license key before the simulators will run. For OVPsim please visit OVPworld.org/likey and provide the required information and an evaluation/demo license key will be automatically sent to you. If you are using Imperas, then please contact Imperas for a license key.

7.2 Normal runs

To run a platform, read the section below on command line control of the platform and the section on setting command line arguments.

7.3 Loading Software

For most virtual platforms the platform is already configured to run the default software application/program and there is normally a script to run that sets some arguments. You can then copy/edit this script to select your own applications etc.

The example application programs are typically .elf format files and are provided pre-compiled. There are normally makefiles and associated scripts to recompile the example applications.

To find more information about compiling and loading software, the following document should be looked at: Imperas_Installation_and_Getting_Started.pdf.

7.4 Semihosting

In a virtual platform, semihosting is not normally used as there is normally hardware that implements the appropriate functionality - for example I/O will be handled by UARTs etc.

7.5 Using a terminal (UART)

If the platform includes one or more UARTs you will need to connect a terminal program to it so that you can see output and type into the simulated program. Review the list of peripherals below and see what configuration options it has been set with. In most cases there is an option to set to instruct the simulator to 'pop up' a terminal window connected to the simulated UART.

7.6 Interacting with the simulation (keyboard and mouse)

If the platform has a simulated UART you can normally set a command to get the simulator to pop up a terminal window allowing you to see output from the simulated UART and also allowing you to type characters into the UART that can be processed by the simulated software.

If your simulated platform has an LCD device then you can often configure it to recognize mouse movements and mouse clicks - allowing full interaction.

To see these interactions in action, have a look at some of the available videos available at OVPworld.org/demosandvideos.

7.7 More Information (Documentation) on Simulation

To find more information about running simulations and more of the options the simulators provide, the following documents should be looked at:
Imperas_Installation_and_Getting_Started.pdf
Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
Advanced_Simulation_Control_of_Platforms_and_Modules_User_Guide.pdf
OVP_Control_File_User_Guide.pdf

A full list of the currently available OVP documentation is available: OVPworld.org/documentation.

8.0 Debugging Software running on an Imperas OVP Virtual Platform

The Imperas and OVP simulators have several different interfaces to debuggers. These include several proprietary formats and also the standard GNU RSP format is supported allowing many compatible debuggers to be used. Below are some examples that Imperas directly support.

Copyright (c) 2021 Imperas Software Limited

8.1 Debugging with GDB

A GNU debugger (GDB) can be connected to a processor in a platform using the RSP protocol. This allows the application program running on a processor to be debugged using a specific GDB for the processor selected. When using the Imperas Professional products many connections can be made allowing a GDB to be connected to all the processors in the platform.

The use of GDB is documented: OVPsim_Debugging_Applications_with_GDB_User_Guide.pdf.

*8.2 Debugging with Imperas M*DBG*

The Imperas multi-processor debugger can be connected to a platform and through this connection you can debug application programs running on all of the processors instanced within the platform. It is also capable, within this single unified environment, to debug peripheral model behavioral code in conjunction with the processor application programs.

For more information please see the Imperas M*DBG user guide.

The Imperas multi-processor debugger is also capable of controlling the Imperas Verification Analysis aand Profiling (VAP) tools in real time, making them invaluable to application program development, debugging and analysis.

For more information please see the Imperas VAP tools user guide.

8.3 Debugging with the Imperas eGui and GDB

Imperas eGui gives a GUI front end to the use of the GDB debugger. It allows use of all the features of GDB including source level application program debugging on processors.

*8.4 Debugging with the Imperas eGui and M*DBG*

Imperas eGui gives a GUI front end to the Imperas multi-processor debugger. It provides all the features of this debugger but does so with source level application program debugging on processors and source level debugging of the behavioral code on peripheral components in the platform. A context view shows all the processor and peripheral components within the platform and allows switching between them to examine the state of each at the event at which the simulation was stopped

Imperas eGui provides a menu from which the Imperas VAP tools can be controlled.

8.5 Debugging with Imperas eGui and Eclipse

Imperas provide a GUI based on Eclipse called eGui. This provides a GUI front end to use with a standard GDB or the Imperas MPD (Multi-Processor Debugger).

The use of eGui is documented: eGui_Eclipse_User_Guide.pdf.

A standard Eclipse CDT development environment can be connected to one or more processors in a

Copyright (c) 2021 Imperas Software Limited

platform (multiple processors require an Imperas professional product). The simulation platform is started remotely or using the external tool feature in Eclipse, opens a debug port and awaits the connection with Eclipse. All features provided by the Eclipse CDT development environment are available to be used to debug software applications executing on the processors in the platform.

The use of Eclipse is documented: OVPsim_Debugging_Applications_with_Eclipse_User_Guide.pdf.

8.6 Debugging applications running under a simulated operating system

If the simulated platform is running an Operating System and the platform has a UART or Ethernet etc connection then it is often possible to connect an external debugger and debug the applications running under the simulated operating system.

An example would be a simulated platform running the Linux operating system, such as the MIPS Malta, or ARM Versatile Express. Within the simulated Linux you can start a gdbserver that connects from within the simulation through a UART out to the host PC via a port. Within the host PC you start a terminal program and connect to the port with a debugger such as GDB and can then debug the simulated user application.

9.0 Modifying the Platform / Module

9.1 Platforms / Modules use C/C++ and OVP APIs

The Imperas and OVP simulators execute a platform / module that is written in C/C++ and that makes function calls into the simulator's APIs. Thus the virtual platform / module is compiled from C/C++ into a binary shared object that the simulator loads and runs. OVP provides the definition and documentation that defines the C APIs for modeling the platforms, modules, the peripherals, and the processors. You can find more information about these APIs on the OVP website and in the OVP API documentation.

9.2 Platforms/Modules/Peripherals can be easily built with iGen from Imperas

Imperas provides a product 'iGen' that takes an input script file and creates the C/C++ files needed for platforms, modules, and peripherals - it creates the C/C++ file that is compiled into the platform, module or peripheral that is needed as an object file by the simulator. iGen creates the C/C++ files, you then need to add any necessary behaviors or further details etc. For platforms iGen creates either a C platform or a C++ SystemC TLM2 platform. For peripherals or modules iGen creates the C files and also provides a native C++ SystemC TLM2 interface to allow the peripheral/module to be instantiated in SystemC TLM2 platforms.

Information on iGen is available from: imperas.com/products.

9.3 Re-configuring the platform

There will nornmally be several configuration options that you can set when running the platform without the neeed to change any source. Refer to the section above on command line arguments. If these do not allow you to make the changes you need, then you may need to edit and recompile the source of the platform.

Copyright (c) 2021 Imperas Software Limited

The source of the platform, modules, and the source of the peripherals will be installed as part of the packages you are using. The sources are located in the Imperas/OVP installation VLNV source tree. The VLNV term refers to: Vendor (eg arm.ovpworld.org), Library (eg platform), Name, (eg ArmVersatileExpress-CA15), and Version (eg 1.0). To modify the platform, locate the platform source files.

If you are an Imperas user and have access to iGen, we recommend you modify the source script files and regenerate and recompile the C that makes up the platform. Refer to the Imperas iGen model generator guide and the Imperas platform generator guide.

If you are using the C or SystemC TLM2 platforms with OVPsim, then you can edit the C/C++ files, recompile the source directly using the supplied makefiles, and the run the simulator directly with the resultant shared object.

9.4 Replacing peripherals components

If you need to replace peripherals, find the appropriate place in the source of the platform, make the change you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

9.5 Adding new peripherals components

If you need to add peripherals, find the appropriate place in the source, make the additions you need, and recompile etc. Look in the library for documentation on available peripherals and their configuration options.

If you need to create new peripheral components then use iGen to very quickly create the necessary C/C++ files that get you started. With iGen you can create peripherals with register/memory state in a few lines of iGen source. When adding behavior to the peripherals refer to the OVP API documentation.

Copyright (c) 2021 Imperas Software Limited

10.0 Available Virtual Platforms

Table 8. Imperas / OVP Extendable Platform Kits (13 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
FreescaleKinetis60	freescale.ovpworld.org
FreescaleKinetis64	freescale.ovpworld.org
FreescaleVybridVFxx	freescale.ovpworld.org
MipsMalta	mips.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
XilinxML505	xilinx.ovpworld.org

Table 9. Imperas General Virtual Platforms (6 available)

Name	Vendor
arm-ti-eabi	arm.imperas.com
armm-ti-coff	arm.imperas.com
armm-ti-eabi	arm.imperas.com
HeteroAlteraCycloneV_HPS_CycloneIII_3c120	imperas.ovpworld.org
HeteroArmNucleusMIPSLinux	imperas.ovpworld.org
SiFiveFU540	imperas.ovpworld.org

Table 10. Imperas Modules (component of other platforms) (55 available)

Name	Vendor
AlteraCycloneIII_3c120	altera.ovpworld.org
AlteraCycloneV_HPS	altera.ovpworld.org
AE350	andes.ovpworld.org
ARMv8-A-FMv1	arm.ovpworld.org
ArmIntegratorCP	arm.ovpworld.org
ArmVersatileExpress	arm.ovpworld.org
ArmVersatileExpress-CA15	arm.ovpworld.org
ArmVersatileExpress-CA9	arm.ovpworld.org
AtmelAT91SAM7	atmel.ovpworld.org
ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
ArmuKernel	imperas.ovpworld.org
ArmuKernelDual	imperas.ovpworld.org
BareMetalMIPS	imperas.ovpworld.org
Dual_ARMv8-A-FMv1_VLAN	imperas.ovpworld.org
Hetero_1xArm_3xMips32	imperas.ovpworld.org
Hetero_ARM_RISCV_NeuralNetwork	imperas.ovpworld.org

Copyright (c) 2021 Imperas Software Limited

Hetero_ARMv8-A-FMv1_Cortex-M3	imperas.ovpworld.org
Hetero_ARMv8-A-FMv1_MIPS_microAptiv	imperas.ovpworld.org
Hetero_AlteraCycloneV_HPS_AlteraCycloneIII_3c120	imperas.ovpworld.org
Hetero_ArmIntegratorCP_XilinxMicroBlaze	imperas.ovpworld.org
Hetero_ArmVersatileExpress_MipsMalta	imperas.ovpworld.org
Hetero_ArmVersatileExpress_XilinxMicroBlaze	imperas.ovpworld.org
Quad_ArmVersatileExpress-CA15	imperas.ovpworld.org
RiscvRV32FreeRTOS	imperas.ovpworld.org
MipsMalta	mips.ovpworld.org
iMX6S	nxp.ovpworld.org
RenesasUPD70F3441	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org
virtio	riscv.ovpworld.org
FaultInjection	safepower.ovpworld.org
PublicDemonstrator	safepower.ovpworld.org
Zynq_PL_DualMicroblaze	safepower.ovpworld.org
Zynq_PL_NoC	safepower.ovpworld.org
Zynq_PL_NoC_node	safepower.ovpworld.org
Zynq_PL_NostrumNoC	safepower.ovpworld.org
Zynq_PL_NostrumNoC_node	safepower.ovpworld.org
Zynq_PL_RO	safepower.ovpworld.org
Zynq_PL_SingleMicroblaze	safepower.ovpworld.org
Zynq_PL_TTELNoC	safepower.ovpworld.org
Zynq_PL_TTELNoC_node	safepower.ovpworld.org
Zynq_PL_TTELNoC_processing_node_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_public_demonstrator	safepower.ovpworld.org
Zynq_PL_TTELNoC_sensor_actor_node_public_demonstrator	safepower.ovpworld.org
FU540	sifive.ovpworld.org
S51CC	sifive.ovpworld.org
coreip-s51-arty	sifive.ovpworld.org
coreip-s51-rtl	sifive.ovpworld.org
dualFifo	vendor.com
XilinxML505	xilinx.ovpworld.org
Zynq	xilinx.ovpworld.org
Zynq_PL_Default	xilinx.ovpworld.org
Zynq_PS	xilinx.ovpworld.org
zc702	xilinx.ovpworld.org
zc706	xilinx.ovpworld.org

Table 11. Imperas / OVP Bare Metal Virtual Platforms (22 available)

Name	Vendor
BareMetalNios_IISingle	altera.ovpworld.org
BareMetalArcSingle	arc.ovpworld.org
BareMetalArm7Single	arm.ovpworld.org
BareMetalArmCortexADual	arm.ovpworld.org
BareMetalArmCortexASingle	arm.ovpworld.org
BareMetalArmCortexASingleAngelTrap	arm.ovpworld.org
BareMetalArmCortexMSingle	arm.ovpworld.org

Copyright (c) 2021 Imperas Software Limited

ArmCortexMFreeRTOS	imperas.ovpworld.org
ArmCortexMuCOS-II	imperas.ovpworld.org
BareMetalArmx1Mips32x3	imperas.ovpworld.org
Or1kUclinux	imperas.ovpworld.org
BareMetalM14KSingle	mips.ovpworld.org
BareMetalMips32Dual	mips.ovpworld.org
BareMetalMips32Single	mips.ovpworld.org
BareMetalMips64Single	mips.ovpworld.org
BareMetalMipsDual	mips.ovpworld.org
BareMetalMipsSingle	mips.ovpworld.org
BareMetalOr1kSingle	ovpworld.org
BareMetalM16cSingle	posedgesoft.ovpworld.org
BareMetalPowerPc32Single	power.ovpworld.org
BareMetalV850Single	renesas.ovpworld.org
ghs-multi	renesas.ovpworld.org

#

Copyright (c) 2021 Imperas Software Limited