OVP Guide to Using Processor Models

Model Specific Information for variant ARM_Cortex-M0plus

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

Author	Imperas Software Limited
Version	0.5
Filename	OVP_Model_Specific_Information_armm_Cortex-M0plus.pdf
Created	27 February 2018
Status	OVP Standard Release

Copyright Notice

Right to Copy Documentation

Destination Control Statement

Disclaimer

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

1 Overview

This document provides the details of an OVP Fast Processor Model variant.

OVP Fast Processor Models are written in C and provide a C API for use in C based platforms. The models also provide a native interface for use in SystemC TLM2 platforms.

The models are written using the OVP VMI API that provides a Virtual Machine Interface 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.

Most models are provided as a binary shared object 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 model families are validated using technology provided by the processor IP owners.

There is a companion document (OVP Guide to Using Processor Models) which explains the general concepts of OVP Fast Processor Models and their use. It is downloadable from the OVPworld website documentation pages.

1.1 Description

ARMM Processor Model

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.

The License agreement does not entitle Licensee to use the model to emulate an ARM based system to run application software in a production or live environment.

Source of model available under separate Imperas Software License Agreement.

1.3 Limitations

Performance Monitors are not implemented.
Debug Extension and related blocks are not implemented.

1.4 Verification

Models have been extensively tested by Imperas. ARM Cortex-M models have been successfully used by customers to simulate the Micrium uC/OS-II kernel and FreeRTOS.

1.5 Features

The model is configured with 16 interrupts and 2 priority bits (use override_numInterrupts parameter to change the number of interrupts; the number of priority bits is fixed in this profile).
Thumb instructions are supported.
MPU is present. Use parameter override_MPU_TYPE to disable it or change the number of MPU regions if required.
SysTick timer is present. Use parameter SysTickPresent to disable it if required.
Unprivileged/Privileged Extension is present. Use parameter unprivilegedExtension to disable it if required.
VTOR register is present. Use parameter VTORPresent to disable it if required.

2 Configuration

2.1 Location

The model source and object file is found in the VLNV tree at:
arm.ovpworld.org/processor/armm/1.0

2.2 GDB Path

The default GDB for this model is found at:
$IMPERAS_HOME/lib/$IMPERAS_ARCH/gdb/arm-none-eabi-gdb

2.3 Semi-Host Library

The default semi-host library file is found in the VLNV tree at :
arm.ovpworld.org/semihosting/armNewlib/1.0

2.4 Processor Endian-ness

This model can be set to either endian-ness (normally by a pin, or the ELF code).

2.5 QuantumLeap Support

This processor is qualified to run in a QuantumLeap enabled simulator.

2.6 Processor ELF Code

The ELF code supported by this model is: 0x28

3 Other Variants in this Model

Table 1. All variants in this model

Variant
ARMv6-M
ARMv7-M
Cortex-M0
Cortex-M0plus
Cortex-M1
Cortex-M3
Cortex-M4
Cortex-M4F

4 Bus Ports

Table 2. Bus Ports

Type	Name	min	max
master (initiator)	INSTRUCTION	32	32
master (initiator)	DATA	32	32

5 Net Ports

Table 3. Net Ports

Name	Type
sysResetReq	output
intISS	output
eventOut	output
lockup	output
int	input
reset	input
nmi	input
eventIn	input
int0	input
int1	input
int2	input
int3	input
int4	input
int5	input
int6	input
int7	input

int8	input
int9	input
int10	input
int11	input
int12	input
int13	input
int14	input
int15	input

6 FIFO Ports

No FIFO Ports in this model.

7 Parameters

Table 4. Parameters that can be set in the model, type:

Name	Type	Description
verbose	Boolean	Specify verbosity of output
showHiddenRegs	Boolean	Show hidden registers during register tracing
UAL	Boolean	Disassemble using UAL syntax
compatibility	Enumeration	Specify compatibility mode ISA=0 gdb=1 nopBKPT=2
override_debugMask	Uns32	Specifies debug mask, enabling debug output for model components
instructionEndian	Endian	The architecture specifies that instruction fetch is always little endian; this attribute allows the defined instruction endianness to be overridden if required
resetAtTime0	Boolean	Reset the model at time=0 (default=1)
unprivilegedExtension	Boolean	Specify presence of Unprivileged/Privileged Extension
VTORPresent	Boolean	Specify presence of VTOR register
SysTickPresent	Boolean	Specify presence of SysTick timer
override_CPUID	Uns32	Override system CPUID register
override_MPU_TYPE	Uns32	Override system MPU_TYPE register
override_VTOR	Uns32	Override VTOR register reset value
override_STRoffsetPC12	Uns32	Specifies that STR/STR of PC should do so with 12:byte offset from the current instruction (if 1), otherwise an 8:byte offset is used
override_ERG	Uns32	Specifies exclusive reservation granule
override_numInterrupts	Uns32	Specifies number of external interrupt lines

8 Execution Modes

Table 5. CPU modes implemented in the model, type:

Name	Code

Thread	0
Handler	1

9 Exceptions

Table 6. Exceptions handled by the model, type:

Name	**Code**
None	0
Reset	1
NMI	2
HardFault	3
SVCall	11
PendSV	14
SysTick	15
ExternalInt000	16
ExternalInt001	17
ExternalInt002	18
ExternalInt003	19
ExternalInt004	20
ExternalInt005	21
ExternalInt006	22
ExternalInt007	23
ExternalInt008	24
ExternalInt009	25
ExternalInt00a	26
ExternalInt00b	27
ExternalInt00c	28
ExternalInt00d	29
ExternalInt00e	30
ExternalInt00f	31

10 Hierarchy of the model

A CPU core may allow the user to configure it to instance many processors of a Symmetrical Multi Processor (SMP). A CPU core may also have sub elements within a processor, for example hardware threading blocks.
OVP processor models can be written to include SMP blocks and to have many levels of hierarchy.

Some OVP CPU models may have a fixed hierarchy, and some may be configured by settings in a configuration register. Please see the register definitions of this model.

This model documentation shows the settings and hierarchy of the default settings for this model variant.

10.1 Level 1:

This level in the model hierarchy has 2 commands.

This level in the model hierarchy has 4 register groups:

Table 7. Register groups

Group name	Registers
Core	16
Control	5
System	31
Integration_support	2

This level in the model hierarchy has no children.

11 Model Commands

11.1 Level 1:

11.1.1 isync

specify instruction address range for synchronous execution

Table 8. isync command arguments

Argument	Type	Description
-addresshi	Uns64	end address of synchronous execution range
-addresslo	Uns64	start address of synchronous execution range

11.1.2 itrace

enable or disable instruction tracing

Table 9. itrace command arguments

Argument	Type	Description
-after	Uns64	apply after this many instructions
-enable	Boolean	enable instruction tracing
-instructioncount	Boolean	include the instruction number in each trace
-off	Boolean	disable instruction tracing
-on	Boolean	enable instruction tracing
-registerchange	Boolean	show registers changed by this instruction
-registers	Boolean	show registers after each trace

12 Registers

12.1 Level 1:

12.1.1 Core

Table 10. Registers at level 1, type: , register group: 'Core'

Name	Bits	Initial value (Hex)		Description
r0	32	0	rw	
r1	32	0	rw	
r2	32	0	rw	
r3	32	0	rw	
r4	32	0	rw	
r5	32	0	rw	
r6	32	0	rw	
r7	32	0	rw	

r8	32	0	rw	
r9	32	0	rw	
r10	32	0	rw	
r11	32	0	rw	frame pointer
r12	32	0	rw	
sp	32	0	rw	stack pointer
lr	32	0	rw	
pc	32	0	rw	program counter

12.1.2 Control

Table 11. Registers at level 1, type: , register group: 'Control'

Name	Bits	Initial value (Hex)		Description
cpsr	32	0	rw	xPSR register. Includes APSR, IPSR and EPSR
control	32	0	rw	
primask	32	0	rw	
sp_process	32	0	rw	stack pointer
sp_main	32	0	rw	stack pointer

12.1.3 System

Table 12. Registers at level 1, type: , register group: 'System'

Name	Bits	Initial value (Hex)		Description
ACTLR	32	0	rw	Address 0xe000e008
SYST_CSR	32	4	rw	Address 0xe000e010
SYST_RVR	32	0	rw	Address 0xe000e014
SYST_CVR	32	0	rw	Address 0xe000e018
SYST_CALIB	32	0	rw	Address 0xe000e01c
NVIC_ISER0	32	0	rw	Address 0xe000e100
NVIC_ICER0	32	0	rw	Address 0xe000e180
NVIC_ISPR0	32	0	rw	Address 0xe000e200
NVIC_ICPR0	32	0	rw	Address 0xe000e280
NVIC_IPR0	32	0	rw	Address 0xe000e400
NVIC_IPR1	32	0	rw	Address 0xe000e404
NVIC_IPR2	32	0	rw	Address 0xe000e408
NVIC_IPR3	32	0	rw	Address 0xe000e40c
CPUID	32	410cc601	r-	Address 0xe000ed00
ICSR	32	1000	rw	Address 0xe000ed04
VTOR	32	0	rw	Address 0xe000ed08
AIRCR	32	fa050000	rw	Address 0xe000ed0c

SCR	32	0	rw	Address 0xe000ed10
CCR	32	200	rw	Address 0xe000ed14
SHPR2	32	0	rw	Address 0xe000ed1c
SHPR3	32	0	rw	Address 0xe000ed20
SHCSR	32	0	rw	Address 0xe000ed24
MPU_RNR	32	0	rw	Address 0xe000ed98
MPU_RBAR	32	0	rw	Address 0xe000ed9c
MPU_RASR	32	0	rw	Address 0xe000eda0
MPU_RBAR_A1	32	0	rw	Address 0xe000eda4
MPU_RASR_A1	32	0	rw	Address 0xe000eda8
MPU_RBAR_A2	32	0	rw	Address 0xe000edac
MPU_RASR_A2	32	0	rw	Address 0xe000edb0
MPU_RBAR_A3	32	0	rw	Address 0xe000edb4
MPU_RASR_A3	32	0	rw	Address 0xe000edb8

12.1.4 Integration_support

Table 13. Registers at level 1, type: , register group: 'Integration_support'

Name	**Bits**	**Initial value (Hex)**		**Description**
executionPri	32	7fffffff	r-	current execution priority level
stackDomain	32	98f6a08	r-	stack domain for current execution level

#

