

Cortex[®]-M7 MPCore Processor Cycle Model

Version 9.6.0

User Guide

Non-Confidential



Cortex-M7 MPCore Processor Cycle Model

User Guide

Copyright © 2017 Arm Limited (or its affiliates). All rights reserved.

Release Information

The following changes have been made to this document.

Change History			
Issue	Date	Confidentiality	Change
0906-00-00	November 2017	Non-Confidential	Release with 9.6
0903-00-01	June 2017	Non-Confidential	Release with 9.3.

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017 Arm Limited. All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Contents

Preface

About This Guide	7
Audience	7
Conventions	8
Further reading	9
Glossary	10

Chapter 1. Using the Cycle Model in SoC Designer

Cortex-M7 Functionality	12
Supported Hardware Features	12
Unsupported Hardware Features	12
Features Additional to the Hardware	13
Adding and Configuring the SoC Designer Component	14
SoC Designer Component Files	14
Adding the Cycle Model to the Component Library	15
Adding the Component to the SoC Designer Canvas	15
ESL Ports	16
Available Component ESL Ports	16
Reset Behavior and Ports	17
Setting Component Parameters	18
Debug Features	23
Register Information	23
Core Registers	24
Debug Registers	25

System Control Registers	25
NVIC Registers	26
MPU Registers	26
SysTick Registers	26
FPB Registers	27
DWT Registers	27
TCM Control Registers	28
VFP Registers	28
Cache Control Registers	29
Run To Debug Point Feature	29
Memory Information	29
Disassembly View	30
Available Profiling Data	30
Hardware Profiling	30
Software Profiling	30

Preface

A Cycle Model component is a library developed from Arm intellectual property (IP) that is generated through Cycle Model Studio™. The Cycle Model then can be used within a virtual platform tool, for example, SoC Designer.

About This Guide

This guide provides all the information needed to configure and use the Cortex™-M7 Cycle Model in SoC Designer.

Audience

This guide is intended for experienced hardware and software developers who create components for use with SoC Designer. You should be familiar with the following products and technology:

- SoC Designer
- Hardware design verification
- Verilog or SystemVerilog programming language

Conventions

This guide uses the following conventions:

Convention	Description	Example
<code>courier</code>	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	<code>sparseMem_t SparseMemCreateNew();</code>
<i>italic</i>	New or unusual words or phrases appearing for the first time.	<i>Transactors</i> provide the entry and exit points for data ...
bold	Action that the user performs.	Click Close to close the dialog.
<text>	Values that you fill in, or that the system automatically supplies.	<platform>/ represents the name of various platforms.
[text]	Square brackets [] indicate optional text.	\$CARBON_HOME/bin/modelstudio [<filename>]
[text1 text2]	The vertical bar indicates “OR,” meaning that you can supply text1 or text 2.	\$CARBON_HOME/bin/modelstudio [<name>.symtab.db <name>.ccfg]

Also note the following references:

- References to C code implicitly apply to C++ as well.
- File names ending in .cc, .cpp, or .cxx indicate a C++ source file.

Further reading

This section lists related publications. The following publications provide information that relate directly to SoC Designer:

- *SoC Designer User Guide* (100996)

The following publications provide reference information about Arm products:

- *Cortex-M7 Technical Reference Manual* (100340)
- *AMBA Specification (Rev 2.0)* (IHI0011)

See <http://infocenter.arm.com/help/index.jsp> for access to Arm documentation.

The following publications provide additional information on simulation:

- IEEE 1666™ SystemC Language Reference Manual, (IEEE Standards Association)
- SPIRIT User Guide, Revision 1.2, SPIRIT Consortium.

Glossary

AMBA	<i>Advanced Microcontroller Bus Architecture.</i> The Arm open standard on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a System-on-Chip (SoC).
AHB	<i>Advanced High-performance Bus.</i> A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA AXI protocol.
APB	<i>Advanced Peripheral Bus.</i> A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports.
AXI	<i>Advanced eXtensible Interface.</i> A bus protocol that is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.
Cycle Model	A software object created by the Cycle Model Studio (or <i>Cycle Model compiler</i>) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Cycle Model Studio	Graphical tool for generating, validating, and executing hardware-accurate software models. It creates a Cycle Model, and it also takes a Cycle Model as input and generates a component that can be used in SoC Designer, Platform Architect, or Accellera SystemC for simulation.
CASI	<i>ESL API Simulation Interface</i> , is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	<i>ESL API Debug Interface</i> , enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	<i>ESL API Profiling Interface</i> , enables collecting historical data from a component and displaying the results in various formats.
Component	Building blocks used to create simulated systems. Components are connected together with unidirectional transaction-level or signal-level connections.
ESL	<i>Electronic System Level.</i> A type of design and verification methodology that models the behavior of an entire system using a high-level language such as C or C++.
HDL	<i>Hardware Description Language.</i> A language for formal description of electronic circuits, for example, Verilog.
RTL	<i>Register Transfer Level.</i> A high-level hardware description language (HDL) for defining digital circuits.
SoC Designer	High-performance, cycle accurate simulation framework which is targeted at System-on-a-Chip hardware and software debug as well as architectural exploration.
SystemC	SystemC is a single, unified design and verification language that enables verification at the system level, independent of any detailed hardware and software implementation, as well as enabling co-verification with RTL design.
Transactor	<i>Transaction adaptors.</i> You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.

Chapter 1

Using the Cycle Model in SoC Designer

This chapter describes the functionality of the Cycle Model component, and how to use it in SoC Designer. It contains the following sections:

- [Cortex-M7 Functionality](#)
- [Adding and Configuring the SoC Designer Component](#)
- [ESL Ports](#)
- [Setting Component Parameters](#)
- [Debug Features](#)
- [Available Profiling Data](#)

1.1 Cortex-M7 Functionality

This section provides a summary of the functionality of the Cycle Model compared to that of the hardware, and the performance and accuracy of the Cycle Model. For details, see the *Cortex-M7 Technical Reference Manual* (100340).

- [Supported Hardware Features](#)
- [Unsupported Hardware Features](#)
- [Features Additional to the Hardware](#)

1.1.1 Supported Hardware Features

The following features of the Cortex-M7 hardware are fully implemented in the Cortex-M7 Cycle Model:

- Cortex-M7 Integer Core
- NVIC – Nested Vectored Interrupt Controller
- WIC – Wakeup Interrupt Controller Interface Support (support is for the interface only).
- AXI4 Master Memory Interface
- APB v3.0 interface for accessing the external Private Peripheral Bus
- FPB – Flash Patch and Debug
- DWT – Debug Watchpoint and Trace
- MPU – Memory Protection Unit (optional)
- Floating Point Unit (optional) — Single and/or double precision
- Level 1 memory system includes a three stage prefetch unit (PU) including a static branch predictor and call-return stack. The LSU is also three-stage and includes a store queue for efficient access to the Tightly Coupled Memory (TCM) ports.
- ETM supported to enable software profiling.
- Configurable DCache and ICache sizes.

1.1.2 Unsupported Hardware Features

The following features of the Cortex-M7 hardware are not implemented in the Cortex-M7 Cycle Model:

- SW/JTAG-DP
- ITM
- Semihosting

1.1.3 Features Additional to the Hardware

The following features that are implemented in the Cortex-M7 Cycle Model to enhance usability do not exist in the Cortex-M7 hardware:

- Debug and Profiling. For more information about debug and profiling features, refer to the sections [Debug Features](#) and [Available Profiling Data](#), respectively.
- The “run to debug point” feature has been added. This feature forces the debugger to advance the processor to the debug state instead of having the Cycle Model get into a non-debuggable state. See [Run To Debug Point Feature](#) for more information.

1.2 Adding and Configuring the SoC Designer Component

The following topics briefly describe how to use the component. See the *SoC Designer User Guide* for more information.

- [SoC Designer Component Files](#)
- [Adding the Cycle Model to the Component Library](#)
- [Adding the Component to the SoC Designer Canvas](#)

1.2.1 SoC Designer Component Files

The component files are the final output from the Cycle Model Studio compile and are the input to SoC Designer. There are two versions of the component; an optimized *release* version for normal operation, and a *debug* version.

On Linux, the *debug* version of the component is compiled without optimizations and includes debug symbols for use with gdb. The *release* version is compiled without debug information and is optimized for performance.

On Windows, the *debug* version of the component is compiled referencing the debug runtime libraries so it can be linked with the debug version of SoC Designer. The *release* version is compiled referencing the release runtime library. Both release and debug versions generate debug symbols for use with the Visual C++ debugger on Windows.

The provided component files are listed below:

Table 1-1 SoC Designer Component Files

Platform	File	Description
Linux	maxlib.lib<model_name>.conf	SoC Designer configuration file
	lib<component_name>.mx.so	SoC Designer component runtime file
	lib<component_name>.mx_DBG.so	SoC Designer component debug file
Windows	maxlib.lib<model_name>.windows.conf	SoC Designer configuration file
	lib<component_name>.mx.dll	SoC Designer component runtime file
	lib<component_name>.mx_DBG.dll	SoC Designer component debug file

Additionally, this User Guide PDF file is provided with the component.

1.2.2 Adding the Cycle Model to the Component Library

The compiled Cycle Model component is provided as a configuration file (*.conf*). To make the component available in the Component Window in SoC Designer Canvas, perform the following steps:

1. Launch SoC Designer Canvas.
2. From the *File* menu, select **Preferences**.
3. Click on **Component Library** in the list on the left.
4. Under the *Additional Component Configuration Files* window, click **Add**.
5. Browse to the location where the Cycle Model is located and select the component configuration file:
 - `maxlib.lib<model_name>.conf` (for Linux)
 - `maxlib.lib<model_name>.windows.conf` (for Windows)
6. Click **OK**.
7. To save the preferences permanently, click the **OK & Save** button.

The component is now available from the SoC Designer *Component Window*.

1.2.3 Adding the Component to the SoC Designer Canvas

Locate the component in the *Component Window* and drag it out to the Canvas. The component's appearance may vary depending on your specific device configuration.

Additional ports are provided depending on the model RTL configuration file, *default.conf*, used to create the Cycle Model.

1.3 ESL Ports

This section describes the differences between the pins listed in the *Arm Cortex-M7 Technical Reference Manual* (100340) and those on the Cortex-M7 Cycle Model.

- [Available Component ESL Ports](#) — Describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer, or those created by wrapping multiple hardware pins into transactors.
- [Reset Behavior and Ports](#) — Describes the default reset behavior of the Cycle Model and how to generate a reset sequence during simulation.

1.3.1 Available Component ESL Ports

Table 1-2 describes ports that have been added to the Cycle Model. Additional ports are visible in SoC Designer, which correspond to the hardware pins. See the *Cortex-M7 Technical Reference Manual* (100340) for descriptions of these pins.

Note: Some ESL component port values can be set using a component parameter. In those cases, the parameter value is used whenever the ESL port is not connected. If the port is connected, the connection value takes precedence over the parameter value.

Table 1-2 ESL Component Ports

ESL Port	Description	Type
AHBLiteInitiator_AHBD_slave	AHB Initiator Slave Transactor.	Transaction Slave
AHBLiteInitiator_AHBP_master	AHB Initiator Master Transactor.	Transaction Master
AHBLiteTarget_AHBS_slave	AHB Target Slave Transactor.	Transaction Slave
APB_EPPB_master	APB EPPB Master Transactor.	Transaction Master
AXI4_master	AXI4 Master Transactor.	Transaction Master
CLKIN	Free running clock.	Clock Slave
clk-in	This port is used internally. Leave unconnected.	Clock Slave
DnTCM_Debug	TCM Debug Transactor.	Transaction Master
extSemi_cpu0	Semihosting can be enabled by connecting this port to the SoC Designer semi-host component, contained in the Arm SoC Designer Standard Model Library.	Transaction Master
ITCM_Debug	ITCM Debug Transactor.	Transaction Master

1.3.2 Reset Behavior and Ports

The Cycle Model is reset internally each time SoC Designer Simulator is initialized. This behavior is standard and can not be changed. To view the internal reset sequence, set the *Align Waveforms* parameter to False (see [Setting Component Parameters](#)), and this data appears in the waveform.

At simulation time zero and while simulation is running, you can generate a reset sequence. To do so, drive the reset pins on the component using external signals (for example, using the MxSigDriver component).

For information about reset pin names, bit ordering (for multiple cores), and required reset sequence, refer to the *Technical Reference Manual* for your IP.

1.4 Setting Component Parameters

You can change the settings of all the component parameters in SoC Designer Canvas, and of some of the parameters in SoC Designer Simulator. To modify the component's parameters :

1. In the Canvas, right-click on the component and select **Component Information**. You can also double-click the component. The *Edit Parameters* dialog box appears. The list of available parameters may differ slightly depending on the settings that you enabled in the configuration file (*default.conf*) when creating the component.
2. In the *Parameters* window, double-click the **Value** field of the parameter that you want to modify.
3. If it is a text field, type a new value in the *Value* field. If a menu choice is offered, select the desired option. The parameters are described in Table 1-3.

Table 1-3 Component Parameters

Name	Description	Allowed Values	Default Value	Init/ Runtime
ACLKEN	Clock enable for the AXI master port.	0, 1	1	Runtime
AFVALIDMD	ATB interface FIFO flush request (data trace).	0, 1	0	Runtime
AFVALIDMI	ATB interface FIFO flush request (instruction trace).	0, 1	0	Runtime
AHBLiteInitiator_AHBD_slave Align Data	AHB Lite Initiator Transactor Data Alignment.	true, false	false	Init
AHBLiteInitiator_AHBD_slave Big Endian	Endianness of data in AHBD Transactor.	true, false	false	Init
AHBLiteInitiator_AHBD_slave Enable Debug Messages	Enables/disables AHBLiteInitiator_AHBD_slave port debug.	true, false	false	Runtime
AHBLiteInitiator_AHBD_slave HREADYIN	AHBD Transactor HREADY Signal.	0, 1	0	Init
AHBLiteInitiator_AHBD_slave region size [0-5]	AHBD Transactor region size.	0 - 0x100000000	0	Init
AHBLiteInitiator_AHBD_slave region start [0-5]	AHBD Transactor region start address	0 - 0xFFFFFFFF	0x0	Init
AHBLiteInitiator_AHBP_master Align Data	AHBP Transactor data alignment.	true, false	false	Init
AHBLiteInitiator_AHBP_master Big Endian	Endianness of data in AHBP Transactor.	true, false	false	Init
AHBLiteInitiator_AHBP_master Enable Debug Messages	Enables/disables AHBLiteInitiator_AHBP_master port debug.	true, false	false	Init
AHBLiteTarget_AHBS_slave Align Data	AHB Lite Target Transactor Data Alignment.	true, false	false	Init

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
AHBLiteTarget_AHBS_slave Big Endian	Endianness of data in AHBP Transactor.	true, false	false	Init
AHBLiteTarget_AHBS_slave Enable Debug Messages	Enables/disables AHBLiteTarget_AHBS_slave port debug.	true, false	false	Runtime
AHBLiteTarget_AHBS_slave Filter HREADYIN	AHBD Transactor HREADY Signal.	0, 1	0	Init
AHBLiteTarget_AHBS_slave region size [0-5]	AHBS Transactor region size.	0 - 0x100000000	0 — 0x100000000 1 - 5 — 0x0	Init
AHBLiteTarget_AHBS_slave region start [0-5]	AHBS Transactor region start address	0 - 0x100000000	0x0	Init
Align Waveforms	When set to <i>true</i> , waveforms dumped from the component are aligned with the SoC Designer simulation time. The reset sequence, however, is not included in the dumped data. When set to <i>false</i> , the reset sequence is dumped to the waveform data; however, the component time is not aligned with the SoC Designer time.	true, false not settable in Canvas	true	Init
APB_EPPB_master Base Address	APB EPPB Master base address	0x40000000 - 0x5FFFFFFF	0 — 0x100000000 1 - 5 — 0x0	Init
APB_EPPB_master Enable Debug Messages	Enables/disables APB EPPB Master port debug.	true, false	false	Runtime
APB_EPPB_master PReady Default High	EPPB Master Transactor PReady signal.	true, false	true	Runtime
APB_EPPB_master Protocol Variant	Protocol Variant in use on APB EPPB Master port.	APB3	APB3	Init
APB_EPPB_master Size	EPPB Transactor size region	0 - 0x100000000	0	Init
ATREADYMD	ATDATA can be accepted (data trace).	0, 1	0	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
ATREADYMI	ATDATA can be accepted (instruction trace).	0, 1	0	Runtime
AXI4_master Enable Debug Messages	Enables/disables AXI4_master port debug.	true, false	false	Runtime
AXI4_master Protocol Variant	Protocol Variant in use on AXI4 Master port.	AXI4	AXI4	Init
ARM CycleModels DB Path	Sets the directory path to the database file.	Not Used	empty	Init
CFGAHBPSZ	Indicates the AHB peripheral interface region size.	0 - 7	0	Runtime
CFGBIGEND	Static endianness setting. Refer to the <i>Arm Cortex-M7 Processor Integration and Implementation Manual</i> (100340) for more information.	0, 1	0	Runtime
CFGDTCMSZ	Indicates the DTCM size.	0 - 15	0	Runtime
CFGITCMSZ	Indicates the ITCM size.	0 - 15	0	Runtime
CFGSTCALIB	SysTick calibration.	0, 1	0	Runtime
CLK1EN	Clock enable for dual-redundant core.	0, 1	1	Runtime
CLKEN	Clock enable for clk clock gate.	0, 1	1	Runtime
CTICHIN	CTI Channel In.	0, 1	0	Runtime
DCACHESIZE	Data Cache size in KB	0x0(4KB) 0x1(8KB) 0x3(16KB) 0x7(32KB) 0xf(64KB)	0x3 (16KB)	Init
DnTCM_Debug Enable Debug Messages	Enables/disables DnTCM_Debug port debug.	true, false	false	Runtime
DnTCMERR	Data TCM Ports	N/A	0	Runtime
DnTCMRDATA			0	Runtime
DnTCMRETRY			0	Runtime
DnTCMWAIT			0	Runtime
Dump Waveforms	Determines whether SoC Designer dumps waveforms for this component.	true, false	false	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
Enable Debug Messages	Determines whether debug messages are logged for the component.	true, false	false	Runtime
ETMCLKEN	ETM clock enable.	0, 1	1	Runtime
FCLK1EN	Clock enable for dual-redundant core.	0, 1	1	Runtime
FCLKEN	Clock enable for fclk clock gate.	0, 1	1	Runtime
HCLK1EN	Clock enable for dual-redundant core.	0, 1	1	Runtime
HCLKEN	Clock enable for hclk clock gate.	0, 1	1	Runtime
ICACHESIZE	Instruction Cache size in KB.	0x0(4KB) 0x1(8KB) 0x3(16KB) 0x7(32KB) 0xf(64KB)	0x3 (16KB)	Init
INITAHBPEN	AHBP enable out of reset.	0, 1	0	Runtime
INITRETRYEN	TCM Retry enables out of reset.	0, 1	0	Runtime
INITRMWEN	TCM RMW initialization out of reset. Refer to the <i>Arm Cortex-M7 Processor Integration and Implementation Manual</i> (100340) for more information.	0, 1	0	Runtime
INITTCMEN	TCM enable initialization out of reset. Refer to the <i>Arm Cortex-M7 Processor Integration and Implementation Manual</i> (100340) for more information.	0, 1	0	Runtime
INITVTOR	Vector table offset register out of reset.	0 - 0x1FFFFFFF	0	Runtime
IRQ	External interrupt signals.	Configuration-dependent	0	Runtime
ITCM_Debug Enable Debug Messages	Enables/disables ITCM_Debug port debug.	true, false	false	Runtime

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Init/ Runtime
ITCMERR	ITCM Data Signals	N/A	0	Runtime
ITCMRDATA			0	Runtime
ITCMRETRY			0	Runtime
ITCMWAIT			0	Runtime
NMI	Non Maskable Interrupt.	0, 1	0	Runtime
RXEV	Event in.	0, 1	0	Runtime
STCLKEN	Enable signal for the Sys-Tick logic.	0, 1	1	Runtime
SYNCREQD	Trace synchronization request from data trace sink.	0, 1	0	Runtime
SYNCREQI	Trace synchronization request from instruction trace sink.	0, 1	0	Runtime
Waveform File ¹	Name of the waveform file.	<i>string</i>	arm_cm_CortexM7.vcd	Init
Waveform Format	The format of the waveform dump file.	VCD, FSDB	VCD	Init
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	Init
WICENREQ	Active HIGH request for deep sleep to be WIC-based deep sleep. Driven from the power management unit.	0, 1	0	Runtime

1. When enabled, SoC Designer writes accumulated waveforms to the waveform file in the following situations: when the waveform buffer fills, when validation is paused and when validation finishes, and at the end of each validation run.

1.5 Debug Features

The Cortex-M7 Cycle Model has a debug interface (CADI) that allows the user to view, manipulate, and control the registers and memory, and display disassembly for programs running on the Cycle Model in the SoC Designer Simulator or any debugger that supports CADI. A view can be accessed in SoC Designer Simulator by right clicking on the Cycle Model and choosing the appropriate menu entry.

- [Register Information](#)
- [Run To Debug Point Feature](#)
- [Memory Information](#)
- [Disassembly View](#)

1.5.1 Register Information

The Cortex-M7 Cycle Model has many sets of registers that are accessible via the debug interface. Registers are grouped into sets according to functional area.

- [Core Registers](#)
- [Debug Registers](#)
- [System Control Registers](#)
- [MPU Registers](#)
- [SysTick Registers](#)
- [NVIC Registers](#)
- [DWT Registers](#)
- [TCM Control Registers](#)
- [VFP Registers](#)
- [Cache Control Registers](#)

See the *Cortex-M7 Technical Reference Manual* (100340) for detailed descriptions of these registers. Certain fields in some of the registers are read only.

1.5.1.1 Core Registers

The Core group contains the Arm Architectural registers.

Note that all read-write registers are writeable at debuggable point only. Otherwise, a warning is printed.

Table 1-4 Core Registers

Name	Description	Type
R0	R0 register	read-write
R1	R1 register	read-write
R2	R2 register	read-write
R3	R3 register	read-write
R4	R4 register	read-write
R5	R5 register	read-write
R6	R6 register	read-write
R7	R7 register	read-write
R8	R8 register	read-write
R9	R9 register	read-write
R10	R10 register	read-write
R11	R11 register	read-write
R12	R12 register	read-write
R13	R13/Stack Pointer (SP) register	read-write
R14	R14/Link Register (LR)	read-write
R15	R15/PC (Program Counter) Register	read-write
XPSR	Program Status Register	read-write
PRIMASK	PRIMASK	read-write
BASEPRI	BASEPRI	read-only
FAULTMASK	FAULTMASK	read-write
CONTROL	CONTROL	read-write

1.5.1.2 Debug Registers

The Debug group contains miscellaneous debug-related registers.

Table 1-5 Debug Registers

Name	Description	Type
DFSR	Debug Fault Status Register	read-write
DHCSR	Debug Halting Control and Status Register - 0xE000EDF0	read-write
DEMCR	Debug Exception and Monitor Control Register - 0xE000EDFC	read-write
DCRSR	Debug Core Register Selector Register - 0xE000EDF4	read-write
DCRDR	Debug Core Register Data Register - 0xE000EDF8	read-write

1.5.1.3 System Control Registers

The system control registers provide miscellaneous configuration and status information.

Table 1-6 System Control Registers

Name	Description	Type
ACTLR	Auxiliary Control Register - 0xE000E008	read-write
ICSR	Interrupt Control and State Register - 0xE000ED04	read-only
VTOR	Vector Table Offset register 0xE000ED08	read-write
AIRCR	Application Interrupt Reset Control register 0xE000ED0C	read-write
SCR	System Control register 0xE000ED10	read-write
CCR	Config Control register 0xE000ED14	read-write
SHPR1	System Handlers 4-7Priority Register 1	read-write
SHPR2	System Handlers 8-11Priority Register 2	read-write
SHPR3	System Handlers 12-15 Priority Register 3	read-write
SHCSR	System Handler Control And State register 0xE000ED24	read-write
CFSR	Configurable Fault Status Register 0xE000ED28	read-only
MMFAR	MemManage Fault Address Register	read-write
BFAR	Bus Fault Address register 0xE000ED38	read-only
HFSR	HardFault Status register 0xE000ED2C	read-only
CPACR	Coprocessor Access Control Register - 0xE000ED88	read-write

1.5.1.4 NVIC Registers

The NVIC group contains registers for the Nested Vector Interrupt Controller. The number of registers depends on the number of IRQs specified in your configuration; for every 32-pin increment, the number of registers increments; e.g., for a 32-pin configuration, only NVIC_ISER0 will be present. A 240-pin configuration yields all eight ISER registers.

Table 1-7 NVIC Registers

Name	Description	Type
NVIC_ISER[0 - 7]	Interrupt Set-Enable Registers	read-write
NVIC_ICER[0 - 7]	Interrupt Clear-Enable Registers	read-write
NVIC_ISPR[0 - 7]	Interrupt Set-Pending Registers	read-write
NVIC_ICPR[0 - 7]	Interrupt Clear-Pending Registers	read-write
NVIC_IABR[0 - 7]	Interrupt Active Bit Register	read-only

1.5.1.5 MPU Registers

The MPU group contains registers for the Memory Protection Unit. It is present only if the MPU is enabled.

Table 1-8 MPU Registers

Name	Description	Type
MPU_TYPE	MPU Type register	read-only
MPU_CTRL	MPU Control register	read-write
MPU_RNR	MPU Region Number register	read-write
MPU_RBAR	MPU Base Address register	read-write
MPU_RASR	MPU Region Attribute register	read-write

1.5.1.6 SysTick Registers

The SysTick group contains information about the System Timer.

Table 1-9 SysTick Registers

Name	Description	Type
SYST_CSR	SysTick Control and Status Register- 0xE000E010	read-write ¹
SYST_RVR	SysTick Reload Value Register - 0xE000E014	read-write
SYST_CVR	SysTick Current Value Register - 0xE000E018	read-write
SYST_CALIB	SysTick Calibration Value Register - 0xE000E01C	read-only

1. Contains fields that are read-only. For more information, refer to the Arm documentation on the register.

1.5.1.7 FPB Registers

The FPB group contains registers pertaining to the hardware breakpoints.

Table 1-10 FPB Registers

Name	Description	Type
FP_CTRL	FP_CTRL register	read-write
FP_REMAP	FP_REMAP register	read-write
FP_COMP1	FlashPatch Comparator Register 1 - 0xE000200C	read-write
FP_COMP2	FlashPatch Comparator Register 2 - 0xE0002010	read-write
FP_COMP3	FlashPatch Comparator Register 3 - 0xE0002014	read-write
FP_COMP4	FlashPatch Comparator Register 4 - 0xE0002018	read-write
FP_COMP5	FlashPatch Comparator Register 5 - 0xE000201C	read-write
FP_COMP6	FlashPatch Comparator Register 6 - 0xE0002020	read-write
FP_COMP7	FlashPatch Comparator Register 7 - 0xE0002024	read-write

1.5.1.8 DWT Registers

The DWT group contains registers pertaining to hardware watchpoints.

Table 1-11 DWT Registers

Name	Description	Type
DWT_CTRL	DWT Control Register - 0xE0001000	read-write
DWT_CTRL_CYCCNT	Cycle Count Register	read-write
DWT_CPICNT	DWT CPI Count Register - 0xE0001008	read-write
DWT_EXCCNT	DWT Exception Overhead Count Register - 0xE000100C	read-write
DWT_SLEPCNT	DWT Sleep Count Register - 0xE0001010	read-write
DWT_LSUCNT	DWT LSU Count Register - 0xE0001014	read-write
DWT_FOLDNCNT	DWT Folded-instruction Count Register - 0xE0001018	read-write
DWT_PCSR	DWT Program Counter Sample Register - 0xE000101C	read-only
DWT_COMP0	DWT Comparator Register 0 - 0xE0001020	read-write
DWT_COMP1	DWT Comparator Register 1 - 0xE0001030	read-write
DWT_COMP2	DWT Comparator Register 2 - 0xE0001040	read-write

Table 1-11 DWT Registers (continued)

Name	Description	Type
DWT_COMP3	DWT Comparator Register 3 - 0xE0001050	read-write
DWT_MASK0	DWT Mask Register 0 - 0xE0001024	read-write
DWT_MASK1	DWT Mask Register 1 - 0xE0001034	read-write
DWT_MASK2	DWT Mask Register 2 - 0xE0001044	read-write
DWT_MASK3	DWT Mask Register 3 - 0xE0001054	read-write
DWT_FUNCTION0	DWT Function Register 0 - 0xE0001028	read-write
DWT_FUNCTION1	DWT Function Register 1 - 0xE0001038	read-write
DWT_FUNCTION2	DWT Function Register 2 - 0xE0001048	read-write
DWT_FUNCTION3	DWT Function Register 3 - 0xE0001058	read-write

1.5.1.9 TCM Control Registers

The TCM Control group contains Tightly-Coupled Memory registers.

Table 1-12 TCM Control Registers

Name	Description	Type
ITCMCR	Instruction and Data Tightly-Coupled Memory Control Registers	read-write
DTCMCR		

1.5.1.10 VFP Registers

The VFP group contains the floating-point related registers. These registers are present only if the Cycle Model was configured with an FPU coprocessor.

Table 1-13 VFP Registers

Name	Description	Type
FPCCR	Context Control Register	read-only
FPCAR	Context Address Register	read-write
FPDSCR	Default Status Control Register	read-write
MVFR[0-2]	Media and VFP Feature Register	read only
S[0-31]	32-bit single-word registers	read-write
D[0-31]	64-bit registers	read-write

1.5.1.11 Cache Control Registers

The Cache Control group contains the L1 Cache Memory registers.

Table 1-14 Cache Control Registers

Name	Description	Type
CLIDR	Cache Level ID Register	read only
CTR	Cache Type Register	read only
CCSIDR	Cache Size ID Register	ready only
CSSELR	Cache Size Selection Register	read write
CACR	L1 Cache Control Register	read write

1.5.2 Run To Debug Point Feature

The “run to debug point” feature has been added to enhance Cycle Model debugging. This feature forces the processor into a coherent state called a “debug point”. When debugging, the Cycle Model is brought to the debug point automatically whenever a software breakpoint is hit (including single stepping). However, if a hardware breakpoint is reached, or the system is advanced by cycles within SoC Designer, the Cycle Model can get to a non-debuggable state. In this event, the *run to debug point* will advance the processor to the debug state. It does this by stalling the instruction within the decode stage and allowing all earlier instructions to complete. Once that has been accomplished, the Cycle Model will cause the system to stop simulating.

The run to debug point is available as a context menu item for the component within SoC Designer Simulator. It is also available in the disassembler view.

1.5.3 Memory Information

The SoC Designer memory space view gives a view of the memory as seen from a particular port. Table 1-15 describes the available memory space views.

Table 1-15 Memory Spaces

Name	Description	Address Range	Access Size (bits)	Number of Blocks
Memory	Debug reads and writes to data cache and main memory.	0x00000000 — 0x40000000	8	1
axi_m	Debug reads and writes to main memory.	0x00000000 — 0x40000000	8	1
ITCM	Debug reads and writes to Instruction TCM.	0x00000000 — 0x20000000	8	1
DTCM	Debug reads and writes to Data TCM.	0x20000000 — 0x40000000	8	1

1.5.4 Disassembly View

To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-M7 Cycle Model and select **View Disassembly...** from the context menu.

All CADI windows support breakpoints – when double-clicking on the proper location a red dot will indicate that a breakpoint is currently active. To remove the breakpoints simply double-click on the same location again.

1.6 Available Profiling Data

Profiling data is enabled, and can be viewed using the Profiling Manager, which is accessible from the **Debug** menu in the SoC Designer Simulator. Both hardware and software based profiling is available.

1.6.1 Hardware Profiling

Hardware profiling includes just the Core Events stream. The events supported by this stream are shown in Table 1-16.

Table 1-16 Cortex-M7 Profiling Events

Stream	Event Name	Description
Core Events	CPI	Instruction Cycle Count
	Exception	Exception Overhead Counter
	Sleep	Sleep Overhead Counter
	LSU	Load-Store Counter
	IT Fold	Folded Instruction Counter

1.6.2 Software Profiling

Software-based profiling is provided by SoC Designer. Profiling information is also available in the SoC Designer Profiler. See the user guide for SoC Designer for more information.

Third Party Software Acknowledgement

Arm acknowledges and thanks the respective owners for the following software that is used by our product:

- **ELF (Executable and Linking Format) Tool Chain Product**

Copyright (c) 2006, 2008-2012 Joseph Koshy

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

