Cortex®-M0 Cycle Model

Version 9.0.0

User Guide

Non-Confidential

ARM®

Copyright © 2016 ARM Limited. All rights reserved.
ARM DUI 1006B (ID100716)

Cortex-M0 Cycle Model
User Guide

Copyright © 2016 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
April 2016	A	Non-Confidential	Restamp; Release 8.1.0. r0p0.
November 2016	B	Non-Confidential	Release 9.0.0

Non-Confidential 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 Limited (“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 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, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

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 shall prevail.

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 signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. You must follow the ARM trademark usage guidelines http://www.arm.com/about/trademarks/guidelines/index.php.

Copyright © ARM Limited or its affiliates. All rights reserved.
ARM Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

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

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Contents

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Preface

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

About This Guide

This guide provides all the information needed to configure and use the Cortex-M0 Cycle Model in SoC Designer Plus.

Audience

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

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

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Conventions

This guide uses the following conventions:

Convention	Description	Example
`courier`	Commands, functions, variables, routines, and code examples that are set apart from ordinary text.	`sparseMem_t SparseMemCreate-New();`
italic	New or unusual words or phrases appearing for the first time.	*Transactors* 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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Further reading

The following publications provide information that relate directly to SoC Designer Plus:

- *SoC Designer Plus Installation Guide*
- *SoC Designer Plus User Guide*
- *SoC Designer Plus Models Reference*
- *SoC Designer Plus AHBv2 Protocol Bundle User Guide*

The following publications provide reference information about ARM® products:

- *Cortex-M0 Technical Reference Manual*
- *AMBA 3 AHB-Lite Overview*
- *AMBA Specification (Rev 2.0)*
- *AMBA AHB Transaction Level Modeling Specification*
- *Architecture Reference Manual*

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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Glossary

AMBA	*Advanced Microcontroller Bus Architecture*. 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	*Advanced High-performance Bus*. 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	*Advanced Peripheral Bus*. 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	*Advanced eXtensible Interface*. 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 Carbon Model Studio (or *Carbon compiler*) from an RTL design. The Cycle Model contains a cycle- and register-accurate model of the hardware design.
Carbon 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 Plus, Platform Architect, or Accellera SystemC for simulation.
CASI	*ESL API Simulation Interface*, is based on the SystemC communication library and manages the interconnection of components and communication between components.
CADI	*ESL API Debug Interface*, enables reading and writing memory and register values and also provides the interface to external debuggers.
CAPI	*ESL API Profiling Interface*, 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	*Electronic System Level*. 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	*Hardware Description Language*. A language for formal description of electronic circuits, for example, Verilog.
RTL	*Register Transfer Level*. 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	*Transaction adaptors*. You add transactors to your component to connect your component directly to transaction level interface ports for your particular platform.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Chapter 1

Using the Cycle Model Component in SoC Designer Plus

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

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.1 Cortex-M0 Functionality

The Cortex-M0 processor is a low-power processor that features low gate count, low interrupt latency, and low-cost debug. It is intended for deeply embedded applications that require fast interrupt response features. The processor implements the ARMv7-M architecture.

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.

- Fully Functional and Accurate Features
- Unsupported Hardware Features
- Features Additional to the Hardware

For details of the functionality of the hardware that the Cycle Model simulates, see the *Cortex-M0 Technical Reference Manual*.

1.1.1 Fully Functional and Accurate Features

The following features of the Cortex-M0 hardware are fully implemented in the Cortex-M0 Cycle Model:

- Cortex-M0 Integer Core
- NVIC – Nested Vectored Interrupt Controller
- WIC – Wakeup Interrupt Controller
- AHB-Lite: ICode, DCode, and System Bus Interfaces
- BPU– BreakPoint Unit
- DWT – Data Watchpoint and Trace
- ROM Table

1.1.1.1 Note on Clock-gating

The Cortex-M0 supports architectural clock-gating only. This is controlled by setting the `CLKGATE_PRESENT` parameter to `1` in the *M0.conf* configuration file before creating the Cycle Model. See the *Cortex-M0 Configuration and Sign-off Guide* for more information.

The Cycle Model *does not* currently support RTL clock-gating.

1.1.2 Unsupported Hardware Features

The following features of the Cortex-M0 hardware are not implemented in the Cortex-M0 Cycle Model:

- The following registers are not available to be read / written via debug transactions — for example, in the SoC Designer Plus Registers window, or by accessing them directly from your debugger:
  - Debug register:Debug Core Register Selector Register

  The functionality of these registers, however, does exist and can be accessed by software running on the virtual platform.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.1.3 Features Additional to the Hardware

The following features that are implemented in the Cortex-M0 Cycle Model to enhance usability do not exist in the Cortex-M0 hardware:

- Semihosting Support. Semihosting enables the target application to communicate with the host operating system. This is used for external time synchronization, file handling operations, console input/output, and similar functionality.
- 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” on page 22 for more information.

1.2 Adding and Configuring the SoC Designer Plus Component

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

- SoC Designer Plus Component Files
- Adding the Cycle Model to the Component Library
- Adding the Component to the SoC Designer Canvas

1.2.1 SoC Designer Plus Component Files

The component files are the final output from the Carbon Model Studio compile and are the input to SoC Designer Plus. 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 Plus. 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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

The provided component files are listed below:

Table 1-1 SoC Designer Plus Component Files

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

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

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 Cycle Model RTL configuration file, *M0.conf*, used to create the Cycle Model.

1.3 ESL Ports

This section describes the differences between the pins listed in the *ARM Cortex-M0 Technical Reference Manual* and those on the Cortex-M0 Cycle Model. Certain hardware pins have been converted to init-time Cycle Model parameters.

1.3.1 Available Component ESL Ports

Table 1-2 describes ports that have been added to the Cycle Model, such as clocks and resets required by SoC Designer Plus, or those created by wrapping multiple hardware pins into transactors. See the *Cortex-M0 Technical Reference Manual* for more information.

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	Direction	Type
ahb_m	AHB Lite Master See "AHB-Lite Transaction Master Ports" on page 16 for more information	Output	AHB-Lite Transaction Master
IRQ	This port connects to external interrupt signals. It can be anywhere from 1 to 240 bits wide based on the configuration used to create the Cycle Model. The value must indicate the interrupt number [NumIRQ..0] and the *extValue must indicate whether the IRQ line is asserted (*extValue=1) or deasserted (*extValue=0).	Input	Signal slave
NMI	Non-maskable interrupt input to the NVIC; where 1 is used to assert NMI, and 0 is used to deassert NMI request.	Input	Signal slave
clk-in	Input Clock port. This port must be explicitly connected to a clock master.	Input	Clock Generator
extSemi	Semihosting can be enabled by connecting this port to the SoC Designer Plus semihost component, contained in the SoC Designer Plus Standard Model Library (v3.0 or greater).	Output	Transaction master

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.3.2 AHB-Lite Transaction Master Ports

The *ahb_m* transaction master port implements the AMBA AHB-Lite interface. This transaction master port should be connected to an AHBv2 slaves using either an MxAHBv2 bus component (where one side is an AHB Lite Master and the other side is an AHB Lite Slave) or a PL301 in between. See the *SoC Designer Plus AHBv2 Protocol Bundle User Guide* for more information.

There are a few AHBv2 sideband signals defined specifically for the Cortex-M0. See the *AHBv2 Protocol Bundle User Guide* for details on AHB Cortex-M0 extension signals.

1.3.3 Clock Ports

clk_in is the clock port used to clock the core.

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 will be slightly different depending on the settings that you enabled in the configuration file (*M0.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	Runtime[1]
Align Waveforms	When set to *true*, waveforms dumped from the component are aligned with the SoC Designer Plus simulation time. The reset sequence, however, is not included in the dumped data. When set to *false*, the reset sequence is dumped to the waveform data, however, the component time is not aligned with the SoC Designer Plus time.	true, false	true	No
Carbon DB Path	Sets the directory path to the database file.	Not Used	empty	No
Dump Waveforms	Determines whether SoC Designer Plus dumps waveforms for this component.	true, false	false	Yes
Enable Debug Messages	Determines whether debug messages are logged for the component.	true, false	false	Yes

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Table 1-3 Component Parameters (continued)

Name	Description	Allowed Values	Default Value	Runtime[1]
Enable PC Tracing	This parameter is obsolete and should be left at its default setting.	N/A	false	No
ahb_m Align Data	Determines whether halfword and byte transactions will align data to the transaction size for this port. By default, data is not aligned.	true, false	false	No
ahb_m Big Endian	Determines whether AHB data is treated as big endian for this port. By default, data is not sent as big endian.	true, false	false	No
ahb_m Enable Debug Messages	Determines whether debug messages are logged for the *mem_D* port.	true, false	false	Yes
PC Tracing File	This parameter is obsolete and should be left at its default setting.	N/A	N/A	No
STCALIB	SysTime Clock Calibration	*Valid integer*	1048576	No
STCLKEN	SysTime Clock Enable	*True, False*	False	No
Waveform File [2]	Name of the waveform file.	*string*	CortexM0.fsdb	No
Waveform Timescale	Sets the timescale to be used in the waveform.	Many values in drop-down	1 ns	No

1. *Yes* means the parameter can be dynamically changed during simulation, *No* means it can be changed only when building the system, *Reset* means it can be changed during simulation, but its new value will be taken into account only at the next reset.
2. When enabled, SoC Designer Plus 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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.5 Debug Features

The Cortex-M0 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-M0 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
- SCS Registers
- NVIC Registers
- ROM Registers
- BPU Registers
- DWT Registers

See the *Cortex-M0 Technical Reference Manual* for detailed descriptions of these registers.

1.5.1.1 Core Registers

The Core group contains the ARM Architectural registers.

Table 1-4 Core Registers

Name	Description	Type
R0	R0 register	read-write [1]
R1	R1 register	read-write [1]
R2	R2 register	read-write [1]
R3	R3 register	read-write [1]
R4	R4 register	read-write [1]
R5	R5 register	read-write [1]
R6	R6 register	read-write [1]
R7	R7 register	read-write [1]
R8	R8 register	read-write [1]
R9	R9 register	read-write [1]

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Table 1-4 Core Registers (continued)

Name	Description	Type
R10	R10 register	read-write [1]
R11	R11 register	read-write [1]
R12	R12 register	read-write [1]
R13	R13/Stack Pointer (SP) register	read-write [1]
R14	R14/Link Register (LR)	read-write [1]
R15	R15/PC (Program Counter) Register	read-write [1]
XPSR	Program Status Register	read-write
PRIMASK	Mask Register	read-write
CONTROL	Control Register	read-write

1. Writeable at debuggable point only. Otherwise, a warning is printed.

1.5.1.2 SCS Registers

The SCS group provides access to the System Control Space.

Table 1-5 SCS Registers

Name	Description	Type
ICSR	Int Control State register	read-write
SYST_CSR	Sys Tick Control And Status register	read-write
SYST_RVR	Sys Tick Reload Value register	read-write
SYST_CVR	Sys Tick Current Value register	read-only
SYST_CALIB	Sys Tick Calibration Value register	read-only
CPUID	CPUID Base register	read-only
AIRCR	Application Interrupt Reset Control register	read-write
CCR	Config Control register	read-write
SHPR2	System Handlers Priority Register 2	read-only
SHPR3	System Handlers Priority Register 3	read-only
SHCSR	System Handler Control And State register	read-write
DFSR	Debug Fault Status Register	read-write
DHCSR	Debug Halting Control and Status Register	read-write
DCRSR	Debug Core Register Selector Register	read-write
DCRDR	Debug Core Register Data Register	read-write
DEMCR	Debug Exception and Monitor Control Register	read-write
SCS_PID0	System Control Space Peripheral ID Register 0	read-only
SCS_PID1	System Control Space Peripheral ID Register 1	read-only
SCS_PID2	System Control Space Peripheral ID Register 2	read-only

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Table 1-5 SCS Registers (continued)

Name	Description	Type
SCS_PID3	System Control Space Peripheral ID Register 3	read-only
SCS_PID4	System Control Space Peripheral ID Register 4	read-only
SCS_CID0	System Control Space Control ID Register 0	read-only
SCS_CID1	System Control Space Control ID Register 1	read-only
SCS_CID2	System Control Space Control ID Register 2	read-only
SCS_CID3	System Control Space Control ID Register 3	read-only

1.5.1.3 NVIC Registers

The NVIC group contains registers that control the interrupt controller state.

Table 1-6 NVIC Registers

Name	Description	Type
ISER	Interrupt Set-Enable register	read-write
ICER	Interrupt Clear-Enable register	read-write
ISPR	Interrupt Set-Pending register	read-write
ICPR	Interrupt Clear-Pending register	read-write
IPR0-7	Interrupt Priority registers 0 through 7	read-write

1.5.1.4 ROM Registers

The ROM group contains registers for the Read Only Memory when a debugger is attached.

Table 1-7 ROM Registers

Name	Description	Type
ROM_SCS	ROM System Control Space	read-only
ROM_DWT	ROM Debug Watchpoint Register	read-only
ROM_BPU	ROM Breakpoint Unit Register	read-only
ROM_EOT	ROM End of Table Register	read-only
ROM_CSMT	ROM Statement Register	read-only
ROM_PID0-4	ROM Peripheral ID Registers 0 through 4	read-only
ROM_CID0-3	ROM Component ID Register 0-3	read-only

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.5.1.5 BPU Registers

The BPU group contains registers pertaining to the hardware breakpoints.

Table 1-8 BPU Registers

Name	Description	Type
BP_CTRL	Breakpoint Control register	read-write
BP_COMP0-3	Breakpoint Comparator Register 0-3	read-write
BP_PID0-4	Breakpoint Peripheral Register 0-4	read-write
BP_CID0-3	Breakpoint Component ID Registers 0-3	read-write

1.5.1.6 DWT Registers

The DWT group contains registers pertaining to hardware watchpoints.

Table 1-9 DWT Registers

Name	Description	Type
DWT_CTRL	Watchpoint Control Register	read-write
DWT_PCSR	Watchpoint Counter Sample Register	read-write
DWT_COMP0	Watchpoint Comparator Register	read-write
DWT_MASK0	Watchpoint Mask 0 register	read-write
DWT_FUNCTION0	Watchpoint Function 0 register	read-write
DWT_COMP1	Watchpoint Comparator 1 register	read-write
DWT_MASK1	Watchpoint Mask 1 register	read-write
DWT_FUNCTION1	Watchpoint Function 1 register	read-write
DWT_PERIPHID0-3	Watchpoint Peripheral Registers 0-4	read-write
DWT_PCELLID0-3	Watchpoint Primecell Registers 0-3	read-write

The values shown for the DWT registers will only be valid if the Cortex-M0 is configured with the *dbgPresent* value set to the highest value (1). These values are set in the *M0.conf* file when the Cycle Model was generated. Also, the DWT must be enabled via the debug exception and monitor control register (TRCENA).

If any of these conditions are false, the values shown should not be considered valid.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.5.2 Run To Debug Point Feature

The “run to debug point” feature has been added to enhance Cycle Model debugging. The Cortex-M0 processor is a dual issue out of order completion machine. This means that while the processor is running it does not present a coherent programmer’s view state; instructions in the pipeline may be in different execution states.

This feature forces the processor into a coherent state called “run to 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 Plus, 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

Each memory space represents a different view of memory using a page table. The Cortex-M0 processor memory spaces are selectable using the Space: pulldown menu in the Memory view, and the Memory space pulldown menu in the Disassembly view (see the *SoC Designer Plus User Guide* for more information).

1.5.4 Disassembly View

The Cortex-M0 Cycle Model supports a disassembly view of a program running on the Cycle Model in SoC Designer Simulator. To display the disassembly view in the SoC Designer Simulator, right-click on the Cortex-M0 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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

1.6 Available Profiling Data

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

1.6.1 Hardware Profiling

There are no hardware profiling capabilities within the Cortex-M0 core.

1.6.2 Software Profiling

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

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

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.

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential

Copyright © 2016 ARM Limited. All rights reserved.
Non-Confidential