ARM[®] CoreSight[®] MTB-M23

Revision: r0p0

Technical Reference Manual



Copyright © 2016 ARM. All rights reserved. ARM DDI 0564C (ID112116)

ARM CoreSight MTB-M23 Technical Reference Manual

Copyright © 2016 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
25 April 2016	А	Confidential	First release for r0p0
29 July 2016	В	Confidential	Second release for r0p0
18 November 2016	С	Non-Confidential	Third release for r0p0

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, third party 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 signed written agreement covering this document with ARM, then the 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.

Words and logos marked with ® or TM 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. Please follow ARM's trademark usage guidelines at , http://www.arm.com/about/trademark-usage-guidelines.php

Copyright ©, ARM Limited or its affiliates. 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 ARM CoreSight MTB-M23 Technical Reference Manual

	Prefa	ace	
		About this book Feedback	vi iх
Chapter 1	Intro	oduction	
-	1.1	About the CoreSight MTB-M23	1-2
	1.2	Compliance	1-3
	1.3	Features	1-4
	1.4	Interfaces	1-5
	1.5	Configurable options	1-6
	1.6	Test features	1-7
	1.7	Product documentation and design flow	1-8
	1.8	Product revisions	1-10
Chapter 2	Fund	ctional description	
	2.1	About the functions	
	2.2	Interfaces	2-3
	2.3	Operation	2-6
Chapter 3	Prog	grammers model	
-	3.1	About the programmers model	3-2
	3.2	Memory model	
	33	Register summary	3-4
	3.5		•••••••••••••••••••••••••••••••••••••••

Appendix A	Signa	al descriptions	
••	A.1	About the signal descriptions	A-2
	A.2	Clock, reset, and control signals	A-3
	A.3	AMBA AHB interface	A-4
	A.4	SRAM memory interface	A-6
	A.5	Execution trace interface	A-7
	A.6	External trace control interface	A-8
	A.7	Debug authentication interface	A-9
	A.8	Miscellaneous signals	A-10
Appendix B	Exam	ple Programming Sequences	
••	B.1	Discovery	B-2
	B.2	Trace Enable Programming Sequence	B-3
	B.3	Trace Disable Programming Sequence	B-4
Appendix C	Revis	sions	

Preface

This preface introduces the *ARM*[®] *CoreSight*[™] *MTB-M23 Technical Reference Manual*. It contains the following sections:

- *About this book* on page vii.
- *Feedback* on page ix.

About this book

This book is for the CoreSight Micro Trace Buffer for the Cortex-M23 processor, the CoreSight MTB-M23 macrocell.

Product revision statu		
	The rnpn identifier indicates the revision statrnIdentifies the major revision ofpnIdentifies the minor revision or	us of the product described in this book, where: the product. modification status of the product.
Intended audience		
	 This book is written for: System designers, system integrators, a Software developers who want to use the 	nd verification engineers. he MTB.
Using this book		
	This book is organized into the following cha	pters:
	Chapter 1 Introduction	
	Read this chapter for an introduced	action to the MTB and its features.
	Chapter 2 Functional description	
	Read this chapter for a descript	ion of the functionality of the MTB.
	Chapter 3 Programmers model	
	Read this chapter for a descript information for programming t	ion of the MTB programmable registers and he MTB.
	Appendix A Signal descriptions	
	Read this for a description of the	ne signals in the MTB.
	Appendix B Example Programming Seque	ices
	Read this for a description of so MTB.	ome example programming sequences for the
	Appendix C <i>Revisions</i> Read this for a description of th book.	e technical changes between released issues of this

Glossary

The *ARM*[®] *Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM*[®] *Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See ARM[®] Glossary ,
http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

Conventions

This book uses the conventions that are described in:

Typographical conventions.

Typographical conventions

The following table describes the typographical conventions:

Style	Purpose
italic	Introduces special terminology, denotes cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<u>mono</u> space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
monospace italic	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.
<and></and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <rd>, <crn>, <crm>, <opcode_2></opcode_2></crm></crn></rd>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>ARM glossary</i> . For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- ARM[®]v8-M Architecture Reference Manual (ARM DDI 0553).
- ARM[®] Cortex[®]-M23 Processor Technical Reference Manual (ARM DDI 0550).
- ARM[®] CoreSight[™] Architecture Specification v2.0 (ARM IHI 0029).
- ARM[®] AMBA[®] 5 AHB Protocol Specification, AHB5, AHB-Lite (ARM IHI 0033).

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title.
- The number, ARM DDI 0564C.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

— Note —

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1 Introduction

This chapter introduces the CoreSight *Micro Trace Buffer* (MTB) for the Cortex-M23 processor and its features. It contains the following sections:

- *About the CoreSight MTB-M23* on page 1-2.
- *Compliance* on page 1-3.
- *Features* on page 1-4.
- *Interfaces* on page 1-5.
- *Configurable options* on page 1-6.
- *Test features* on page 1-7.
- *Product documentation and design flow* on page 1-8.
- *Product revisions* on page 1-10.

1.1 About the CoreSight MTB-M23

The CoreSight MTB-M23 provides a simple execution trace capability to the Cortex-M23 processor. The MTB provides a lighter option for instruction trace requirements for software development. Unlike the *Embedded Trace Macrocell* (ETM) or the *Program Trace Macrocell* (PTM) trace solutions, the MTB does not require dedicated trace connection. However, the amount of trace history provided by the MTB is limited by the size of SRAM allocated for trace operations.

1.2 Compliance

The MTB complies with, or implements, the specifications described in:

- Advanced Microcontroller Bus Architecture.
- Debug authentication interface.

This TRM complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

1.2.1 Advanced Microcontroller Bus Architecture

This MTB complies with the AMBA 5 AHB protocol. See the *ARM*[®] *AMBA*[®] 5 *AHB Protocol Specification, AHB5, AHB-Lite.*

1.2.2 Debug authentication interface

This MTB complies with the CoreSight authentication interface. See the *ARM*[®] *AMBA*[®] *5 AHB Protocol Specification, AHB5, AHB-Lite.*

1.3 Features

The MTB features and benefits are:

- Provision of program flow tracing for the Cortex-M23 processor.
- Very small area.
- Power reduction features.
- MTB SRAM can be used for both trace and general-purpose storage by the processor.
- MTB SRAM size is configurable at implementation time.
- The position and size of the trace buffer in SRAM is configurable by software.
- External hardware can control trace start/stop.
- CoreSight compliant.

1.4 Interfaces

The three main interfaces on the MTB are:

- *AHB-Lite slave interface.*
- *Processor execution trace interface.*
- Synchronous SRAM master interface.

1.4.1 AHB-Lite slave interface

The MTB AHB-Lite interface provides access to two memory regions for:

- The Special Function Registers (SFRs).
- The SRAM.

HSELSFR selects the address space for the Special Function Registers and **HSELRAM** selects the address space for the SRAM. See *AMBA AHB interface* on page A-4 for more information.

1.4.2 Processor execution trace interface

The processor execution trace interface transfers the execution information from the processor to the MTB. See *Execution trace interface* on page A-7 for more information. The MTB formats information, received from this interface, into trace packets and writes it into the MTB SRAM.

1.4.3 Synchronous SRAM master interface

The SRAM interface connects to the MTB SRAM, that is used to store the trace packet information formatted by the MTB.

The MTB also operates as a simple AHB-Lite SRAM bridge. The processor has read and write access to the entire MTBSRAM address space using the AHB-Lite interface. This enables the processor to access the trace packet information and also to store program and data information in the SRAM.

The SRAM interface requires single cycle access to the SRAM and does not support wait states.

1.5 Configurable options

Table 1-1 shows the MTB configurable options available at implementation time.

Feature	Configurable option
SRAM address width (MTBAWIDTH) ^a	5-32 bits.
Reset all registers	Present or absent.
Number of watchpoint comparators	0, 1, 2, 3, 4.
a. Because the SRAM interface is 32 width of the SRAM address bus is	2 bits wide, the actual 3 MTBAWIDTH-2. See

Table A-3 on page A-6.

Table 1-1 MTB configurable options

1.6 Test features

The MTB has no test features, because the necessary *Design For Test* (DFT) logic is inserted automatically during implementation by your EDA tools.

1.7 Product documentation and design flow

This section describes the MTB books and how they relate to the design flow. It includes:

- Documentation.
- Design flow.

See *Additional reading* on page viii for more information about the books described in this section. For information on the relevant protocols, see *Compliance* on page 1-3.

1.7.1 Documentation

The MTB documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the MTB. It is required at all stages of the design flow. The choices made in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the MTB, then contact:

- The implementer to determine:
 - The build configuration of the implementation.
 - What integration, if any, was performed before implementing the MTB.
- The integrator to determine the pin configuration of the device that you are using.

1.7.2 Design flow

The MTB is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a netlist.

Integration

The integrator connects the implemented design into a SoC. This includes connecting it to a memory, processor, and AHB-Lite bus.

Programming

This is the last process. The tools developer creates the software required to configure and initialize the MTB, and tests the required debug software.

Each process:

- Can be performed by a different party.
- Can include implementation and integration choices affect the behavior and features of the MTB.

Typically, a single design team integrates the MTB into a SoC before synthesizing the complete design. Alternatively, the team can synthesize the MTB on its own or partially integrated, to produce a macrocell that is then integrated, possibly by a separate team.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell.

Configuration inputs

The integrator configures some features of the MTB by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

The programmer configures the MTB by programming particular values into registers. This affects the behavior of the MTB.

— Note —

This manual refers to implementation-defined features that are applicable to build configuration options. Reference to a feature that is included means that the appropriate build and pin configuration options are selected. Reference to an enabled feature means one that has also been configured by software.

1.8 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2 Functional description

This chapter describes the functionality of the MTB. It contains the following sections:

- *About the functions* on page 2-2.
- Interfaces on page 2-3.
- *Operation* on page 2-6.

2.1 About the functions

Figure 2-1 shows the main interfaces on the MTB and how they are connected in a simple Cortex-M23 based system.



Figure 2-1 MTB system diagram

When enabled, the MTB records changes in program flow, reported by the Cortex-M23 processor over the execution trace interface. This information is stored as trace packets in the SRAM. An off-chip debugger can extract the trace information using the DAP to read the trace information from the SRAM, over the AHB-Lite interface. The debugger can then reconstruct the program flow from this information.

The processor accesses the SRAM using the AHB-Lite interface. The MTB simultaneously stores trace information into the SRAM, and gives the processor access to the SRAM. The MTB ensures that trace write accesses have priority over processor accesses.

The MTB does not:

- Include any form of load/store data trace capability.
- Include tracing of any other information.

2.2 Interfaces

This section describes the MTB interfaces. It contains:

- *Clock and reset interface* on page 2-4.
- *AHB-Lite interface* on page 2-4.
- *Execution trace interface* on page 2-4.
- *External trace enable interface* on page 2-5.
- SRAM interface on page 2-5.
- *Debug authentication interface* on page 2-5.

Figure 2-2 shows the interface details.



Figure 2-2 MTB interfaces

2.2.1 Clock and reset interface

The clock and reset interface consists of the HCLK, RESETn, and IDLE signals. The IDLE signal can be used to gate HCLK outside the MTB to save power. See *Clock, reset, and control signals* on page A-3 for additional information.

2.2.2 AHB-Lite interface

The following sections describe the AHB-Lite interface used by the MTB:

- Wait state behavior.
- Buffering.

The MTB AHB-Lite interface provides access to two memory regions, one for the *Special Function Registers* (SFR) and the other for the SRAM. The memory regions are selected by the **HSELSFR** and **HSELRAM** inputs respectively. Only one of these select inputs can be HIGH at a time. If they are both HIGH at the same time the behavior is UNPREDICTABLE.

SFR accesses must be word accesses. SRAM accesses can be either byte or word accesses.

See the ARM[®] AMBA[®] 5 AHB Protocol Specification, AHB5, AHB-Lite for more information.

Wait state behavior

SRAM accesses from the AHB-Lite interface occur with zero wait states when there is no trace data being written to the SRAM and no error is reported (**HRESP** is LOW).

Faults happen when User access is performed to SFR or SRAM and not permitted.

Trace packet write access to the SRAM takes priority over access from the AHB-Lite interface. Therefore, one or more wait states can be inserted into the AHB-Lite accesses if trace information is simultaneously written to the SRAM.

SFR accesses always occur with zero wait states when no error is reported (HRESP is LOW).

Buffering

The MTB AHB-Lite interface can buffer two address phases and one data write phase for SRAM accesses. This enables:

- AHB-Lite accesses to the SRAM to occur at the same time as a trace packet is written to the SRAM.
- AHB-Lite read access to the SRAM to follow an AHB-Lite write access without inserting wait states.
- AHB-Lite write access to be adapted to the SRAM interface protocol.

2.2.3 Execution trace interface

The execution trace interface consists of the IAXEN, IAEXSEQ, IAEX[30:0], ATOMIC, and EDBGRQ signals.

Optionally, you can program the MTB to use the **EDBGRQ** output to request the processor to enter the halt state when the trace buffer is full. This avoids loss of trace information. See *MTB_FLOW Register* on page 3-11.

2.2.4 External trace enable interface

This interface enables external control when tracing starts and stops. It consists of the **TSTART** and **TSTOP** signals and the **DWTMATCH** bus. See *Trace start and stop* on page 2-7 and *External trace control interface* on page A-8.

2.2.5 SRAM interface

This is a synchronous interface to the SRAM. The MTB uses this interface for trace and AHB-Lite accesses to the SRAM. See *MTB execution trace packet format* on page 2-6 and *SRAM memory interface* on page A-6.

2.2.6 Debug authentication interface

This interface permits trace to be disabled for security purposes. See also *Debug authentication interface* on page A-9 for signal information.

Table 2-1 shows a summary of the authentication behavior.

Table 2-1 Debug authentication interface behavior

Non-secure Non-invasive debug enable ^a	Secure Non-invasive debug enableª	Processor state	Execution trace enabled	SRAM access allowed from AHB	SFR access allowed from AHB
0	-	-	No	Yes	Yes
1	0	Non-secure	Yes	Yes	Yes
1	0	Secure	No	Yes	Yes
1	1	-	Yes	Yes	Yes

a. Defined by the ARMv8-M architecture. See the ARM[®]v8-M Architecture Reference Manual.

— Note ———

- CoreSight MTB-M23 supports only the DBGEN and NIDEN signals which are not directly connected to the top level. The processor changes their values depending on:
 - The Secure state of the processor.
 - The authentication signals.
- Execution trace can be enabled, but trace may not be sent out. For example, if Secure debug is not permitted, no trace is sent when the core is running in Secure, but trace is sent when the core is running in Non-secure state.

2.3 Operation

This section describes the operation of the MTB. It contains the following sections:

- MTB execution trace packet format.
- *Trace start and stop* on page 2-7.

2.3.1 MTB execution trace packet format

•

The execution trace packet consists of a pair of 32-bit words that the MTB generates when it detects the processor PC value changes non-sequentially. A non-sequential PC change can occur during branch instructions or during exception entry.

— Note — —

The processor can cause a trace packet to be generated for any instruction.

Figure 2-3 shows the signal combination that detects a non-sequential PC change.



Figure 2-3 MTB non-sequential PC diagram

— Note —

IAEXEN and IAEXSEQ are internal signals of the Cortex-M23 processor.

Figure 2-4 shows how the execution trace information is stored in memory as a sequence of packets.



Figure 2-4 MTB execution trace storage format

The first, lower addressed, word contains the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb[®] instructions are halfword-aligned. The least significant bit of the value is the A-bit. The A-bit indicates the atomic state of the processor at the time of the branch, and can differentiate whether the branch originated from an instruction in a program, an exception, or a PC update in Debug state. When it is zero, the branch originated from an instruction. When it is one, the branch originated from an exception or PC update in Debug state. This word is always stored at an even word location.

The second, higher addressed, word contains the address it branched to. The value stored only records bits[31:1] of the branch address. The least significant bit of the value is the S-bit. The S-bit indicates where the trace started. An S-bit value of 1 indicates where the first packet after the trace started and a value of \emptyset is used for other packets. Because it is possible to start and stop tracing multiple times in a trace session, the memory might contain several packets with the S-bit set to 1. This word is always stored in odd word alignment with the next higher memory address.

When the A-bit is set to 1, the source address field contains the architecturally preferred return address for the exception. For example, if an exception was caused by an SVC instruction, then the source address field contains the address of the following instruction. This is different from the case where the A-bit is set to \emptyset . In this case, the source address contains the address of the branch instruction.

For an exception return operation, two packets are generated:

- The first packet has the:
 - Source address field set to the address of the instruction that causes the exception return, BX or POP.
 - Destination address field set to bits[31:1] of the EXC_RETURN value. See the ARM[®]v8-M Architecture Reference Manual.
 - The A-bit set to 0.
- The second packet has the:
 - Source address field set to bits[31:1] of the EXC_RETURN value.
 - Destination address field set to the address of the instruction where execution commences.
 - A-bit set to 1.

2.3.2 Trace start and stop

Tracing is enabled when the MTB_MASTER.EN bit in the MTB_MASTER Trace Control Register is 1. There are various ways to set the bit to 1 to start tracing, or to 0 to stop tracing. See *MTB_MASTER Register* on page 3-9.

You can control trace externally using the **TSTART** and **TSTOP** signals or using the MTB_MASTER Trace Control Register, see *TSTART and TSTOP signals* on page 2-8. You can program the MTB to stop tracing automatically when the memory fills to a specified watermark level or you can start or stop tracing by writing directly to the MTB_MASTER.EN bit, see *MTB_MASTER Register* on page 3-9. If you do not use the watermark mechanism, and the trace buffer overflows, then the buffer wraps around overwriting previous trace packets.

You can also control trace externally using the **DWTMATCH** bus. The **DWTMATCH** bus is connected to the match output of the DWT comparators, see *Trace start and stop using the DWTMATCH bus* on page 2-9.

If more than one source attempts to modify the MTB_MASTER.EN bit value in the same cycle, then the following priority order is used to determine which value is accepted. The lowest number is the higher priority.

- 1. Watermark stop.
- 2. Software write to the MTB_MASTER.EN bit.
- 3. MTB_MASTER.TSTARTEN, **TSTART**, or **MTB_TSTART** on the DWTMATCH bus.
- 4. MTB_MASTER.TSTOPEN, **TSTOP**, or **MTB_TSTOP** on the DWTMATCH bus.

— Note —

- If you use the watermark auto stop feature to stop trace, you cannot restart trace until software clears the watermark auto stop. You can achieve this in one of the following ways:
 - By setting the MTB_POSITION.POINTER field to point to the beginning of the trace buffer. See MTB_POSITION Register on page 3-8.
 - By setting the MTB_FLOW.AUTOSTOP bit to 0. See MTB_FLOW Register on page 3-11.
- If tracing is stopped while a trace packet is being written into memory, the MTB ensures that the packet write is completed.
- The **DBGEN** or **NIDEN** signal must be HIGH for tracing to occur. See *Debug authentication interface* on page 1-3. After tracing has started the MTB stops tracing when both of these signals are LOW. The value of the MTB_MASTER.EN bit is not affected by these signals. If the **DBGEN** or **NIDEN** signal is HIGH again, then tracing continues and the S-bit is set to 1 in the first trace packet when tracing resumes. There is no restriction on which methods are used to start and stop tracing and they can be combined in any way. For example, the **TSTART** signal can start the trace and the MTB_MASTER.EN bit can stop the trace.
- When Secure debug is not permitted, the MTB does not trace Secure instructions, only Non-secure.

TSTART and TSTOP signals

When the **TSTART** signal is HIGH for one or more clock cycles and the MTB_MASTER.TSTARTEN bit is 1, then the MTB_MASTER.EN bit is set to 1 and tracing starts from the memory location indicated by the MTB_POSITION.POINTER field in the Position Trace Control Register. See *MTB_MASTER Register* on page 3-9 and *MTB_POSITION Register* on page 3-8. Tracing continues when the **TSTART** signal is LOW, until stopped.

When the **TSTOP** signal is HIGH for one or more cycles and the MTB_MASTER.EN bit is 1, then the MTB_MASTER.EN bit is set to 0 and the trace stops.

_____Note _____

If TSTART and TSTOP signals are both HIGH, in the same cycle, TSTART takes priority.

Trace start and stop using control registers

You can use the control registers to start or stop trace as follows:

- Tracing is started when the MTB_MASTER.EN bit is set to 1.
- Tracing can be stopped automatically using the MTB_FLOW Trace Control Register. See *MTB FLOW Register* on page 3-11.
- Tracing can also be stopped under software control by setting the MTB_MASTER.EN bit to 0 in the MTB_MASTER Trace Control Register, see *MTB_MASTER Register* on page 3-9.

Trace start and stop using the DWTMATCH bus

You can use the DWTMATCH bus to start or stop trace as follows:

- Program the MTB_TSTART register to select the DWT comparators that start the trace when a match is detected, see *MTB_TSTART Register* on page 3-13. This sets the MTB_MASTER.EN bit, see *MTB_MASTER Register* on page 3-9.
- Program the MTB_TSTOP register to select the DWT comparators that stop the trace when a match is detected, see *MTB_TSTOP Register* on page 3-14. This clears the MTB_MASTER.EN bit, see *MTB_MASTER Register* on page 3-9.

Trace efficiency

To achieve low area and low power, the MTB trace format is less efficient than the ARM ETM or PTM solutions. Each branch requires eight bytes, resulting in 128 branches per kilobyte of RAM. Based on Dhrystone 2.1, this is approximately 875 instructions per kilobyte, or using a zero-wait state memory system, approximately 1600 **HCLK** cycles per kilobyte, or approximately 2.5 loops of Dhrystone 2.1.

Chapter 3 Programmers model

This chapter describes the MTB registers and provides information for programming the MTB. It contains the following sections:

- *About the programmers model* on page 3-2.
- *Memory model* on page 3-3.
- *Register summary* on page 3-4.
- *Register descriptions* on page 3-8.

3.1 About the programmers model

The following information applies to the MTB registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in UNPREDICTABLE behavior.
- The behavior of the MTB is UNPREDICTABLE if the registers with UNKNOWN reset values are not programmed prior to enabling trace.
- Unless otherwise stated:
 - Do not modify reserved register bits.
 - Ignore reserved register bits on reads.
 - All register bits are reset to a logic 0 by a system or power-on reset.
 - Use only word size, 32-bit, transactions to access all registers.
- Access type in Table 3-1 on page 3-4 is described as follows:
 - **RW** Read and write.
 - **RO** Read only.
 - **WO** Write only.

3.2 Memory model

The MTB has a 4KB *Special Function Register* (SFR) memory region. This contains the trace control registers and CoreSight registers.

The base address of the MTB SFR region, in the processor memory map, must be present in a CoreSight ROM table to permit a debug agent to determine the location of the CoreSight registers.

Figure 3-1 shows the SFR memory map with the address offsets for each set of registers.

0xFFC 0xF00	CoreSight Registers
0xEFC	Reserved
0x018	UKN/SBZP
0x014 0x000	Trace Control Registers

Figure 3-1 MTB special function register memory map

3.3 Register summary

This section summarizes the MTB registers in:

- Trace control registers.
- *CoreSight registers* on page 3-5.

3.3.1 Trace control registers

The MTB has programmable registers to control the behavior of the trace features, MTB_POSITION, MTB_MASTER, MTB_FLOW, MTB_BASE, MTB_TSTART, and MTB_TSTOP.

Table 3-1 shows the registers in offset order from the base memory address 0xE0043000.

Offset	Name	Туре	Reset ^a	Width	Description
0x000	MTB_POSITION	RW	-	32	MTB_POSITION Register on page 3-8
0x004	MTB_MASTER	RW	-	32	MTB_MASTER Register on page 3-9
0x008	MTB_FLOW	RW	-	32	MTB_FLOW Register on page 3-11
0x00C	MTB_BASE	RO	-	32	MTB_BASE Register on page 3-12
0x010	MTB_TSTART	RW	-	32	MTB_TSTART Register on page 3-13
0x014	MTB_TSTOP	RW	-	32	MTB_TSTOP Register on page 3-14

Table 3-1 Register summary

a. See register descriptions for reset values.

— Note —

The behavior of the trace functionality is UNPREDICTABLE if the MTB_POSITION and MTB_FLOW registers are not programmed prior to enabling trace, that is when software writes a 1 to either the MTB_MASTER.EN or MTB_MASTER.TSTART bit. See *Trace start and stop* on page 2-7.

3.3.2 CoreSight registers

Table 3-2 on page 3-6 shows the CoreSight registers and their values.

Table 3-2 CoreSight registers

Address offset	Register	Value	Description
0XFF0	Component ID0	0×000000D	See the ARM $^{\text{\tiny{(8)}}}$ CoreSight TM Architecture
0XFF4	Component ID1	0×0000090	<i>Specification v2.0</i> for more information about the registers and access types.
0XFF8	Component ID2	0×0000005	
0XFFC	Component ID3	0x00000B1	
0xFE0	Peripheral ID0	0×0000020	
0xFE4	Peripheral ID1	0x00000BD	-
0xFE8	Peripheral ID2	0×000000B	-
0xFEC	Peripheral ID3[31:8]	0×000000	-
	Peripheral ID3[7:4]	ECOREVNUM	-
	Peripheral ID3[3:0]	0x0	-
0xFD0	Peripheral ID4	0x00000004	-
0xFD4	Peripheral ID5	0×0000000	
0xFD8	Peripheral ID6	0x0000000	-
0xFDC	Peripheral ID7	0×0000000	-
0xFCC	Device Type Identifier	0x0000031	-
	Reserved [31:8]	0x000000	-
	Sub-Type [7:4]	0x3 (Basic Trace Router)	-
	Class [3:0]	Øx1 (Trace Sink)	-
0xFC8	Device Configuration	MTBAWIDTH-1 ^a	-
0xFBC	Device Architecture	0x47710A31	-
	Architect [31:21]	0x23B (ARM)	-
	Reserved [20]	0x1	-
	Revision [19:16]	0x1 (MTB Architecture v1.1)	-
	Architecture ID [15:0]	0x0A31 (Basic Trace Router Architecture, Major Architecture Revision 0)	-
0xFB8	Authentication Status [31:4]	0x000000	-
	Authentication Status [3]	1	
	Authentication Status [2]	NIDEN	-
	Authentication Status [1]	1	-
	Authentication Status [0]	DBGEN	-
0xFB4	Lock Status	0x0000000	
0xFB0	Lock Access	0x0000000	
0xFA4	Claim TAG Clear	0x0000000	-
-			

Copyright © 2016 ARM. All rights reserved. Non-Confidential a. To calculate the size of the SRAM use:

 $Size = 2^{(DEVID[4:0]+1)}$

The minimum allowed value stored is 4, which represents 32 bytes, see ARM®v8-M Architecture Reference Manual.

— Note ———

If you modify the MTB or incorporate parts of it into another module, then the Peripheral ID fields must be changed from the ARM values to partner-specific values. These fields are:

- JEP106 Identity (PID2[2:0], PID[7:4]).
- JEP Continuation Code (PID4[3:0]).
- Part Number (PID1[3:0], PID0[7:0]).
- Revision (PID2[7:4]).

You might also choose to change these fields:

- Part Number (PID1[3:0], PID0[7:0]).
- Revision (PID2[7:4]).
- Customer Modified (PID3[3:0]).
- RevAnd (PID3[7:4]).
- 4KB Count (PID4[7:4).

See the *ARM*[®] *CoreSight*[™] *Architecture Specification v2.0* for more information.

3.4 Register descriptions

This section describes the MTB trace control registers. CoreSight registers are described in the ARM^{\circledast} CoreSightTM Architecture Specification v2.0.

Table 3-1 on page 3-4 provides cross references to individual trace control registers.

3.4.1 MTB_POSITION Register

The MTB_POSITION Register characteristics are:

Purpose	Contains the trace write pointer and the wrap bit.	
Usage constraints	There are no additional usage constraints.	
Configurations	Available in all MTB configurations.	
Attributes	See Table 3-1 on page 3-4.	
	• You can modify all fields by software.	
	• Automatic hardware mechanisms update all fields.	

Figure 3-2 shows the MTB_POSITION Register bit assignments.



Figure 3-2 MTB_POSITION Register bit assignments

Table 3-3 shows the MTB_POSITION Register bit assignments.

Table 3-3 MTB_POSITION Register bit assignments

Bits	Name	Reset	Description
[31:3]	POINTER	UNKNOWN	Trace packet location pointer. Because a packet consists of two words, the POINTER field is the location of the first word of a packet. This field contains bits [31:3] of the address, in the SRAM, where the next trace packet will be written. The field points to an unused location and is automatically incremented.
			A debug agent can calculate the system address, on the AHB-Lite bus, of the SRAM location pointed to by the MTB_POSITION register using the following equation:
			system address = BASE + ((P + $(2^{\text{MTBAWIDTH}} - (\text{BASE MOD } 2^{\text{MTBAWIDTH}})))$ MOD $2^{\text{MTBAWIDTH}}$).
			Where $P = POSITION AND 0xFFFF_FFF8$.
			Where BASE is the MTB_BASE register value. See <i>MTB_BASE Register</i> on page 3-12.
			 Note
[2]	WRAP	UNKNOWN	This bit is set to 1 automatically when the POINTER value wraps as determined by the
			MTB_MASTER.MASK field in the MTB_MASTER Trace Control Register.
[1:0]	Reserved	UNKNOWN	These bits are reserved for future use and must be treated as UNK/SBZP.

A debug agent might use the WRAP bit to determine whether the trace information above and below the pointer address is valid.

3.4.2 MTB_MASTER Register

The MTB_MASTER Register characteristics are:

Purpose Contains:

P	
	• The main trace enable bit.
	• Other trace control fields.
Usage constraints	• Before the MTB_MASTER.EN or MTB_MASTER.TSTARTEN bits are set to 1, software must initialize the MTB_POSITION and MTB_FLOW registers.
	• If the MTB_FLOW.WATERMARK field is used to stop tracing or to halt the processor, the MTB_MASTER.MASK field must still be set to a value that prevents the MTB_POSITION.POINTER field from wrapping before it reaches the MTB_FLOW.WATERMARK value.
	• The EDBGRQ output value is also affected by the Debug authentication interface. See <i>Debug authentication interface</i> on

page 1-3.

Configurations	Available in all MTB configurations.				
Attributes	See Table 3-1 on page 3-4.				
	• You can modify all fields by software.				
	• Automatic hardware mechanisms update the EN and HALTREQ bits.				

Figure 3-3 shows the MTB_MASTER Register bit assignments.



Figure 3-3 MTB_MASTER Register bit assignments

Table 3-4 shows the MTB_MASTER Register bit assignments.

	Table 3-4 MTB	MASTER	Register bit	assignments
--	---------------	--------	---------------------	-------------

Bits	Name	Reset	Description
[31]	EN	0	Main trace enable bit. When this bit is 1 trace data is written into the SRAM memory location addressed by MTB_POSITION.POINTER. The MTB_POSITION.POINTER value auto increments after the trace data packet is written. The EN bit can be automatically set to 0 using the MTB_FLOW.WATERMARK field and the MTB_FLOW.AUTOSTOP bit. The EN bit is automatically set to 1 if the TSTARTEN bit is 1 and the TSTART signal is HIGH. The EN bit is automatically set to 0 if TSTOPEN bit is 1 and the TSTOP signal is HIGH.
			Note If the EN bit is set to 0 because the MTB_FLOW.WATERMARK field is set, then it is not automatically set to 1 if the TSTARTEN bit is 1 and the TSTART input is HIGH. In this case tracing can only be restarted if the MTB_FLOW.WATERMARK or MTB_POSITION.POINTER value is changed by software.
[30:10]	Reserved	UNKNOWN	These bits are reserved for future use and must be treated as UNK/SBZP.
[9]	HALTREQ	0	Halt request bit. This bit is connected to the halt request signal of the trace logic, EDBGRQ . When HALTREQ is set to 1, EDBGRQ is asserted if DBGEN is also HIGH. The HALTREQ bit can be automatically set to 1 using the MTB_FLOW.WATERMARK field. See <i>MTB_FLOW Register</i> on page 3-11 for more information.
[8]	RAMPRIV	0	SRAM Privilege bit. If this bit is 0, then User or Privileged AHB-Lite read and write accesses to the SRAM are permitted. If this bit is 1, then only Privileged AHB-Lite read and write accesses to the SRAM are permitted and User accesses are RAZ/WI. The HPROT[1] signal determines if an access is User or Privileged. See <i>AMBA 5 AHB Protocol Specification</i> or <i>AMBA AHB interface signals</i> on page A-4 for more information.
[7]	Reserved	0	This bit is reserved for future use and must be treated as RAZ/WI.

Table 3-4 MTB	_MASTER	Register bit	assignments	(continued)
---------------	---------	--------------	-------------	-------------

Bits	Name	Reset	Description
[6]	TSTOPEN	0	Trace stop input enable. If this bit is 1 and the TSTOP signal is HIGH, then the EN bit is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped.
[5]	TSTARTEN	0	Trace start input enable. If this bit is 1 and the TSTART signal is HIGH, then the EN bit is set to 1. Tracing continues until a stop condition occurs.
[4:0]	MASK	UNKNOWN	This value determines the maximum size of the trace buffer in SRAM. It specifies the most-significant bit of the MTB_POSITION.POINTER field that can be updated by automatic increment. If the trace tries to advance past this power of two, the MTB_POSITION.WRAP bit is set to 1, the MTB_POSITION.POINTER[MASK:0] bits are set to zero, and the MTB_POSITION.POINTER[MTBAWIDTH-4:MASK+1] bits remain unchanged. This field causes the trace packet information to be stored in a circular buffer of size 2 ^(MASK+4) bytes, that can be positioned in memory at multiples of this size. Valid values of this field are zero to MTBAWIDTH-4. Values greater than the maximum have the same effect as the maximum.

Table 3-5 shows some example MASK and POINTER values to illustrate the effect of the MASK field on the POINTER.

Table 3-5 MASK and POINTER examples

MASK	POINTER	POINTER + 1	POINTER next
0x0	0x1	0x2	0x0
0x0	0x5	0x6	0x4
0x3	0xF	0x10	0x0
0x3	0x1F	0x20	0x10

3.4.3 MTB_FLOW Register

The MTB_FLOW Register characteristics are:

Purpose	Contains:			
	• The WATERMARK address.			
	• The AUTOSTOP and AUTOHALT control bits.			
Usage constraints	There are no additional usage constraints.			
Configurations	Available in all MTB configurations.			
Attributes	See Table 3-1 on page 3-4.			
	You can modify all fields by software.			

Figure 3-4 on page 3-12 shows the MTB_FLOW Register bit assignments.

31					3	2	1	0
		WATI	ERMARK					
				Re	eserved-			
				AUT	OHALT -			
				AUT	OSTOP-			

Figure 3-4 MTB_FLOW Register bit assignments

Table 3-6 shows the MTB_FLOW Register bit assignments.

Table 3-6 MTB_FLOW Register bit assignments

Bits	Name	Reset	Description
[31:3]	WATERMARK	UNKNOWN	WATERMARK value. This field contains an address in the same format as the MTB_POSITION.POINTER field. When the MTB_POSITION.POINTER matches the WATERMARK field value, actions defined by the AUTOHALT and AUTOSTOP bits are performed.
[2]	Reserved	UNKNOWN	This bit is reserved for future use and must be treated as UNK/SBZP.
[1]	AUTOHALT	0	If this bit is 1 and WATERMARK is equal to MTB_POSITION.POINTER, then the MTB_MASTER.HALTREQ bit is automatically set to 1. If the DBGEN signal is HIGH, the MTB asserts this halt request to the Cortex-M23 processor by asserting the EDBGRQ signal.
[0]	AUTOSTOP	0	If this bit is 1 and WATERMARK is equal to MTB_POSITION.POINTER, then the MTB_MASTER.EN bit is automatically set to 0. This stops tracing.

— Note ———

- If you stop tracing, using the watermark auto stop feature, you cannot restart tracing until software clears the watermark auto stop. You can achieved this in one of the following ways:
 - Changing the MTB_POSITION.POINTER field value to point to the beginning of the trace buffer.
 - Setting the MTB_FLOW.AUTOSTOP bit to 0.
- A debug agent can use the AUTOSTOP bit to fill the trace buffer once only without halting the processor.
- A debug agent can use the AUTOHALT bit to fill the trace buffer once before causing the Cortex-M23 processor to enter the Debug state. When the request to enter Debug state is asserted, the processor may be doing a branch operation and this would trigger another trace packet. Therefore you must set the WATERMARK field below the final entry in the trace buffer region.

3.4.4 MTB_BASE Register

The MTB BASE Register characteristics are:

Purpose Indicates where the SRAM is located in the processor memory map. This register is provided to enable auto discovery of the MTB SRAM location, by a debug agent.

Usage constraints There are no additional usage constraints.

Configurations Available in all MTB configurations.

Attributes See Table 3-1 on page 3-4.

Figure 3-5 shows the MTB_BASE Register bit assignments.

31					0
		В	ASE		

Figure 3-5 MTB_BASE Register bit assignments

Table 3-7 shows the MTB_BASE Register bit assignments.

Table 3-7 MTB_BASE Register bit assignments

Bits	Name	Reset	Description
[31:0]	BASE	SRAMBASE	The value provided is the value of the SRAMBASEADDR[31:0] signal. See <i>Miscellaneous signals</i> on page A-10 for more information.

3.4.5 MTB_TSTART Register

The MTB_TSTART Register characteristics are:

Purpose	Configures MTB to start tracing from DWT triggers.
Usage constraints	Privileged access permitted only. Unprivileged accesses generate a fault. Word accessible only. Halfword and byte accesses are UNPREDICTABLE.
Configurations	Available in all MTB configurations.
Attributes	See Table 3-1 on page 3-4.

Figure 3-6 shows the MTB_TSTART Register bit assignments.

31					4 3	0
		Rese	erved			
			,,,,,,,			

СМРМАТСН —

Figure 3-6 MTB_TSTART Register bit assignments

Table 3-8 shows the MTB_TSTART Register bit assignments.

Table 3-8 MTB_TSTART Register bit assignments

Bits	Name	Reset	Description	
[31:4]	Reserved	0	These bits are reserved for future use and must be treated as UNK/SBZP.	
[3:0] ^a	CMPMATCH	0	When a DWT raises a CMPMATCH event on a comparator, and the associated CMPMATCH field is:	
			 Trigger a trace start event. The DWT event is ignored. 	

a. The number of bits in CMPMATCH depend on the number of the DWT comparators.

3.4.6 MTB_TSTOP Register

The MTB_TSTOP Register characteristics are:

Purpose	Configures MTB to stop tracing from DWT triggers.
Usage constraints	Privileged access permitted only. Unprivileged accesses generate a fault. Word accessible only. Halfword and byte accesses are UNPREDICTABLE.
Configurations	Available in all MTB configurations.
Attributes	See Table 3-1 on page 3-4.

Figure 3-7 shows the MTB_TSTOP Register bit assignments.



Figure 3-7 MTB_TSTOP Register bit assignments

Table 3-9 shows the MTB_TSTOP Register bit assignments.

Table 3-9 MTB_TSTOP Register bit assignments

Bits	Name	Reset	Description	
[31:4]	Reserved	0	These bits are reserved for future use and must be treated as UNK/SBZP.	
[3:0] ^a	СМРМАТСН	0	When a DWT raises a CMPMATCH event on a comparator, and the associated CMPMATCH field is: 1 Trigger a trace stop event.	
			0 The DWT event is ignored.	

a. The number of bits in CMPMATCH depend on the number of the DWT comparators.

Appendix A Signal descriptions

This appendix describes the MTB signals. It contains the following sections:

- *About the signal descriptions* on page A-2.
- *Clock, reset, and control signals* on page A-3.
- *AMBA AHB interface* on page A-4.
- *SRAM memory interface* on page A-6.
- *Execution trace interface* on page A-7.
- *External trace control interface* on page A-8.
- *Debug authentication interface* on page A-9.
- *Miscellaneous signals* on page A-10.

A.1 About the signal descriptions

The tables in this appendix list the MTB signals, along with their direction, input or output, and a description.

— Note —

All input and output signals are synchronous to HCLK.

A.2 Clock, reset, and control signals

Table A-1 shows the MTB clock, reset, and control signals.

Table A-1 Clock, reset, and control signals

Signal	Туре	Description
HCLK	Input	AHB-Lite interface and processor clock. All signals in or out of the MTB are processed on the positive, rising edge, of this clock.
RESETn	Input	Active-LOW reset signal. The reset can be asserted asynchronously, but must be deasserted synchronously to HCLK . RESETn must be asserted for at least two HCLK cycles. If trace is required during a local processor reset, then RESETn must be connected to the SoC power-on reset signal, otherwise connect RESETn to the HRESETn signal.
IDLE	Output	Indicates that there are no pending operations, therefore HCLK can be externally gated to save power. Note There are combinatorial paths from signals in the AHB-Lite and execution trace interfaces to the IDLE output signal.

A.3 AMBA AHB interface

Table A-2 shows the AMBA AHB interface signals. See *ARM*[®] *AMBA*[®] *5 AHB Protocol Specification, AHB5, AHB-Lite* for more information.

Table A-2 AMBA AHB interface signals

Signal	Direction	Description
HADDR[31:0]	Input	 32-bit address used to identify the memory or device being accessed by a transaction. The value on the bus is only valid for the following signal combinations: HTRANS[1], HREADY, and HSELRAM are HIGH. HTRANS[1], HREADY, and HSELSFR are HIGH.
HBURST[2:0]	Input	Indicates transfer type, single or burst. Note The MTB does not use HBURST[2:0].
HMASTLOCK	Input	Indicates the transfer is part of a locked sequence. Note The MTB does not use HMASTLOCK.
HPROT[3]	Input	Indicates a cacheable information transaction. HPROT[3] is HIGH for a cacheable transfer and LOW for a non-cacheable transfer. Note
HPROT[2]	Input	Indicates a bufferable information transaction. HPROT[2] is HIGH for a bufferable transfer and LOW for a non-bufferable transfer. —— Note The MTB does not use HPROT[2].
HPROT[1]	Input	 Indicates a Privilege information access. HPROT[1] is HIGH for a Privileged access and LOW for a User access. The value on the bus is only valid for the following signal combinations: HTRANS[1], HREADY, and HSELRAM are HIGH. HTRANS[1], HREADY, and HSELSFR are HIGH. Note
HPROT[0]	Input	Indicates a data versus opcode operation. HPROT[0] is HIGH for data access and LOW for an opcode fetch. Note

Table A-2 AMBA AHB interface signals (continued)

Signal	Direction	Description
HSIZE[2:0]	Input	 Indicates the size of a transfer. b000 indicates a byte transfer, b001 indicates a halfword transfer, and b010 indicates a word transfer. The value on the bus is only valid for the following signal combinations: HTRANS[1], HREADY, and HSELRAM are HIGH. HTRANS[1], HREADY, and HSELSFR are HIGH. Mote
HTRANS[1:0]	Input	Indicates the transfer type. All transfers are issued as non-sequential transfers so there is no guaranteed relationship to any previous transfer address. b00 indicates an IDLE cycle and b10 indicates a NONSEQUENTIAL transfer. Note
HWDATA[31:0]	Input	The write data associated with the address of an AHB-Lite write cycle. The destination of the write data is dependent on the values of HADDR[1:0] . The value on the bus is only valid during a write transfer. The value is only accepted by the MTB when HREADYOUT is HIGH.
HWRITE	Input	 Indicates the data transfer direction. HWRITE is HIGH for a write transfer, and LOW for a read transfer. The value on this signal is only valid for the following signal combinations: HTRANS[1], HREADY, and HSELRAM are HIGH. HTRANS[1], HREADY, and HSELSFR are HIGH.
HSELRAM	Input	SRAM address space select. Indicates the transfer destination is the MTB SRAM.
HSELSFR	Input	Special function registers address space select. Indicates the transfer is the MTB special function registers.
HREADY	Input	When HIGH, the HREADY input indicates to the MTB that the previous transfer is complete. Note The HREADY input value in the address-phase of a transfer to or from the MTB is equal to the MTB HREADYOUT output value, if it is also the data-phase of MTB transfer. In other words, the previous transfer was also to or from the MTB.
HRDATA[31:0]	Output	The read data associated with the address of an AHB-Lite read cycle. The destination of the read data is dependent on the values of HADDR[1:0] and HSIZE[1:0] . The value on the bus is only valid when HREADYOUT is HIGH and HRESP is LOW during a read transfer.
HREADYOUT	Output	When HIGH, the HREADYOUT signal indicates that a transfer has finished and the MTB can accept the next transfer. This signal can be driven LOW to extend the transfer.
HRESP	Output	The transfer response that provides the AHB-Lite master with additional information on the status of the transfer. When LOW, HRESP indicates the transfer status is OK. When HIGH, HRESP indicates the transfer status is ERRORNoteHRESP is continuously driven LOW because the MTB does not generate any errors.

A.4 SRAM memory interface

Table A-3 shows the SRAM memory interface signals.

Table A-3 SRAM memory interface signals

Signal	Direction	Description
RAMRD[31:0]	Input	Read data associated with the SRAM memory location addressed by RAMAD .
RAMAD[MTBAWIDTH-3:0]	Output	Address for the SRAM memory location accessed in a transaction.
RAMWD[31:0]	Output	Write data associated with the SRAM memory location addressed by RAMAD.
RAMCS	Output	SRAM chip select.
RAMWE[3:0]	Output	 SRAM byte write enables. When HIGH the requested byte of the word addressed by RAMAD is written. When LOW the requested byte of the word addressed by RAMAD is read. The write enable bit to data byte association is: RAMWE[0] accesses RAMWD[7:0]. RAMWE[1] accesses RAMWD[15:8]. RAMWE[2] accesses RAMWD[23:16]. RAMWE[3] accesses RAMWD[31:24].

A.5 Execution trace interface

Table A-4 shows the execution trace interface signals.

Table A-4 Execution trace interface signals

Signal	Direction	Description
IAESEQ	Input	Indicates the next instruction address in execute, IAEX, is sequential, that is non-branching.
IAEXEN	Input	IAEX register enable.
IAEX[30:0]	Input	Registered address of the instruction in the execution stage, shifted right by one bit.
ATOMIC	Input	Indicates the processor is performing non-instruction related activities.
EDBGRQ	Output	Request for the processor to enter the Debug state, if enabled, and halt.

A.6 External trace control interface

Table A-5 shows the external trace control signals.

Table A-5 External trace control interface signals

Signal	Direction	Description
TSTART	Input	Trace start. This is the external trace start input. See <i>TSTART and TSTOP signals</i> on page 2-8.
ТЅТОР	Input	Trace stop. This is the external trace stop input. See <i>TSTART and TSTOP signals</i> on page 2-8.

A.7 Debug authentication interface

Table A-6 shows the debug authentication interface signals.

	U	5
Signal	Direction	Description
DBGEN	Input	Invasive debug enable.
NIDEN	Input	Non-invasive debug enable.

Table A-6 Debug authentication interface signals

A.8 Miscellaneous signals

Table A-7 shows the miscellaneous signals

Table A-7 Miscellaneous signals

Signal	Direction	Description
ECOREVNUM[3:0]	Input	ECO revision number. This value can be read directly from the CoreSight Peripheral ID3 register. See Table 3-2 on page 3-6.
SRAMBASEADDR[31:0]	Input	MTB SRAM base address in the processor memory map.
		Note SRAMBASEADDR[31:0] must match the base address of the memory region indicated by HSELRAM.

Appendix B Example Programming Sequences

This appendix describes some example programming sequences for the MTB. It contains the following sections:

- *Discovery* on page B-2.
- Trace Enable Programming Sequence on page B-3.
- *Trace Disable Programming Sequence* on page B-4.

B.1 Discovery

The MTB occupies two separate regions of the processor memory map:SFRTrace control and CoreSight registers.SRAMContains the trace packets, and can be used as general-purpose memory.

The MTB is CoreSight compliant. You can discover its presence in a system in the following way:

- 1. The system ROM Table contains an entry that points to the base address of the SFR region.
- 2. The SFR region contains the CoreSight Peripheral ID registers that identify the MTB component. See *CoreSight registers* on page 3-5.
- 3. Read the base address of the SRAM region from the SFR base register.

If required, you can determine the MTB SRAM size in the following way:

- 1. Write all 1s to the MTB_POSITION.POINTER field.
- 2. Read the value of the MTB_POSITION register.
- 3. Determine the MTBAWIDTH value using the equation:

MTBAWIDTH = N+3

Where N is the number of bits set to 1 in the MTB_POSITION.POINTER field.

—— Note

Only the minimum number of LSBs, required to enable writing trace into all SRAM locations, are present in the POINTER field. The remaining MSBs of the field are RAZ/WI.

4. Determine the maximum SRAM size supported by the MTB implementation using the equation:

maximum SRAM size = $2^{\text{MTBAWIDTH}}$ bytes.

An implementation might have less SRAM than this and the unimplemented locations might be an alias of the implemented locations. In this case, you can determine the actual SRAM size in the following way:

- Write a different value to all locations in the MTB SRAM region. That is from location SRAMBASEADDR to location SRAMBASEADDR + (2^{MTBAWIDTH}) – 4, using word size accesses
- 2. Read back the memory locations and compare with the original values.

B.2 Trace Enable Programming Sequence

The following is an example programming sequence showing how to enable trace:

LDR	r2, =MTB_SFRBASE	; MTB SFR Base Address
MOVS	r1, #0	; POSITION POINTER = 0, WRAP = 0
STR	r1, [r2]	; Write POSITION register
MOVS	r0, #0	; FLOW WATERMARK =0, AUTOHALT = 0, AUTOSTOP = 0
STR	r0, [r2, #8]	; Write FLOW register
MOVS	r1, #12-4	; MASTER MASK = 12-4 (Trace buff size = 4KB)
MOVS	r0, #1	
LSLS	r0, #31	; MASTER.EN = 1
ORRS	r1, r0	
STR	r1, [r2, #4]	; Write MASTER register
	– Note –––––	

In this example it is assumed that the MTB SRAM is dedicated for trace and not shared with the application. If the SRAM needs to be shared with the application, the code must be changed.

B.3 Trace Disable Programming Sequence

The following is an example programming sequence showing how to disable trace:

LDR	r2,	=MTB_SFRBASE	;	MTB SFR Base Addr
LDR	r1,	[r2, #4]	;	Read MASTER register
LDR	r0,	=0x7fffffff		
ANDS	r0,	r1		
STR	r0,	[r2, #4]	;	MASTER.EN = \emptyset
LDR	r1,	[r2]	;	Read POSITION register
LSRS	r1,	#3	;	POSITION.POINTER value. Carry flag contains WRAP-bit value

Appendix C **Revisions**

This appendix describes the technical changes between released issues of this book.

Table C-1 Issue A

Change	Location	Affects	
First release	-	-	

Table C-2 Differences between issue A and issue B

Change	Location	Affects
Updated the debug authentication interface section	Debug authentication interface on page 2-5	r0p0
Updated the priority order in the trace start and stop section	Trace start and stop on page 2-7	r0p0
Updated the MTB_MASTER Register bit assignments	MTB_MASTER Register on page 3-9	r0p0

Table C-3 Differences between issue B and issue C

Change	Location	Affects
Changed the product name to Cortex-M23	-	r0p0
Updated the CoreSight registers table	CoreSight registers on page 3-6	r0p0
Added the base memory address	Trace control registers on page 3-4	r0p0

Revisions