CoreSight™ MTB-M0+

Revision: r0p1

Technical Reference Manual

ARM DDI 0486B (ID011213)

CoreSight MTB-M0+
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
19 January 2012	A	Confidential	First release for r0p0
14 December 2012	B	Non-Confidential	First release for r0p1

Proprietary Notice

Confidentiality Status

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreSight MTB-M0+ Technical Reference Manual

Preface

About this book vi
Feedback ix

Chapter 1 Introduction

1.1 About the CoreSight MTB-M0+ 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 Functional description

2.1 About the functions 2-2
2.2 Interfaces 2-3
2.3 Operation 2-6

Chapter 3 Programmers model

3.1 About the programmers model 3-2
3.2 Memory model 3-3
3.3 Register summary 3-4
3.4 Register descriptions 3-8

Appendix A Signal descriptions

A.1 About the signal descriptions A-2
A.2 Clock, reset, and control signals A-3

A.3 AMBA AHB-Lite 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 **Example 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 **Revisions**

Preface

This preface introduces the *CoreSight MTB-M0+Technical Reference Manual*. It contains the following sections:

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

About this book

This book is for the CoreSight Micro Trace Buffer for the Cortex™-M0+ processor, the CoreSight MTB-M0+ macrocell.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n*	Identifies the major revision of the product.
p*n*	Identifies the minor revision or modification status of the product.

Intended audience

This book is written for:

- System designers, system integrators, and verification engineers.
- Software developers who want to use the MTB.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the MTB and its features.

Chapter 2 *Functional description*

Read this chapter for a description of the functionality of the MTB.

Chapter 3 *Programmers model*

Read this chapter for a description of the MTB programmable registers and information for programming the MTB.

Appendix A *Signal descriptions*

Read this for a description of the signals in the MTB.

Appendix B *Example Programming Sequences*

Read this for a description of some example programming sequences for the MTB.

Appendix C *Revisions*

Read this for a description of the technical changes between released issues of this book.

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* on page vii.
- *Timing diagrams* on page vii.

- *Signals*.

Typographical conventions

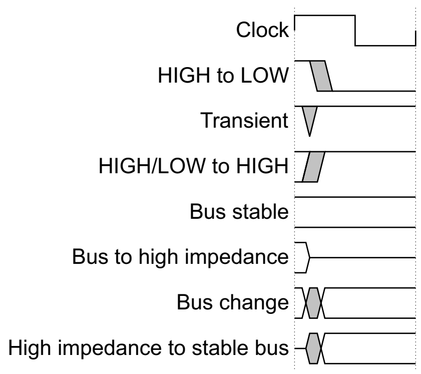
The following table describes the typographical conventions:

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.
`monospace`	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>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are UNDEFINED, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: • HIGH for active-HIGH signals. • LOW for active-LOW signals.
Lower-case n	At the start or end of a signal name denotes an active-LOW signal.

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:

- *CoreSight MTB-M0+ Implementation and Integration Manual* (ARM DIT 0031).
- *Cortex-M0+ Technical Reference Manual* (ARM DDI 0484).
- *AMBA® 3 AHB-Lite™ Protocol Specification* (ARM IHI 0033).
- *CoreSight Architecture Specification* (ARM IHI 0029).
- *ARM v6-M Architecture Reference Manual* (ARM DDI 0419).

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 0486B.
- 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 MTB-M0+ and its features. It contains the following sections:

- *About the CoreSight MTB-M0+* 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-M0+

The *CoreSight MTB-M0+* (MTB), provides a simple execution trace capability to the Cortex-M0+ processor. The MTB is not intended to be competitive with an ARM *Embedded Trace Macrocell* (ETM™) or *Program Trace Macrocell* (PTM™) trace solution.

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 3 AHB-Lite protocol. See the *AMBA 3 AHB-Lite Protocol*.

1.2.2 Debug authentication interface

This MTB complies with the CoreSight authentication interface. See the *CoreSight Architecture Specification*.

1.3 Features

The MTB features and benefits are:

- Provision of program flow tracing for the Cortex-M0+ 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-Lite 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.

1.5 Configurable options

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

Table 1-1 MTB configurable options

Feature	Configurable option
SRAM address width (AWIDTH)[a]	5-32 bits.
User/Privileged support	Present or absent.
Reset all registers	Present or absent.

a. Because the SRAM interface is 32-bits wide, the actual width of the SRAM address bus is AWIDTH-2. See Table A-3 on page A-6.

1.6 Test features

The MTB has no test features, because the necessary 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.

Integration and Implementation Manual

The *Implementation and Integration Manual* (IIM) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the *Register Transfer Level* (RTL) with the build configuration options.
- How to integrate the MTB into a SoC. This includes describing the pins that the integrator must tie off to configure the macrocell for the required integration.
- The processes to sign off the integration and implementation of the design.

The IIM is a confidential book that is only available to licensees.

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 synthesise 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.
r0p1	**ECOREVNUM** is in the Peripheral ID3 register instead of the Peripheral ID2 register.

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-M0+ based system.

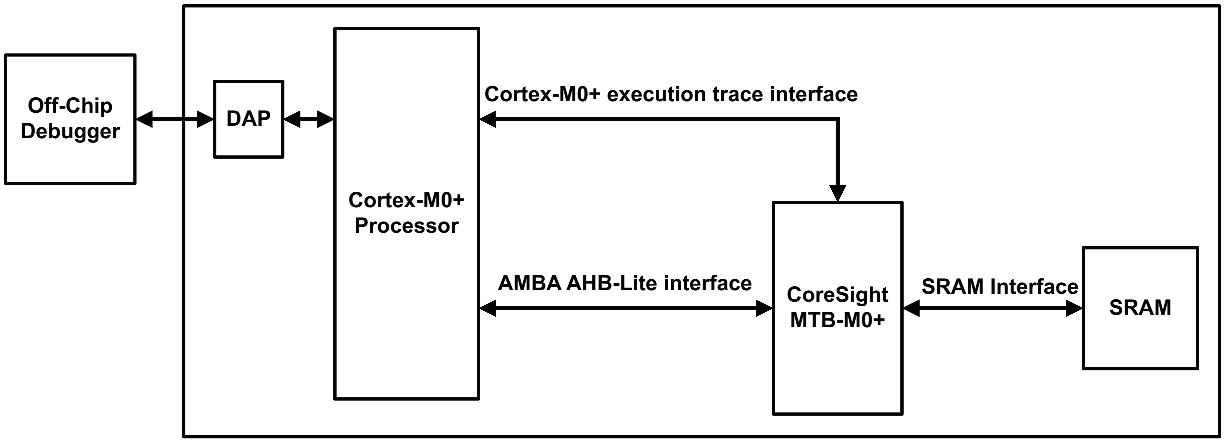


Figure 2-1 MTB system diagram

When enabled, the MTB records changes in program flow, reported by the Cortex-M0+ 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.

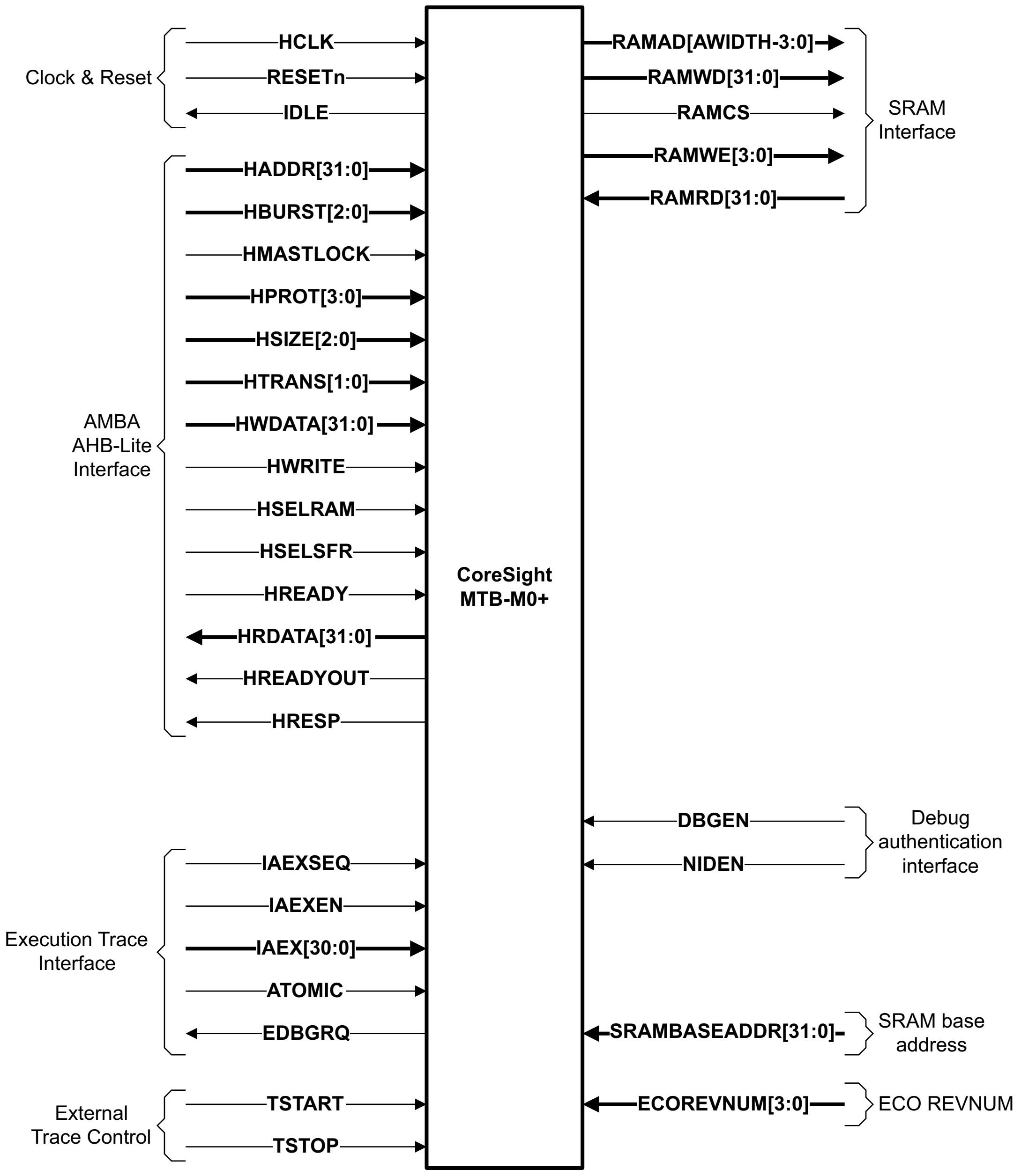


Figure 2-2 MTB interfaces

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 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, halfword or word accesses.

See the *AMBA 3 AHB-Lite Protocol Specification* 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.

Trace packet write access to the SRAM take 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.

Special Function Register, SFR, accesses always occur with zero wait states.

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 *FLOW Register* on page 3-11.

2.2.4 External trace enable interface

This interface enables external control over when tracing starts and stops. It consists of the **TSTART** and **TSTOP** signals. 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

DBGEN	NIDEN	Execution trace enabled	SRAM access allowed from AHB	SFR access allowed from AHB	EDBGRQ
0	0	No	Yes	Yes	0
0	1	Yes	Yes	Yes	0
1	X	Yes	Yes	Yes	MASTER.HALTREQ

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.

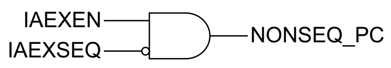


Figure 2-3 MTB non-sequential PC diagram

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

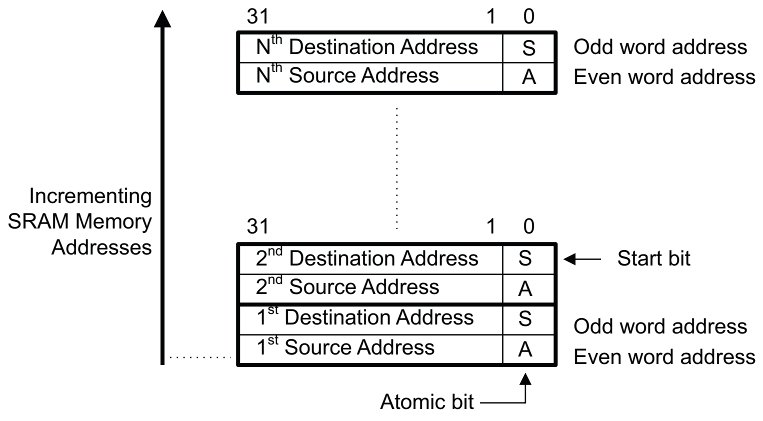


Figure 2-4 MTB execution trace storage format

The first, lower addressed, word contains the source of the branch, the address it branched from. The value stored only records bits[31:1] of the source address, because Thumb instructions are at least 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 destination of the branch, 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 `0` 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 the next higher word in memory, an odd word 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 `0`. 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 v6-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 MASTER.EN bit in the 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 *MASTER Register* on page 3-9.

You can control trace externally using the **TSTART** and **TSTOP** signals or using the 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 MASTER.EN bit, see *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.

If more than one source attempts to modify the 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 MASTER.EN bit.
3. **TSTART** and **TSTOP** signals.

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 POSITION.POINTER field to point to the beginning of the trace buffer. See *POSITION Register* on page 3-8.
  - By setting the FLOW.AUTOSTOP bit to `0`. See *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 MASTER.EN bit is not affected by these signals. If **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 MASTER.EN bit can stop the trace.

TSTART and TSTOP signals

When the **TSTART** signal is HIGH for one or more clock cycles and the MASTER.TSTARTEN bit is `1`, then the MASTER.EN bit is set to `1` and tracing starts from the memory location indicated by the POSITION.POINTER field in the Position Trace Control Register. See *MASTER Register* on page 3-9 and *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 MASTER.EN bit is `1`, then the 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 MASTER.EN bit is set to `1`.
- Tracing can be stopped automatically using the FLOW Trace Control Register. See *FLOW Register* on page 3-11.
- Tracing can also be stopped under software control by setting the MASTER.EN bit to `0` in the MASTER Trace Control Register, see *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 in128 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 Dhrystone2.1.

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 in the accompanying text:
  - 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 4K byte *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.

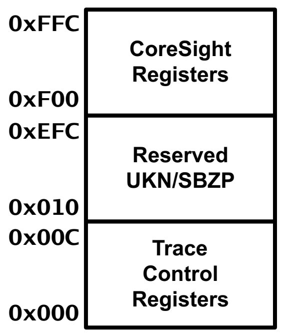


Figure 3-1 MTB special function register memory map

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, POSITION, MASTER, FLOW, and BASE.

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

Table 3-1 Register summary

Offset	Name	Type	Reset[a]	Width	Description
`0x000`	POSITION	RW	-	32	*POSITION Register* on page 3-8.
`0x004`	MASTER	RW	-	32	*MASTER Register* on page 3-9.
`0x008`	FLOW	RW	-	32	*FLOW Register* on page 3-11.
`0x00C`	BASE	RO	-	32	*BASE Register* on page 3-13.

a. See register descriptions for reset values.

Note

The behavior of the trace functionality is UNPREDICTABLE if the POSITION and FLOW registers are not programmed prior to enabling trace, that is when software writes a 1 to either the MASTER.EN or 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	0x0000000D	See the *CoreSight Architecture Specification* for more information about the registers and access types.
0XFF4	Component ID1	0x00000090	
0XFF8	Component ID2	0x00000005	
0XFFC	Component ID3	0x000000B1	
0xFE0	Peripheral ID0	0x00000032	
0xFE4	Peripheral ID1	0x000000B9	
0xFE8	Peripheral ID2	0x0000001B	
0xFEC	Peripheral ID3[31:8]	0x000000	
	Peripheral ID3[7:4]	**ECOREVNUM**	
	Peripheral ID3[3:0]	0x0	
0xFD0	Peripheral ID4	0x00000004	
0xFD4	Peripheral ID5	0x00000000	
0xFD8	Peripheral ID6	0x00000000	
0xFDC	Peripheral ID7	0x00000000	
0xFCC	Device Type Identifier	0x00000031	
	Reserved[31:8]	0x000000	
	Sub-Type[7:4]	0x3 (Basic Trace Router)	
	Class[3:0]	0x1 (Trace Sink)	
0xFC8	Device Configuration	0x00000000	
0xFBC	Device Architecture	0x47700A31	
	Architect [31:21]	0x23B (ARM)	
	Reserved [20]	0x1	
	Revision [19:16]	0x0 (Minor Architecture Revision 0)	
	Architecture ID [15:0]	0x0A31 (Basic Trace Router Architecture, Major Architecture Revision 0)	
0xFB8	Authentication Status [31:4]	0x0000000	
	Authentication Status [3]	1	
	Authentication Status [2]	**NIDEN** or **DBGEN**	
	Authentication Status [1]	1	
	Authentication Status [0]	**DBGEN**	
0xFB4	Lock Status	0x00000000	
0xFB0	Lock Access	0x00000000	

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 *CoreSight Architecture Specification* for more information.

3.4 Register descriptions

This section describes the MTB trace control registers. CoreSight registers are described in the *CoreSight Architecture Specification*.

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

3.4.1 POSITION Register

The 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 POSITION Register bit assignments.

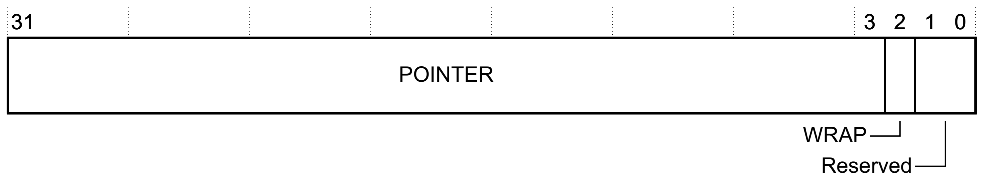


Figure 3-2 POSITION Register bit assignments

Table 3-3 shows the POSITION Register bit assignments.

Table 3-3 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 POSITION register using the following equation: system address = BASE + ((P + (2^{AWIDTH} - (BASE MOD 2^{AWIDTH}))) MOD 2^{AWIDTH}). Where P = POSITION AND 0xFFFF_FFF8. Where BASE is the BASE register value. See *BASE Register* on page 3-13. **Note** • The size of the SRAM is parameterized and the most significant bits of the POINTER field can be RAZ/WI, depending on the AWIDTH parameter value. See *Configurable options* on page 1-6. • POSITION register bits greater than or equal to AWIDTH are RAZ/WI, therefore, the active POINTER field bits are [AWIDTH-4:0]. • The POINTER field value is relative to the base address of the SRAM in the system memory map.
[2]	WRAP	UNKNOWN	This bit is set to 1 automatically when the POINTER value wraps as determined by the MASTER.MASK field in the 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 MASTER Register

The MASTER Register characteristics are:

Purpose Contains:

- The main trace enable bit.
- Other trace control fields.

Usage constraints

- Before the MASTER.EN or MASTER.TSTARTEN bits are set to 1, software must initialize the POSITION and FLOW registers.
- If the FLOW.WATERMARK field is used to stop tracing or to halt the processor, the MASTER.MASK field must still be set to a value that prevents the POSITION.POINTER field from wrapping before it reaches the FLOW.WATERMARK value.
- The **EDBGRQ** output value is also affected by the Debug authentication interface. See *Debug authentication interface* 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 MASTER Register bit assignments.

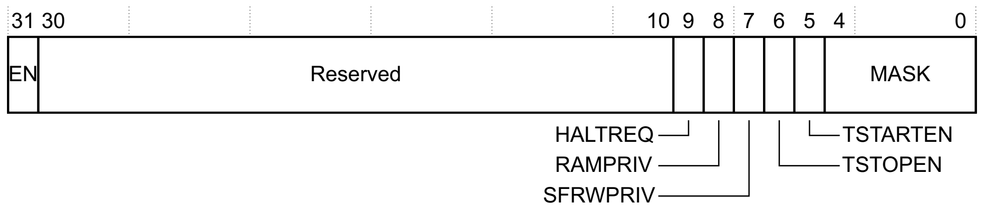


Figure 3-3 MASTER Register bit assignments

Table 3-4 shows the MASTER Register bit assignments.

Table 3-4 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 POSITION.POINTER. The POSITION.POINTER value auto increments after the trace data packet is written. The EN bit can be automatically set to `0` using the FLOW.WATERMARK field and the 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 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 FLOW.WATERMARK or 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 FLOW.WATERMARK field. See *FLOW Register* 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 *AMBA 3 AHB-Lite Protocol Specification* or *AMBA AHB-Lite interface signals* on page A-4 for more information. **Note** This bit is implemented only if User/Privilege support is present in the MTB configuration. If User/Privilege support is absent, then the behavior of this bit is RAZ/WI and the value of the **HPROT[1]** signal is ignored. See *Configurable options* on page 1-6.

Table 3-4 MASTER Register bit assignments (continued)

Bits	Name	Reset	Description
[7]	SFRWPRIV	1	Special Function Register Write Privilege bit. If this bit is `0`, then User or Privileged AHB-Lite read and write accesses to the Special Function Registers are permitted. If this bit is `1`, then only Privileged write accesses are permitted and User write accesses are ignored. The **HPROT[1]** signal determines if an access is User or Privileged. See *AMBA 3 AHB-Lite Protocol Specification* or *AMBA AHB-Lite interface signals* on page A-4 for more information. **Note** • SFR read accesses are not controlled by this bit and are always permitted. • This bit is implemented only if User/Privilege support is present in the MTB configuration. If User/Privilege support is absent, then the behavior of this bit is RAZ/WI and the value of the **HPROT[1]** signal is ignored. See *Configurable options* on page 1-6.
[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 POSITION.POINTER field that can be updated by automatic increment. If the trace tries to advance past this power of two, the POSITION.WRAP bit is set to `1`, the POSITION.POINTER[MASK:0] bits are set to zero, and the POSITION.POINTER[AWIDTH-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 AWIDTH-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 FLOW Register

The 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 shows the FLOW Register bit assignments.

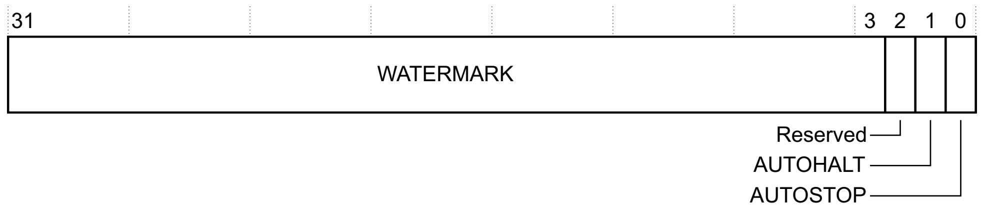


Figure 3-4 FLOW Register bit assignments

Table 3-6 shows the FLOW Register bit assignments.

Table 3-6 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 POSITION.POINTER field. When the 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 POSITION.POINTER, then the MASTER.HALTREQ bit is automatically set to 1. If the **DBGEN** signal is HIGH, the MTB asserts this halt request to the Cortex-M0+ processor by asserting the **EDBGRQ** signal.
[0]	AUTOSTOP	0	If this bit is 1 and WATERMARK is equal to POSITION.POINTER, then the 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 POSITION.POINTER field value to point to the beginning of the trace buffer.
  - Setting the 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-M0+ processor to enter the Debug state. To enter Debug state, the Cortex-M0+ processor might have to perform additional branch type operations. Therefore you must set the WATERMARK field below the final entry in the trace buffer region.

3.4.4 BASE Register

The 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.

Table 3-7 shows the BASE Register bit assignments.

Table 3-7 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 *Miscellaneous signals* on page A-10 for more information.

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-Lite 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	Type	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-Lite interface

Table A-2 shows the AMBA AHB-Lite interface signals. See *AMBA 3 AHB-Lite Protocol Specification* for more information.

Table A-2 AMBA AHB-Lite 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** **The MTB does not use HPROT[3]**.
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** • The MTB ignores **HPROT[1]** if User/Privilege support is absent in its configuration. • The MTB uses **HPROT[1]** only if User/Privilege support is present in its configuration. See *Configurable options* on page 1-6 and the SFRWPRIV and RAMPRIV bits in the *MASTER Register* on page 3-9.
HPROT[0]	Input	Indicates a data versus opcode operation. **HPROT[0]** is HIGH for data access and LOW for an opcode fetch. **Note** The MTB does not use **HPROT[0]**.

Table A-2 AMBA AHB-Lite 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. ——— **Note** ——— The MTB does not use **HSIZE[2]**. ———————
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** ——— The MTB does not use **HTRANS[0]**. ———————
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 ERROR. ——— **Note** ——— **HRESP** 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[AWIDTH-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 *TSTART and TSTOP signals* on page 2-8.
TSTOP	Input	Trace stop. This is the external trace stop input. See *TSTART and TSTOP signals* on page 2-8.

A.7 Debug authentication interface

Table A-6 shows the debug authentication interface signals.

Table A-6 Debug authentication interface signals

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

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:

SFR	Trace control and CoreSight registers.
SRAM	Contains 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 POSITION.POINTER field.
2. Read the value of the POSITION register.
3. Determine the AWIDTH value using the equation:

   AWIDTH = N+3

   Where N is the number of bits set to 1 in the 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^{AWIDTH} 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:

1. Write a different value to all locations in the MTB SRAM region. That is from location SRAMBASEADDR to location SRAMBASEADDR + (2^{AWIDTH}) – 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
```

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 = 0
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 debug authentication interface behavior table headings.	Table 2-1 on page 2-5	All revisions
Updated PeripheralID 2 and 3 definitions.	Table 3-2 on page 3-6	All revisions
Updated note on modifications you must make to PeripheralID, if you modify the MTB.	Table 3-2 on page 3-6	All revisions
Updated POSITION.POINTER address calculation.	Table 3-3 on page 3-9	All revisions
Changed how the ECO revision number is generated from **ECOREVNUM[3:0]**.	Table A-7 on page A-10	r0p1
Changed discovery algorithm description to add improvements.	*Discovery* on page B-2	All revisions