ARM® CoreSight™ MTB-M33

Revision: r0p2

Technical Reference Manual

ARM

ARM 100231_0002_00_en

ARM® CoreSight™ MTB-M33

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	28 September 2016	Confidential	First release for r0p0
0001-00	03 February 2017	Confidential	First release for r0p1
0002-00	10 May 2017	Non-Confidential	First release for r0p2

Non-Confidential Proprietary Notice

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

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.

Unrestricted Access is an ARM internal classification.

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-M33 Technical Reference Manual

Preface

This preface introduces the *ARM® CoreSight™ MTB-M33 Technical Reference Manual*.

It contains the following:

About this book

This book is for MTB-M33.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This book is written for system designers, system integrators, and verification engineers. It is also aimed at software developers who want to use the MTB.

Using this book

This book is organized into the following chapters:

Part A MTB-M33 Functional Description
This part describes the MTB-M33 functionality.

Chapter A1 Introduction
This chapter introduces MTB-M33 and its features.

Chapter A2 Functional description
This chapter describes the MTB-M33 functionality.

Chapter A3 Programmers model
This chapter describes the MTB-M33 registers and provides information for programming MTB-M33.

Part B MTB-M33 Register Descriptions
This part describes the MTB-M33 system registers.

Chapter B1 MTB-M33 registers
This chapter summarizes the MTB-M33 registers.

Part C Appendices

Appendix A Example Programming Sequences
This appendix describes some example programming sequences for the MTB-M33.

Appendix B Revisions
This appendix describes 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 the *ARM Glossary* for more information.

Typographic conventions

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 following figure 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.

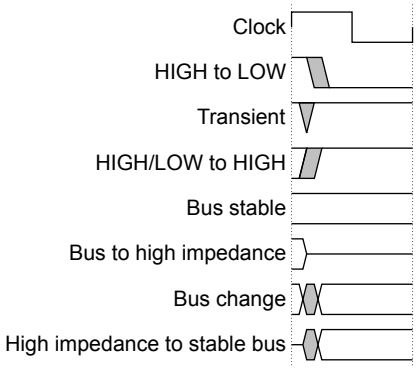

Figure 1 Key to timing diagram conventions

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.

Lowercase n
: At the start or end of a signal name denotes an active-LOW signal.

Additional reading

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

ARM publications

- *AMBA® APB Protocol Version 2.0 Specification* (ARM IHI 0024).
- *ARM® Cortex®-M33 Processor Technical Reference Manual* (ARM 100230).
- *ARM® AMBA® 5 AHB Protocol Specification* (ARM IHI 0033).
- *ARM® CoreSight™ Architecture Specification v2.0* (ARM IHI 0029).
- *ARM®v8-M Architecture Reference Manual* (ARM DDI 0553).

The following confidential books are only available to licensees:

- *ARM® Cortex®-M33 Processor Integration and Implementation Manual* (ARM 100323).

Other publications

- None.

Feedback

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 *ARM CoreSight MTB-M33 Technical Reference Manual*.
- The number ARM 100231_0002_00_en.
- If applicable, the page number(s) to which your comments refer.
- 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.

Part A
MTB-M33 Functional Description

Chapter A1
Introduction

This chapter introduces MTB-M33 and its features.

It contains the following sections:

A1.1 About the MTB-M33

The MTB-M33 provides a simple execution trace capability to the Cortex-M33 processor.

The *Micro Trace Buffer* (MTB) is not intended to be competitive with an ARM *Embedded Trace Macrocell* (ETM).

A1.2 Compliance

The MTB-M33 complies with, or implements, the specifications that are identified in this section.

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

Advanced Microcontroller Bus Architecture

The MTB-M33 complies with the following AMBA protocols:

- AMBA 5 AHB protocol. See *ARM® AMBA® 5 AHB Protocol Specification*.
- AMBA 4 APB slave interface. See *AMBA® APB Protocol Version 2.0 Specification*.

A1.3 Features

This section identifies the MTB-M33 features and benefits.

The MTB-M33 features and benefits are:

- Provision of program flow tracing for the Cortex-M33 processor.
- Small area.
- Power reduction features.
- The MTB-M33 connects to the SRAM that can be used for both trace and general-purpose storage by the processor.
- The SRAM size is configurable.
- The position and size of the trace buffer in the SRAM is configurable by software.
- External hardware can control trace start and stop.
- CoreSight compliant.
- ARM TrustZone® technology, using the ARMv8-M Security Extension supporting Secure and Non-secure states, subject to configuration.

A1.4 Interfaces

The interfaces on the MTB-M33 are:

AHB slave interface
The slave interface provides memory-mapped access to the SRAM.

Synchronous SRAM master interface
The SRAM interface connects to SRAM. SRAM can be used for storing trace packets and as a general-purpose memory within the Cortex-M33 memory map. When the SRAM is used as a general-purpose memory within the Cortex-M33 memory map, MTB-M33 operates as a simple AHB SRAM bridge.

Related references

A2.2.1 AHB slave interface on page A2-27.
A2.2.2 SRAM interface on page A2-27.

A1.5 Configurable options

The following table shows the MTB-M33 configurable options available.

Table A1-1 MTB-M33 configurable options

Feature	Options	Done at
ARM TrustZone technology	Security Extensions are not implemented.	Implementation
	Security Extensions are implemented.	
Cross Trigger Interface (CTI)	No Cross Trigger Interface.	Implementation
	Cross Trigger Interface included.	
SRAM address width (MTBAWIDTH)[a]	5-32 bits.	Integration

[a] Because the SRAM interface is 32 bits wide, the actual width of the SRAM address bus is MTBAWIDTH-2.

A1.6 Design process

The MTB-M33 is delivered as synthesizable RTL that must go through implementation, integration, and programming processes before you can use it in a product.

The following definitions describe each top-level process in the design flow:

Implementation
The implementer configures and synthesizes the RTL.

Integration
The integrator connects the implemented design into a SoC. This includes connecting it to a memory system and peripherals.

Programming
The system programmer develops the software required to configure and initialize the MTB-M33, and tests the required application software.

A different party can perform each stage in the process. Implementation and integration choices affect the behavior and features of the MTB-M33.

For MCUs, often a single design team integrates the MTB-M33 before synthesizing the complete design. Alternatively, the team can synthesize the MTB-M33 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-M33 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-M33 by programming particular values into registers. This affects the behavior of the MTB-M33.

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.

A1.7 Documentation

The MTB-M33 documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the MTB-M33 behavior. 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-M33, then contact:

- The implementer to determine:
  - The build configuration of the implementation.
  - What integration, if any, was performed before implementing the MTB-M33.
- The integrator to determine the pin configuration of the device that you are using.

Integration and Implementation Manual

The MTB-M33 is implemented and integrated as part of the processor implementation and integration, see *ARM® Cortex®-M33 Processor Integration and Implementation Manual* for more details.

The *Integration and Implementation Manual* (IIM) is a confidential book that is only available to licensees.

A1.8 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

r0p1 Various engineering errata fixes.

r0p2
Various engineering errata fixes.

Chapter A2
Functional description

This chapter describes the MTB-M33 functionality.

It contains the following sections:

A2.1 Functionality

The following figure shows the main interfaces on the MTB-M33 and how they are connected in a Cortex-M33-based system.

For simplicity, only connections related to the MTB-M33 are shown without trace control. See the *ARM® Cortex®-M33 Processor Technical Reference Manual* for more information about the Cortex-M33 processor.

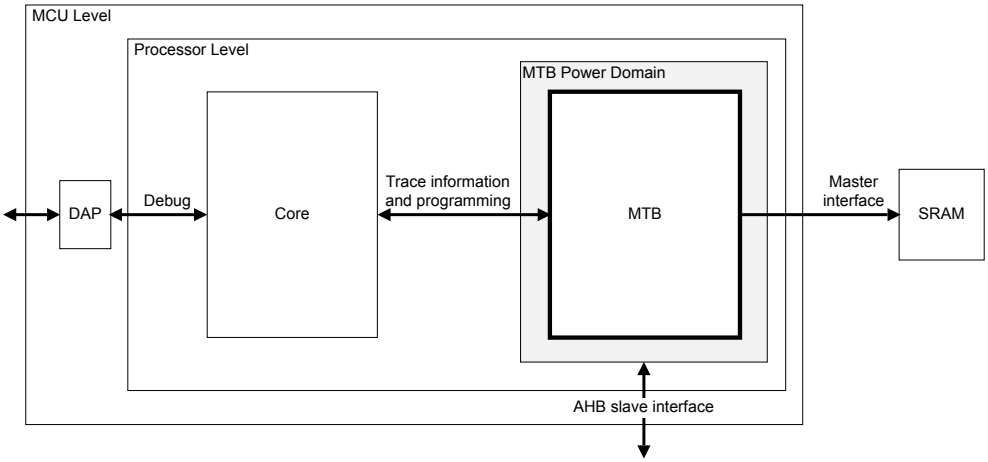

Figure A2-1 MTB-M33 system diagram

When enabled, the MTB-M33 records changes in program flow reported by the Cortex-M33 processor. This information is stored as trace packets in the SRAM.

An off-chip debugger can extract the trace information from the SRAM. The trace information is extracted over the AHB slave interface. The debugger is able to reconstruct the program flow from this information.

The MTB-M33 can function as a bridge to the SRAM if selected during integration. Software running on the processor can access the SRAM using the AHB slave interface of the MTB-M33 and the AHB interface of the core, if both are connected during the integration process. The MTB-M33 simultaneously stores trace information into the SRAM and gives the processor access to the SRAM. The MTB-M33 ensures that trace write accesses have priority over AHB accesses. In a low-cost or low-area implementation, or a low-cost and low-area implementation, the SRAM can be the only RAM included in the system with the memory shared between software and trace.

The MTB-M33 includes a register, MTB_BASE, that indicates the position of the SRAM in the processor memory map. An external input signal, **MTBSRAMBASE**, is connected to the register. When tied appropriately in the system, it allows software running on the processor to auto-discover the SRAM base value.

The MTB-M33 records all nonsequential instructions and does not record conditional instructions, unless they are taken branches.

The MTB-M33 does not include any form of load/store data trace capability, nor does it include tracing of any other information.

A2.2 Interfaces

The MTB-M33 has the following interfaces:

A2.2.1 AHB slave interface

This section describes the AHB slave interface as used by the MTB-M33.

The AHB slave interface provides access to the SRAM.

SRAM accesses can be either byte, halfword, or word accesses.

See the *ARM® AMBA® 5 AHB Protocol Specification* for more information.

Wait state behavior

SRAM accesses from the AHB slave interface occur with zero wait states when there is no trace data being written to the SRAM.

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

Buffering

If two addresses are equal, the AHB slave interface can buffer two address phases and one data write phase for SRAM accesses, enabling:

- Accesses to the SRAM to occur at the same time a trace packet is written to the SRAM.
- Read access to the SRAM to follow an AHB write access without inserting wait states.
- Write access to the SRAM without inserting wait states, if a trace packet is not being written to SRAM.

A2.2.2 SRAM interface

The MTB-M33 uses the SRAM interface for trace storage in the SRAM.

The MTB-M33 also operates as a simple AHB to SRAM bridge.

The MTB-M33 supports SRAM size from 32 bytes to 4GB.

If, during the integration phase, the AHB of the core is set to communicate with the MTB-M33 AHB:

- For Non-secure location in SRAM when ARM TrustZone technology is implemented, the core can have Read, Write, or both Read and Write accesses. This is subject to the MTB-M33 programming, instruction addresses, and the security state of the core.
- If ARM TrustZone technology is not implemented, the core can have Read and Write accesses to the entire SRAM address space.

For all the accesses to SRAM:

- For any location in SRAM when ARM TrustZone technology is implemented, the Non-secure accesses can have Read, Write, or both Read and Write permissions to Non-secure SRAM region. This is subject to the MTB-M33 programming and SRAM addresses. The Secure accesses have Read and Write permissions to the entire SRAM address space.
- If ARM TrustZone technology is not implemented, the accesses have Read and Write permissions to the entire SRAM address space.

The core can read the trace packet information stored in the SRAM if:

- During the integration phase, the AHB of the core is connected to the MTB-M33 AHB.
- The MTB-M33 programming gives Read permission. When ARM TrustZone technology is implemented, both MTB-M33 programming and security of the core must give Read permission.

A2.3 Operation

This section describes the MTB-M33 operation.

A2.3.1 MTB-M33 execution trace packet format

This section describes the MTB-M33 execution trace packet format.

The execution trace packet consists of a pair of 32-bit words that the MTB-M33 generates when it detects the processor PC value changes non-sequentially.

Note

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

The MTB-M33 detects non-sequential PC value change through information that is obtained from the core.

The following figure shows how the execution trace information is stored in memory as a sequence of packets.

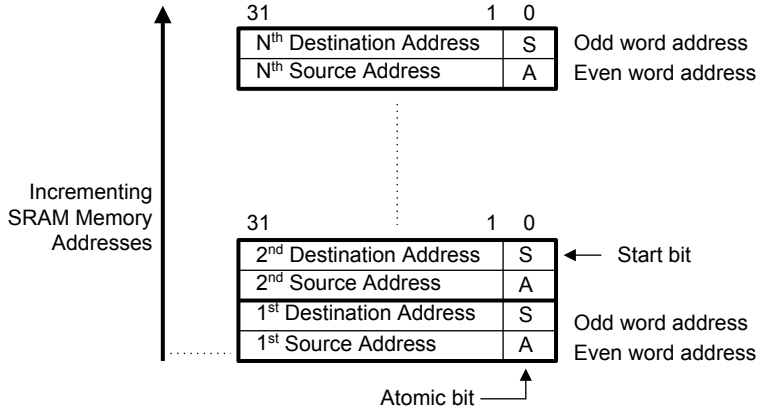

Figure A2-2 MTB-M33 execution trace storage format

The first, lower addressed, word contains the source of the branch, that is, the address it branched from. The stored value only records bits[31:1] of the source address, because Thumb® instructions are always at least halfword-aligned. The least significant bit of the value is the A-bit. The A-bit indicates the state of the processor at the time of the branch.

The A-bit can differentiate whether the branch originated:

- From an instruction in a program.
- As a result of an exception or Lockup.
- As a PC update in Debug state.

The branch originates from an instruction when the A-bit is 0. When the A-bit is 1, the branch:

- Originates from a PC update in Debug state.
- Originates from an exception or Lockup entry.
- Presents destination target for exception return.

This word is always stored at an even word location.

The second, higher addressed word contains the destination of the branch, that is, the address it branched to. This word is always stored at the next higher location in memory (an odd word address). The stored value only records bits[31:1] of the branch target 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 the first packet after the trace started and a value of 0 is used for other packets.

It is possible to start and stop tracing multiple times in a trace session. Therefore the memory might contain several packets with the S-bit set to 1.

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 SVC instruction causes an exception, 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, exception exit, 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*.
  - A-bit set to 0.
- The second, exception return, 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.

Authenticating trace information in the SRAM with no TrustZone technology implemented

Where TrustZone technology is not implemented, the MTB-M33 is authenticated to trace if both of the following are true:

- The external input *Non-Invasive Debug Enable* (**NIDEN**) signal, is set HIGH.
- The MTB-M33 is programmed to trace (MTB_MASTER.EN is HIGH) and is able to store trace information in the SRAM. The other registers are programmed correctly, see *Part B MTB-M33 Register Descriptions* on page B-37.

If the MTB-M33 is not authenticated to trace, no trace information is produced and sent to the SRAM. If the MTB-M33 loses authentication while the two packets are produced for exception return, the following might happen:

- Both the exception exit and the exception return packets are not produced if authentication is lost during the execution of a `BX` or `POP` instruction.
- The first exception exit packet is produced and the second exception return packet is not produced if authentication is granted during the execution of a `BX` or `POP` instruction, but lost before the core leaves corresponding Handler address space.

the MTB-M33 can stop writing trace to SRAM when:

- Authentication is lost when the external input signal, **NIDEN**, goes LOW, no tracing is allowed in the system.
- The MTB-M33 is programmed to stop tracing, either by:
  - Deasserting the MTB_MASTER.EN bit.
  - Setting other registers to automatically stop tracing. For example, asserting the MTB_FLOW.AUTOSTOP bit.
- The MTB-M33 is programmed to stop writing trace to SRAM.

Integrity of a single packet is always maintained, regardless of authentication or reprogramming sequence. That is, when the source word is generated, the destination word follows in the next SRAM cycle.

Authenticating trace information in the SRAM with TrustZone technology implemented

When TrustZone technology is implemented, in addition to authenticating trace information in the SRAM, consider the following:

- Additional authentication extends to Secure and Non-secure:
  - Code executed by the core.
  - SRAM regions.
  - AHB accesses.

  When the Cortex-M33 processor executes Secure or Non-secure code and the MTB-M33 is programmed to trace, it generates:
- From Secure code, Secure trace that can be written to Secure and Non-secure SRAM regions.
- From Non-secure code, Non-secure trace that can be written only to Non-secure SRAM regions.

The MTB-M33 recognizes a given SRAM address as Secure or Non-secure according to how it was programmed. See *B1.8 MTB_SECURE register* on page B1-52

Note

Only Secure core code can write MTB_SECURE register, whereas Non-secure code is only granted read access.

The MTB-M33 recognizes Secure and Non-secure AHB accesses:

- Secure AHB accesses are allowed to read and write both Secure and Non-secure SRAM regions.
- Non-secure AHB accesses are only allowed to read and write Non-secure SRAM regions.

The MTB-M33 can generate source or target address as a combination of all bits equal to 1, bits[31:1] are set to `0xFFFFFFFF` including the A-bit and S-bit fields. This is a masked Secure address value that might occur in the packet if the MTB-M33 is not authenticated to trace Secure regions of the code (**SPNIDEN** LOW) and authenticated to trace (**NIDEN** HIGH and MTB programming allowing to trace).

Such an address value informs the debugger that a non-sequential PC change occurred on the transition between Non-secure and Secure code regions, although the exact Secure address cannot be presented in the packet. This situation might happen in two cases, when the MTB-M33 is not authenticated to trace Secure code:

- Secure to Non-secure region non-sequential PC change. In this case, the source address and A-bit field are masked to `0xFFFFFFFF` and the destination address and S-bit field remain unmasked.
- Non-secure to Secure region non-sequential PC change. In this case, the source address and A-bit field remain unmasked and the destination address and S-bit is masked to `0xFFFFFFFF`.

See the *ARM® Cortex®-M33 Processor Technical Reference Manual* for more authentication information.

A2.3.2 Trace start and trace stop

This section describes the trace start and stop functions.

Tracing is enabled when the MTB_MASTER.EN bit in the Master Trace Control Register is asserted. If Security Extensions are implemented, the MTB_MASTER.NSEN bit and MTB_SECURE register define if a given trace can be stored in SRAM, see *B1.3 MTB_MASTER register* on page B1-44 and *B1.8 MTB_SECURE register* on page B1-52. There are various ways to set the MTB_MASTER.EN bit to 1 to start tracing, or to 0 to stop tracing, see *B1.3 MTB_MASTER register* on page B1-44.

Trace can be started or stopped by:

- Programming the trace MTB trace control registers.
- Internal processor signals generated by the *Cross Trigger Interface* (CTI) or the *Data Watchpoint and Trace* (DWT) logic, depending on your implementation. See the *ARM® Cortex®-M33 Processor Technical Reference Manual* for more information.

You can program the MTB-M33 to stop tracing automatically when the memory reaches a specified watermark level. If you do not use the watermark mechanism, and the trace buffer overflows, then the buffer can wrap and overwrite previous trace packets, subject to programming.

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

1. A watermark stop.
2. A software write to the MTB_MASTER.EN bit.
3. Internal processor signals generated by the *Cross Trigger Interface* (CTI) or the *Data Watchpoint and Trace* (DWT) logic, depending on your implementation. See the *ARM® Cortex®-M33 Processor Technical Reference Manual* for more information.

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. For more information, see *B1.2 MTB_POSITION register* on page B1-42.
  - By setting the MTB_FLOW.AUTOSTOP bit to 0. For more information, see *B1.4 MTB_FLOW register* on page B1-47.
- If tracing is stopped while a trace packet is being written into memory, the MTB-M33 ensures that the packet write is completed.

Chapter A3
Programmers model

This chapter describes the MTB-M33 registers and provides information for programming MTB-M33.

It contains the following sections:

A3.1 About the programmers model

The following information applies to the MTB-M33 registers:

- The behavior of the MTB-M33 is UNPREDICTABLE if it is not programmed correctly.
- Use only word size, 32-bit, transactions to access all registers.
- Access type in *Table B1-1 Register summary* on page B1-40 is described as follows:

  RW Read and write.

  RO Read only.

A3.2 Memory model

This section describes the MTB-M33 memory model.

The MTB-M33 has a 4KB *Special Function Register* (SFR) memory region. This SFR contains the trace control registers and CoreSight registers.

The base address of the MTB-M33 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.

The following figure shows the SFR memory map with the address offsets for each set of registers.

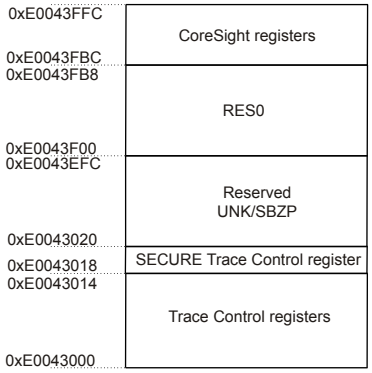

Figure A3-1 MTB-M33 special function register memory map

The MTB_SECURE Trace Control register, accessible at address offset `0xE0043018`, is implemented only if TrustZone is implemented, see *A1.5 Configurable options* on page A1-20. Otherwise this memory region is reserved and UNK/SBZP.

Part B
MTB-M33 Register Descriptions

Chapter B1
MTB-M33 registers

This chapter summarizes the MTB-M33 registers.

It contains the following sections:

B1.1 Register summary

The register tables include information about:

- The register addresses.
- Additional information about the implementation of the register, where appropriate.

Note

- Registers not listed here are not implemented. Reading a non-implemented register address returns 0. Writing to a non-implemented register address has no effect.
- In the following table:
  - The Reset value column shows the value of the register immediately after an MTB-M33 reset. For read-only registers, every read of the register returns this value.
  - Access type is described as follows:

    RW Read and write.

    RO Read only.

All the MTB-M33 registers are 32 bits wide.

B1.1.1 Trace control registers

MTB-M33 has programmable registers to control the behavior of the trace features.

Trace control registers are only accessible using word (32-bit) transactions.

The following table shows the trace control registers and their memory addresses.

Table B1-1 Register summary

Address	Name	Type	Reset value	Width	Description
0xE0043000	MTB_POSITION	RW/RO[b]	-	32	*B1.2 MTB_POSITION register* on page B1-42
0xE0043004	MTB_MASTER	RW/RO[b]	-	32	*B1.3 MTB_MASTER register* on page B1-44
0xE0043008	MTB_FLOW	RW/RO[b]	-	32	*B1.4 MTB_FLOW register* on page B1-47
0xE004300C	MTB_BASE	RO	-	32	*B1.5 MTB_BASE register* on page B1-49
0xE0043010	MTB_TSTART	RW/RO[b]	-	32	*B1.6 MTB_TSTART register* on page B1-50
0xE0043014	MTB_TSTOP	RW/RO[b]	-	32	*B1.7 MTB_TSTOP register* on page B1-51
0xE0043018	MTB_SECURE	RW/RO[c]	-	32	*B1.8 MTB_SECURE register* on page B1-52

Note

The behavior of the trace functionality is UNPREDICTABLE if the MTB_POSITION and MTB_FLOW registers are not programmed before enabling trace, that is, when software writes a 1 to either the MTB_MASTER.EN or MTB_MASTER.MTB_TSTART bit. See *A2.3.2 Trace start and trace stop* on page A2-30.

B1.1.2 CoreSight registers

The following table shows the CoreSight registers and their values.

[b] These registers are RW from Secure memory accesses or if the ARMv8-M Security Extensions are not included in the processor. They are RO from Non-secure memory accesses unless MTB_MASTER.NSEN is HIGH.

[c] MTB_SECURE register is RW in Secure mode and RO in Non-secure mode. The register is UNK/SBZP when the ARMv8-M Security Extensions are not included in the processor.

See the *ARM® CoreSight™ Architecture Specification v2.0* for more information about the registers and access types, and the peripheral ID fields.

Table B1-2 CoreSight registers

Offset	Name	Type	Reset value	Width	Description
0xFF0	CID0	RO	0x0000000D	32	CoreSight Component ID0
0xFF4	CID1	RO	0x00000090	32	CoreSight Component ID1
0xFF8	CID2	RO	0x00000005	32	CoreSight Component ID2
0xFFC	CID3	RO	0x000000B1	32	CoreSight Component ID3
0xFE0	PID0	RO	0x00000021	32	CoreSight Peripheral ID0
0xFE4	PID1	RO	0x000000BD	32	CoreSight Peripheral ID1
0xFE8	PID2	RO	0x0000000B	32	CoreSight Peripheral ID2
0xFEC	PID3[31:8]	RO	0x000000	32	CoreSight Peripheral ID3
	PID3[7:4]	RO	-[d]		
	PID3[3:0]	RO	0x0		
0xFD0	PID4	RO	0x00000004	32	CoreSight Peripheral ID4
0xFD4	PID5	RO	0x00000000	32	CoreSight Peripheral ID4
0xFD8	PID6	RO	0x00000000	32	CoreSight Peripheral ID4
0xFDC	PID7	RO	0x00000000	32	CoreSight Peripheral ID4
0xFCC	DEVTYPE[31:8]	RO	0x000000 (Reserved)	32	CoreSight Device Type Identifier[e]
	DEVTYPE[7:4]	RO	0x3 (Sub-type) Basic Trace Router		
	DEVTYPE[3:0]	RO	0x1 (Class) Trace Sink		
0xFC8	DEVID[31:6]	RO	0x0000000	32	CoreSight Device Identifier
	DEVID[5]	RO	0x1 TrustZone is implemented. 0x0 TrustZone is not implemented.		
	DEVID[4:0]	RO	MTBAWIDTH-1[f]		
0xFBC	DEVARCH[31:21]	RO	0x476 (Architect) ARM	32	CoreSight Device Architecture[g]
	DEVARCH[20]	RO	0x1 (Reserved)		
	DEVARCH[19:16]	RO	0x1 (Revision) Minor Architecture Revision 1		
	DEVARCHVER[15:12]	RO	0x0 Architecture version		
	DEVARCHPART[11:0]	RO	0xA31 Architecture part number.		

[d] Dependent on the exact revision of the silicon as specified in the CoreSight.
[e] Combined reset value for the CoreSight Device Architecture register is 0x00000031.
[f] To calculate the size of the SRAM use:

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

The minimum allowed value stored in DEVID[4:0] is 4, which represents 32 bytes, see *ARM®v8-M Architecture Reference Manual*.
[g] Combined reset value for the CoreSight Device Architecture register is 0x47700A31.

B1.2 MTB_POSITION register

The MTB_POSITION register contains the trace write pointer and the wrap bit.

Usage constraints There are no additional usage constraints.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40.

- All fields can be modified by software running in Secure state, or from a debugger with Secure access connected to the processor.
- All fields are read-only from Non-secure unless MTB_MASTER.NSEN is set to 1.
- The MTB-M33 updates the fields as trace is written to the SRAM interface.

The following figure shows the MTB_POSITION register bit assignments.

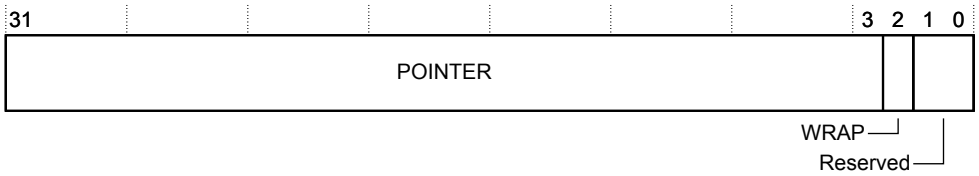

Figure B1-1 MTB_POSITION Register bit assignments

The following table shows the MTB_POSITION register bit assignments.

Table B1-3 MTB_POSITION register bit assignments

Bits	Name	Description
[31:3]	POINTER	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 is written. The field points to an unused location and is automatically incremented. A debugger can calculate the system address of the SRAM location that is on the AHB-Lite bus pointed to by the MTB_POSITION register using the following equation, where P = MTB_POSITION AND `0xFFFF_FFF8` and BASE is the MTB_BASE register value: system address = BASE + ((P + ($2^{MTBAWIDTH}$ - (BASE MOD $2^{MTBAWIDTH}$))) MOD $2^{MTBAWIDTH}$). **Note** • The size of the SRAM is parameterized and the most significant bits of the POINTER field can be RAZ/WI, depending on the MTBAWIDTH parameter value. See *A1.5 Configurable options* on page A1-20. • MTB_POSITION register bits greater than or equal to MTBAWIDTH are RAZ/WI, therefore, the active POINTER field bits are [MTBAWIDTH-4:0]. • The POINTER field value is relative to the base address of the SRAM in the system memory map. • When the MTB_POSITION register is written, bit[0] of the trace address is always cleared because trace packets are always aligned to a 64-bit address in the SRAM. A debugger might use the WRAP bit to determine whether the trace information above and below the pointer address is valid. The reset value is UNKNOWN.
[2]	WRAP	Indicates when the POINTER value has wrapped. The POINTER value is set by the MTB_MASTER.MASK bit field. **1** POINTER value has wrapped. **0** POINTER value has not wrapped. The reset value is UNKNOWN.
[1:0]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.

B1.3 MTB_MASTER register

The MTB_MASTER register contains the main trace enable bit and other trace control fields.

Usage constraints

- If the ARMv8-M Security Extension is implemented, Secure code must first program the MTB_SECURE register and MTB_MASTER.NSEN bit.
- The behavior of HALTREQ field depends on whether invasive debug is enabled on the processor debug authentication interface, as described in the *ARM® Cortex®-M33 Processor Technical Reference Manual*.
- 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 be set to a value that prevents the MTB_POSITION.POINTER field from wrapping before it reaches the MTB_FLOW.WATERMARK value.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40.

- All fields are modified by software running in Secure state, or from a debugger with Secure access connected to the processor.
- All fields are read-only from Non-secure unless MTB_MASTER.NSEN is set to 1.
- Automatic hardware mechanisms update the EN and HALTREQ bits.

The following figure shows the MTB_MASTER register bit assignments.

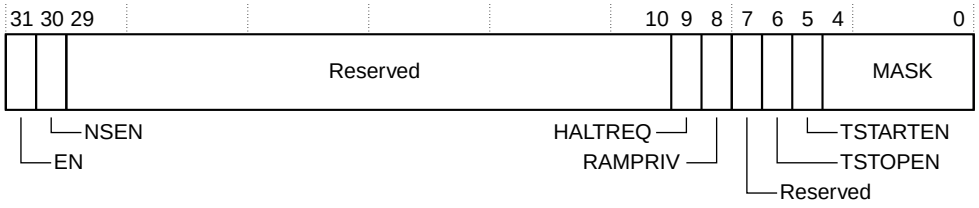

Figure B1-2 MTB_MASTER register bit assignments

The following table shows the MTB_MASTER register bit assignments.

Table B1-4 MTB_MASTER register bit assignments

Bits	Name	Description
[31]	EN	Main trace enable bit. If 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 is automatically set to 1, if the following conditions occur: • TSTARTEN bit is 1 and a trace start event is raised by CTI. • DWT CMPMATCH is 1 and the corresponding field in MTB_TSTART register is 1. **0** Trace data is not written into the SRAM memory location addressed by MTB_POSITION.POINTER.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 0, if the following conditions occur: • TSTOPEN bit is 1 and a trace stop event is raised by CTI. • DWT CMPMATCH feature combined with MTB_TSTOP register value. **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 a trace start event is received. In this case, tracing can only restart if the MTB_FLOW.WATERMARK or MTB_POSITION.POINTER value is changed by software. The reset value is 0.
[30]	NSEN	Non-secure trace enable bit. If this bit is: **1** Non-secure trace enabled. Any trace packets generated by the processor are treated as Non-secure and are not written to SRAM memory that is defined as Secure according to the MTB_SECURE register. When the MTB is programmed for Non-secure trace the MTB_MASTER (except the NSEN bit), MTB_POSITION, MTB_FLOW, MTB_TSTART, and MTB_TSTOP registers can be written by Non-secure memory accesses. **0** Secure trace enabled. All the MTB-M33 registers are read-only from a Non-secure memory access. This bit can only be written by a Secure memory access. If the ARMv8-M Security Extensions are not included in the processor this field is UNKNOWN and must be treated as UNK/SBZP. The reset value is 0.
[29:10]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.
[9]	HALTREQ	Halt request bit. If this bit is: **1** Request to the processor to enter debug halt state. The request is valid when: • The current debug authentication level of processor allows. • EN bit is not required. Request to the processor to enter debug halt state can be made using the MTB_FLOW.WATERMARK field, see *B1.4 MTB_FLOW register* on page B1-47. **0** No request to the processor to enter debug halt state. When HALTREQ is set to HIGH, the debugger needs to clear it to reset halt request. Additionally, the debugger needs to prevent HALTREQ to be set HIGH again by changing the MTB_POSITION.POINTER field value to point to the address value lower than MTB_FLOW.WATERMARK. Otherwise, the cleared HALTREQ is set again in the next clock cycle. To perform restart, DHCSR.C_HALT field in the NVIC must be cleared. The reset value is 0.

Table B1-4 MTB_MASTER register bit assignments (continued)

Bits	Name	Description
[8]	RAMPRIV	SRAM Privilege bit. If this bit is: **1** Only privileged AHB read and write accesses to the SRAM are permitted. Any unprivileged accesses result in no update to the SRAM and an error response on the AHB interface. **0** Unprivileged or privileged AHB read and write accesses to the SRAM are permitted. The reset value is 0.
[7]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.
[6]	TSTOPEN	Trace stop events enable. This field controls whether the trace is stopped by an external event from the CTI or DWT CMPMATCH feature combined with the MTB_TSTOP register value. If this bit is: **1** Tracing stop events occurs. When a stop event occurs, the EN bit is set to 0. If a trace packet is being written to memory, the write is completed before tracing is stopped. **0** Tracing stop event disabled. The reset value is 0.
[5]	TSTARTEN	Trace start event enable. This field controls whether the trace is started by an external event from the CTI or DWT CMPMATCH feature combined with the MTB_TSTART register value. If this bit is: **1** Tracing start until a stop condition occurs. When a start event occurs, the EN bit is set to 1. **0** Tracing start events disabled. The reset value is 0.
[4:0]	MASK	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 attempts to write across the boundary specified by the MASK field, that is 2^{MASK}, the following occurs: • MTB_POSITION.WRAP bit is set to 1. • MTB_POSITION.POINTER[MASK:0] bits are set to zero. • 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. The reset value is UNKNOWN.

The following table shows example MASK and POINTER values to illustrate the effect of the MASK field on the POINTER. If there was no MASK, POINTER+1 is the incremented value of POINTER. POINTER Next is the POINTER with MASK applied.

Table B1-5 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

B1.4 MTB_FLOW register

The MTB_FLOW register contains the WATERMARK address and the AUTOSTOP and AUTOHALT control bits.

Usage constraints There are no additional usage constraints.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40.

All fields can be modified by software running in Secure state, or from a debugger with Secure access connected to the processor.

All fields are read-only from Non-secure unless MTB_MASTER.NSEN is set to 1.

The following figure shows the MTB_FLOW register bit assignments.

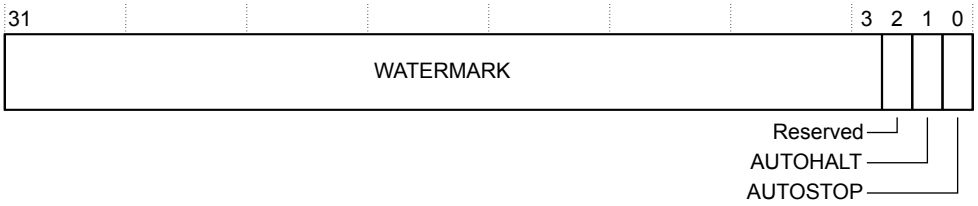

Figure B1-3 MTB_FLOW Register bit assignments

The following table shows the MTB_FLOW register bit assignments.

Table B1-6 MTB_FLOW register bit assignments

Bits	Name	Description
[31:3]	WATERMARK	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. The relation between the number of bits implemented in the WATERMARK field and the configuration Verilog parameter MTBAWIDTH is based on the following conditions: • If MTBAWIDTH is equal to 32, all 29 bits are implemented. • If MTBAWIDTH is less than 32, MSB bits of the WATERMARK field are not implemented and are RAZ/WI. The value at reset is UNKNOWN.
[2]	Reserved	This bit is reserved for future use and must be treated as UNK/SBZP.

Table B1-6 MTB_FLOW register bit assignments (continued)

Bits	Name	Description
[1]	AUTOHALT	Automatic trace halt. If this bit is: **1** MTB requests the processor to halt. When the WATERMARK value is equal to the MTB_POSITION.POINTER value and MTB_MASTER.EN bits are set to 1, then the MTB_MASTER.HALTREQ bit is automatically set to 1, and if permitted, the processor enters debug halt state. **0** MTB does not request processor to halt. The reset value is 0.
[0]	AUTOSTOP	Watermark auto stop. If this bit is: **1** When the WATERMARK value is equal to the MTB_POSITION.POINTER value, then the MTB.MASTER.EN bit is automatically set to 0. This stops tracing. **0** Tracing is unaffected. The reset value is 0.

Note

- If you stop tracing using the watermark auto stop feature, you cannot restart tracing until software clears the watermark auto stop. Clearing the watermark auto stop is achieved in one of the following ways:
  - Changing the MTB_POSITION.POINTER field value to point to the beginning of the trace buffer in SRAM.
  - Setting the MTB_FLOW.AUTOSTOP bit to 0.
- A debugger can use the AUTOSTOP bit to fill the trace buffer once only without halting the processor.
- A debugger can use the AUTOHALT bit to fill the trace buffer once before causing the processor to enter the Debug state. To enter Debug state, the processor might have to perform more branch type operations. Therefore, set the WATERMARK field below the final entry in the trace buffer region.
- If the core enters Debug state using the watermark autohalt feature, you cannot restart tracing and the core cannot leave the Debug state until the software clears the watermark autohalt. You can achieve this by doing both of the following actions:
  - Changing the MTB_POSITION.POINTER field value to point to the address value lower than MTB_FLOW.WATERMARK.
  - Setting the MTB_MASTER.HALTREQ bit to 0.

B1.5 MTB_BASE register

The MTB_BASE register indicates where the SRAM is located in the processor memory map.

This register enables auto discovery of the MTB SRAM location by either software running on the processor or a debugger connected to the processor.

Usage constraints There are no additional usage constraints.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40.
The register is read-only.

Note

- Because the SRAM must be aligned to a 32-byte address, MTB_BASE[4:0] is not used.
- The processor input signal MTBSRAMBASE[31:5] determines the value of MTB_BASE.BASE. The system sets this value to the base address of the SRAM relative to the processor memory map.

The following figure shows the MTB_BASE register bit assignments.

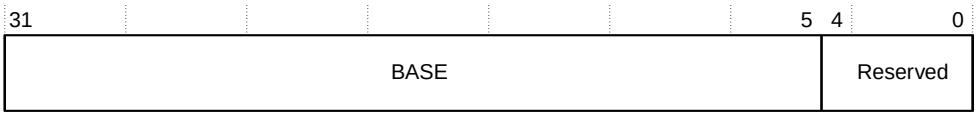

Figure B1-4 MTB_BASE register bit assignments

The following table shows the MTB_BASE register bit assignments.

Table B1-7 MTB_BASE register bit assignments

Bits	Name	Description
[31:5]	BASE	The value of the **MTBSRAMBASE[31:5]** signal. This is the value at reset.
[4:0]	Reserved	This bit is reserved for future use and must be treated as UNK/SBZP.

B1.6 MTB_TSTART register

The MTB_TSTART register controls the trace start events using the DWT CMPMATCH feature.

For more information about the DWT CMPMATCH feature, see *ARM®v8-M Architecture Reference Manual*.

Usage constraints The processor debug configuration, which is based on the number of available DWT comparators, determines the number of CMPMATCH fields in these registers. If a reduced set of debug resources are included, then two watchpoint comparators are included. If a full set of debug resources are included, then four watchpoint comparators are included.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40. All fields can be modified by software running in Secure state, or from a debugger with Secure access connected to the processor.

All fields are read-only from Non-secure unless MTB_MASTER.NSEN is set to 1.

The following figure shows the MTB_TSTART register bit assignments.

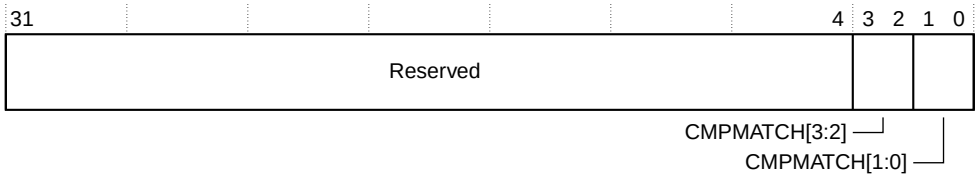

Figure B1-5 MTB_TSTART register bit assignments

The following table shows the MTB_TSTART register bit assignments.

Table B1-8 MTB_TSTART register bit assignments

Bits	Name	Description
[31:4]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.
[3:2]	CMPMATCH[3:2]	When a DWT raises a CMPMATCH event on a comparator, and the associated CMPMATCH field is: **1** Trigger a trace start event. **0** The DWT event is ignored. **Note** If the number of debug resources included is: **Reduced set** The CMPMATCH[3:2] reset value is UNKNOWN. **Full set** The CMPMATCH[3:2] value at reset is `0x0`.
[1:0]	CMPMATCH[1:0]	When a DWT raises a CMPMATCH event on a comparator, and the associated CMPMATCH field is: **1** Trigger trace start event. **0** The DWT event is ignored. This is the value at reset.

B1.7 MTB_TSTOP register

The MTB_TSTOP register controls the trace stop events using the DWT CMPMATCH feature.

For more information, see *ARM®v8-M Architecture Reference Manual*.

Usage constraints The processor debug configuration, which is based on the number of available DWT comparators, determines the number of CMPMATCH fields in these registers. If a reduced set of debug resources are included, then two watchpoint comparators are included. If a full set of debug resources are included, then four watchpoint comparators are included.

Configurations Available in all the MTB-M33 configurations.

Attributes See *Table B1-1 Register summary* on page B1-40. All fields can be modified by software running in Secure state, or from a debugger with Secure access connected to the processor. All fields are read-only from Non-secure unless MTB_MASTER.NSEN is set to 1.

The following figure shows the MTB_TSTOP Register bit assignments.

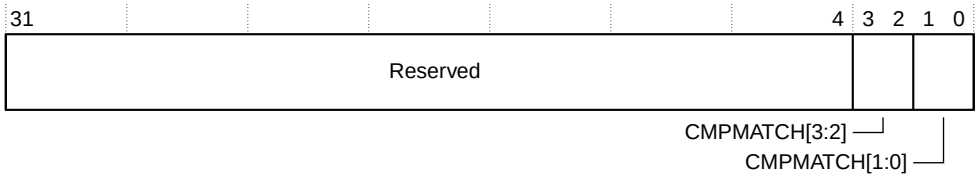

Figure B1-6 MTB_TSTOP register bit assignments

The following table shows the MTB_TSTOP register bit assignments.

Table B1-9 MTB_TSTOP register bit assignments

Bits	Name	Description
[31:4]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.
[3:2]	CMPMATCH[3:2]	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. **Note** If the number of debug resources included is: **Reduced set** The CMPMATCH[3:2] reset value is UNKNOWN. **Full set** The CMPMATCH[3:2] value at reset is `0x0`.
[1:0]	CMPMATCH[1:0]	When a DWT raises a CMPMATCH event on a comparator, and the associated CMPMATCH field is: **1** Trigger trace stop event. **0** The DWT event is ignored. This is the value at reset.

B1.8 MTB_SECURE register

The MTB_SECURE register allows the SRAM region to be partitioned into two regions, with one region being defined as Secure and the other as Non-secure.

This feature can be used to protect the SRAM so that the Secure region cannot be accessed from Non-secure code, or overwritten by Non-secure trace packets or accessed by Non-secure AHB transactions.

If a given Non-secure transaction attempts to access the Secure SRAM region, the transaction is not propagated to the SRAM Interface. In addition, if the access attempt was initialized from the AHB Interface an AHB error response is produced.

Usage constraints There are no additional usage constraints.

Configurations Available only if the ARMv8-M Security Extension is implemented. If the extensions have not been included the register is UNK/SBZP.

Attributes See *Table B1-1 Register summary* on page B1-40.

- Secure software can modify all the defined fields.
- A debugger with Secure access connected to the processor can modify all the defined fields.
- All fields are read-only by the software running in Non-secure state or by debugger with non-secure access connected to the processor.

The following figure shows the MTB_SECURE register bit assignments.

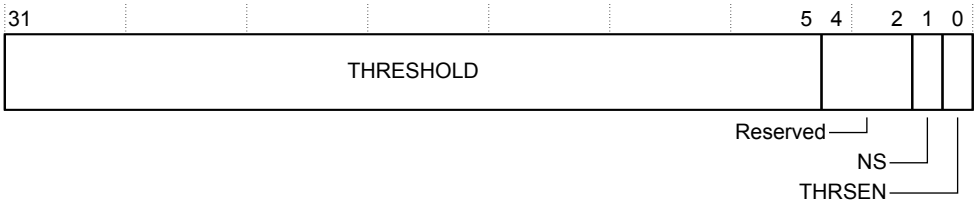

Figure B1-7 MTB_SECURE Register bit assignments

The following table shows the MTB_SECURE register bit assignments.

Table B1-10 MTB_SECURE register bit assignments

Bits	Name	Description
[31:5]	THRESHOLD	Threshold field. Lowest address in the SRAM controlled by the MTB_SECURE register. This field is ignored when THRSEN is LOW. **Note** • Setting THRESHOLD to an address outside the available SRAM is UNPREDICTABLE. • The SRAM size can be determined using the DEVID[4:0] register. The relation between the number of bits implemented in the THRESHOLD field and the SRAM size: • For the maximum SRAM size 4GB, all bits are implemented. • If SRAM size is less than 4GB and more than 32 bytes, the MSB bits of the THRESHOLD field are not implemented, and are RAZ/WI. • Where the SRAM size is equal to 32 bytes, none of the bits are implemented, and entire SRAM can be either Secure or Non-secure, subject to the setting of NS and THRSEN bits. The reset value is UNKNOWN.
[4:2]	Reserved	These bits are reserved for future use and must be treated as UNK/SBZP.

Table B1-10 MTB_SECURE register bit assignments (continued)

Bits	Name	Description
[1]	NS	SRAM Non-secure bit. Functionality of this bit is correlated with the THRSEN bit. When THRSEN is set to 0 and: **NS is 0** Memory locations controlled by the MTB_SECURE Register are treated as Secure. **NS is 1** Memory locations controlled by the MTB_SECURE register are treated as Non-secure. When THRSEN is set to 1, and: **NS is 0** SRAM is divided into Non-secure region (lower addresses) and Secure region (higher addresses). THRESHOLD determines the first Secure address. **NS is 1** SRAM is divided into Secure region (lower addresses) and Non-secure region (higher addresses). THRESHOLD determines the first Non-secure address. The reset value is 0.
[0]	THRSEN	Threshold enable bit. • When the MTB RAM interface address width is greater than five, functionality of this bit is correlated with the NS bit. • When the MTB RAM interface address width is five, this bit is RAZ/WI, having no effect on SRAM security state. When this bit is: **0** The security level of all the SRAM memory is determined by the NS bit and THRESHOLD is ignored. This is the value at reset. **1** The security levels of the memory locations equal to and greater than THRESHOLD are determined by the NS bit and all memory locations less than THRESHOLD are determined by the inverse of the NS bit. The reset value is 0. Except for when the MTB RAM interface address width is five, then the reset value is UNKNOWN

The following table summarizes the relationships between THRSEN, NS and THRESHOLD bit fields in the MTB_SECURE register when the MTB RAM interface address width is greater than 2^5 bytes.

Table B1-11 MTB RAM interface address width is greater than 2^5 bytes

THRSEN	NS	Security state of SRAM
0	0	All secure
0	1	All non-secure
1	0	address >= THRESHOLD : Secure address < THRESHOLD : Non-secure
1	1	address >= THRESHOLD : Non-secure address < THRESHOLD : Secure

The following table summarizes the relationships between THRSEN, NS and THRESHOLD bit fields in the MTB_SECURE register when the MTB RAM interface address is equal to 2^5 bytes. In this case THRSEN bit value is ignored, and is RAZ/WI.

Table B1-12 MTB RAM interface address size is equal to 2^5 bytes

THRSEN	NS	Security state of SRAM
RAZ/WI	0	All Secure
RAZ/WI	1	All Non-secure

Part C
Appendices

Appendix A
Example Programming Sequences

This appendix describes some example programming sequences for the MTB-M33.

It contains the following sections:

A.1 MTB-M33 Discovery

This section introduces example programming sequences for the MTB-M33 discovery in the System.

MTB-M33 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.

MTB-M33 is CoreSight compliant. You can discover its presence in a system in the following way:

1. The processor 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-M33 component. For more information, see *B1.1.2 CoreSight registers* on page B1-40 and *ARM® Cortex®-M33 Processor Technical Reference Manual*.
3. Read the base address of the SRAM region from the SFR MTB_BASE register.

If necessary, you can determine the 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 MTBAWIDTH value reading DEVID register: MTBAWIDTH = DEVID[4:0] + 1.
4. Determine the maximum size of SRAM supported by the MTB-M33 implementation using the equation:Maximum SRAM size = $2^{MTBAWIDTH}$ bytes.

An implementation might have less SRAM than this maximum value, 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 using word size accesses.
2. Read back the memory locations and compare with the original values.

A.2 Trace Enable Programming Sequence

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

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

A.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 MTB_MASTER register
LDR     r0, =0x7fffffff
ANDS    r0, r1
STR     r0, [r2, #4]        ; MTB_MASTER.EN = 0
LDR     r1, [r2]            ; Read MTB_POSITION register
LSRS    r1, #3              ; MTB_POSITION.POINTER value.
                            ; Carry flag contains WRAP-bit value
```

Appendix B
Revisions

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

It contains the following section:

- *B.1 Revisions* on page Appx-B-62.

B.1 Revisions

The tables that follow show the technical changes between released issues of this book.

Table B-1 Issue 0000-00

Change	Location	Affected
First release	-	-

Table B-2 Differences between issue 0000-00 and issue 0001-00

Change	Location	Affected
Updated the product revision to r0p1, as identified in the CPUID Register in the processor	See the CPUID.REVISION bit assignment description in the *ARM® Cortex®-M33 Processor Technical Reference Manual*.	r0p1
Clarified the MTB-M33 execution trace packet format description	*A2.3.1 MTB-M33 execution trace packet format* on page A2-28	All

Table B-3 Differences between issue 0001-00 and issue 0002-00

Change	Location	Affected
Updated the product revision to r0p2, as identified in the CPUID Register in the processor	See the CPUID.REVISION bit assignment description in the *ARM® Cortex®-M33 Processor Technical Reference Manual*.	r0p1
Clarified the use of ARM TrustZone technology	*A1.3 Features* on page A1-18	All
Changed descriptions to use memory address, not the offset address of the register	*B1.1 Register summary* on page B1-40 *B1.1.1 Trace control registers* on page B1-40	All
Moved 'A debugger might use the WRAP bit to determine whether the trace information above and below the pointer address is valid.' into [2] WRAP table description	*B1.2 MTB_POSITION register* on page B1-42	All