CoreSight™ ETM™-M4

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0440C (ID070610)

CoreSight ETM-M4
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
22 December 2009	A	Non-Confidential, Restricted Access	First release for r0p0
02 March 2010	B	Non-Confidential	Second release for r0p0
29 June 2010	C	Non-Confidential	Updated 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 ETM-M4 Technical Reference Manual

List of Tables
CoreSight ETM-M4 Technical Reference Manual

List of Figures
CoreSight ETM-M4 Technical Reference Manual

Preface

This preface introduces the *CoreSight ETM-M4 Technical Reference Manual*. It contains the following sections:

- *About this book* on page viii
- *Feedback* on page x.

About this book

This book is for the CoreSight Embedded Trace Macrocell™ for the Cortex-M4 and Cortex-M4F processors, the CoreSight ETM-M4 macrocell.

You implement the ETM-M4 macrocell with either the Cortex-M4 processor or the Cortex-M4F processor. In this manual, in general:

- any reference to the processor applies to either the Cortex-M4 processor or the Cortex-M4F processor, as appropriate
- any reference to the Cortex-M4 processor applies also to the Cortex-M4F processor, as appropriate.

The context makes it clear if information applies to only one of the processor options.

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:

- Designers of development tools providing support for ETM functionality. Implementation-specific behavior is described in this document. You can find complementary information in the *Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014).
- Hardware and software engineers integrating the macrocell into an ASIC that includes a Cortex™-M4 processor. You can find complementary information in the *Cortex-M4 Integration Manual* (ARM DII 0239).

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the functionality of the macrocell.

Chapter 2 *Functional Description*

Read this for a description of the interfaces, operation, clocking and resets of the macrocell.

Chapter 3 *Programmers Model*

Read this for a description of the programmers model for the macrocell.

Appendix A *Signal Descriptions*

Read this for a summary of ETM signals.

Appendix B *Revisions*

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

Conventions

Conventions that this book can use are described in:

- *Typographical*

Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal 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 >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Further reading

This section lists publications by ARM and by third parties.

See `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:

- *Cortex-M4 Technical Reference Manual* (ARM DDI 0439)
- *Cortex-M4 Integration and Implementation Manual* (ARM DII 0239)
- *Embedded Trace Macrocell™ Architecture Specification* (ARM IHI 0014)
- *CoreSight Components Technical Reference Manual* (ARM DDI 0314)
- *CoreSight Architecture Specification* (ARM IHI 0020)
- *CoreSight Technology System Design Guide* (ARM DGI 0012)
- *AMBA™ 3 APB Protocol Specification* (ARM IHI 0024)
- *AMBA 3 ATB Protocol Specification* (ARM IHI 0032).

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 if appropriate.

Feedback on this book

If you have any comments on this book, send an e-mail to `errata@arm.com`. Give:

- the title
- the number
- the relevant page number(s) to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the ETM-M4 macrocell. It contains the following sections:

1.1 About the ETM-M4

The Cortex-M4 *Embedded Trace Macrocell* (ETM-M4) is an optional debug component that enables a debugger to reconstruct program execution. The CoreSight ETM-M4 supports only instruction trace. You can use it either with the Cortex-M4 *Trace Port Interface Unit* (M4-TPIU), or as part of a CoreSight system.

Figure 1-1 shows a typical Cortex-M4 system in a System-on-Chip (SoC) that includes a CoreSight ETM-M4 macrocell.

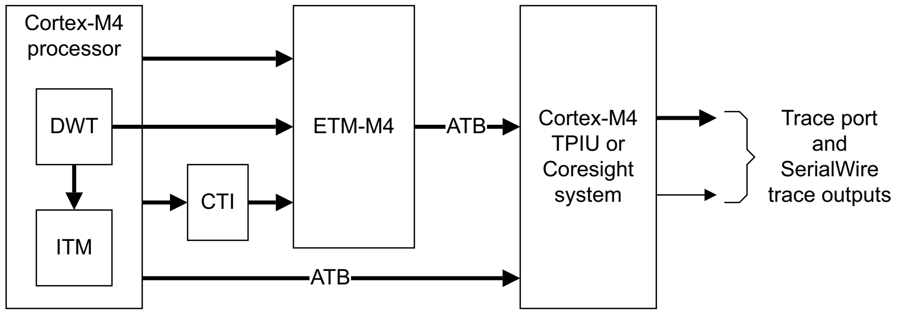
Cortex-M4 processor
DWT
ITM
CTI
ETM-M4
ATB
Cortex-M4 TPIU or Coresight system
ATB
Trace port and SerialWire trace outputs

Figure 1-1 Cortex-M4 with ETM-M4 block diagram

1.2 Compliance

ETM-M4 is compatible with the CoreSight architecture.

ETM-M4 implements version 3.5 of the ETM architecture, ETMv3.5. See the *Embedded Trace Macrocell Architecture Specification* for more information.

For more information about architectural compliance, see *Architecture and protocol information* on page 1-9.

1.3 Features

ETM-M4 provides:

- tracing of 16-bit and 32-bit Thumb instructions
- four EmbeddedICE watchpoint inputs
- a Trace Start/Stop block with EmbeddedICE inputs
- one reduced function counter
- two external inputs
- a 24-byte FIFO queue
- global timestamping.

See the *Embedded Trace Macrocell Architecture Specification* for information about:

- the trace protocol
- controlling tracing using triggering and filtering resources.

See the *Cortex-M4 Integration and Implementation Manual* for information about the macrocell signals.

1.4 Interfaces

The system connections to the ETM-M4 are supported using a *Cross Trigger Interface* (CTI):

- 0-2 external inputs
- trigger output.

See *Configurable options* on page 1-6 and *Interfaces* on page 2-7 for more information about the external inputs and external outputs.

1.5 Configurable options

The ETM-M4 macrocell includes the following configuration inputs:

- the maximum number of external inputs, see *External inputs* on page 2-5
- whether the system supports the FIFOFULL mechanism for stalling the processor, see Table 2-1 on page 2-3.

1.6 Test features

The ETM-M4 does not include any specific Design For Test (DFT) features.

1.7 Product documentation, design flow, and architecture

This section describes the ETM-M4 books, how they relate to the design flow, and the relevant architectural standards and protocols.

See *Further reading* on page ix for more information about the books described in this section.

1.7.1 Documentation

The ETM-M4 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 ETM-M4. It is required at all stages of the design flow. Some behavior described in the TRM might not be relevant because of the way that the ETM-M4 is implemented and integrated.

Integration and Implementation Manual

For both the processor and the ETM, the *Cortex-M4 Integration and Implementation 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 processor into a SoC. This includes a description of the integration kit and describes the pins that the integrator must tie off to configure the macrocell for the required integration.
- How to implement the processor and ETM in your design. This includes floorplanning guidelines, Design for Test (DFT) information, and how to perform netlist dynamic verification on the processor.
- The processes to sign off the integration and implementation of the design.

The ARM product deliverables include reference scripts and information about using the scripts to implement your design.

Reference methodology documentation from your EDA tools vendor complements the IIM.

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

1.7.2 Design flow

The ETM-M4 is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following process:

1. Implementation. The implementer synthesizes the RTL, usually in combination with the processor, then places and routes the netlist to produce a hard macrocell.
2. Integration. The integrator instantiates the macrocell of the combined processor and ETM into a SoC. This includes testing its integration with the other SoC components to which it is connected.
3. Programming. The debug software developer programs the ETM and tests any trace software required for use with a SoC.

For more information see the *Cortex-M4 Integration and Implementation Manual*.

1.7.3 Architecture and protocol information

The ETM-M4 complies with, or implements, the specifications described in:

- *Trace macrocell*
- *Advanced Microcontroller Bus Architecture*.

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

Trace macrocell

The ETM-M4 implements the ETM architecture version 3.5. See *Embedded Trace Macrocell Architecture Specification*.

Advanced Microcontroller Bus Architecture

This ETM-M4 complies with the *Advanced Microcontroller Bus Architecture* (AMBA) 3 *Advanced Peripheral Bus* (APB) and *Advanced Trace Bus* (ATB) protocols. See *AMBA 3 APB Protocol Specification* and *AMBA 3 ATB Protocol Specification*.

1.8 Product revisions

This section describes the differences in functionality between product revisions:

r0p0	First release.
r0p1	There are no differences in fuctionality for this release.

Chapter 2
Functional Description

This chapter describes the interfaces, operation, clocking and resets of the macrocell. It contains the following sections:

2.1 About the functions

Figure 2-1 shows a block diagram of the ETM, and shows how the ETM interfaces to the *Trace Port Interface Unit* (TPIU).

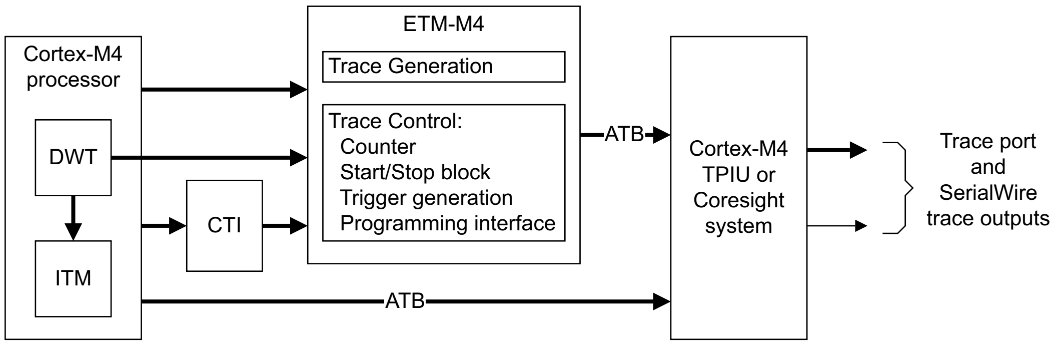

Figure 2-1 ETM block diagram

The Cortex-M4 system can perform low-bandwidth data tracing using the *Data Watchpoint and Trace* (DWT) and *Instruction Trace Macrocell* (ITM) components.

The ETM trace output is compatible with the *AMBA Trace Bus* (ATB) protocol, irrespective of the configuration of the trace port size and trace port mode within the ETM programmers model. The TPIU exports trace information from the processor. An implementation can replace the TPIU with other CoreSight trace components.

For more information see:

- *Cortex-M4 Technical Reference Manual*
- *Embedded Trace Macrocell Architecture Specification*.

The ETM provides a trace ID register for systems that use multiple trace sources. You must configure this register even if only a single trace source is in use.

The following sections provide information on features of the ETM:

- *Resources*
- *Timestamp format* on page 2-5
- *Periodic synchronization* on page 2-5
- *Data and instruction address compare resources* on page 2-5
- *External inputs* on page 2-5
- *Start/stop block* on page 2-5
- *Triggering* on page 2-6.

2.1.1 Resources

Because the ETM does not generate data trace information, the lower bandwidth reduces the requirement for complex triggering capabilities. This means that the ETM only includes a small sub-set of the possible resources allowed by the ETM architecture.

Table 2-1 lists the Cortex-M4 resources.

Table 2-1 Cortex-M4 resources

Feature	Present on ETM-M4
Architecture version	ETMv3.5
Address comparator pairs	0
Data comparators	0
Context ID comparators	0
Memory Map Decoders (MMDs)	0
Counters	1, reduced function counter only
Sequencer	No
Start/stop block	Yes
Embedded ICE comparators	4
External inputs	2
External outputs	0
Extended external inputs	0
Extended external input selectors	0
FIFOFULL	Yes
FIFOFULL level setting	Yes
Branch broadcasting	Yes
ASIC Control Register	No
Data suppression	No
Software access to registers	Yes
Readable registers	Yes
FIFO size	24 bytes
Minimum port size	8 bits
Maximum port size	8 bits
Normal port mode	-
Normal half-rate clocking, 1:1	Yes - asynchronous
Demux port mode	-
Demux half-rate clocking, 1:2	No
Mux port mode, 2:1	No
1:4 port mode	No
Dynamic port mode, including stalling	No. Supported by asynchronous port mode.
Coprocessor Register Transfer (CPRT) data	No

Table 2-1 Cortex-M4 resources (continued)

Feature	Present on ETM-M4
Load PC first	No
Fetch comparisons	No
Load data traced	No

Resource identification encoding

You configure the trace enable event, timestamp event, and trigger event using the same mechanism. For each event, a 17-bit register is used to define the event. This register provides:

- Resource A, bits [6:0]
- Resource B, bits [13:7]
- a Boolean function, bits [16:14].

Table 2-2 shows the encodings used for the Boolean function.

Table 2-2 Boolean function encoding for events

Encoding	Function
`0b000`	A
`0b001`	NOT(A)
`0b010`	A AND B
`0b011`	NOT(A) AND B
`0b100`	NOT(A) AND NOT (B)
`0b101`	A OR B
`0b110`	NOT (A) OR B
`0b111`	NOT (A) OR NOT (B)

Table 2-3 shows the encodings used for Resource identification.

Table 2-3 Resource identification encoding

Resource type[a]	Index range[b]	Description of resource type
`0b010`	0-3	DWT Comparator inputs (0-3)
`0b100`	0	Counter 1 at zero
`0b101`	15	Trace Start/Stop resource
`0b110`	0-1	ExtIn (0-1)
`0b110`	15	HardWired (always True)

a. For Resource A, bits [6:4]. For Resource B, bits [13:11]
b. For Resource A, bits [3:0]. For Resource B, bits [10:7].

2.1.2 Timestamp format

Timestamps are encoded as 48-bit natural binary numbers. A system implementation may provide a timestamp count which can be used by several trace sources as an aid to correlating the trace streams.

2.1.3 Periodic synchronization

The ETM uses a fixed synchronization packet generation frequency of every 1024 bytes of trace.

2.1.4 Data and instruction address compare resources

The DWT provides four address comparators on the data bus that provide debug functionality. Within the DWT unit, you can specify the functions triggered by a match, and one of these functions is to generate an ETM match input. These inputs are presented to the ETM as Embedded *In Circuit Emulator* (ICE) comparator inputs.

A single DWT resource can trigger an ETM event and also generate instrumentation trace directly from the same event.

You can configure the four DWT comparators individually to compare with the address of the current executing instruction to permit the ETM access to an instruction address compare resource. These inputs are presented to the ETM as Embedded ICE comparator inputs. The DWT provides either 1 or 4 comparators, depending on the implementation of the processor.

Note

Using a DWT comparator as an instruction address comparator reduces the number of available data address comparisons.

See the *Cortex-M4 Technical Reference Manual* for more information about the DWT unit.

2.1.5 External inputs

Two external inputs, **ETMEXTIN[1:0]**, enable additional components to generate trigger and enable signals for the ETM.

2.1.6 Start/stop block

The start/stop block provides a single-bit resource that can be used as an input to other parts of the resource logic, including the trace enable logic. The start/stop block can only be controlled by using the EmbeddedICE inputs to the ETM. The DWT controls these inputs.

The start/stop block is set to the start state if any of the EmbeddedICE watchpoint inputs selected as start resources in ETMTESSEICR go HIGH. The start/stop block is set to the stop state if any of the EmbeddedICE watchpoint inputs selected as stop resources in ETMTESSEICR go LOW.

If bit [25] of ETMTECR1 is 1, tracing will only be enabled when the start/stop block is in the start state.

Tracing is also only enabled when the result of evaluating the Trace Enable Event is TRUE. This event can be set to always be TRUE by programming a value of 0x6F to ETMTEEVR. For more information see the *Embedded Trace Macrocell Architecture Specification*.

2.1.7 Triggering

The ETM provides a trigger resource that can be used to identify a point within a trace run. The generation of a trigger does not affect the tracing in any way, but the trigger will be output in the trace stream, and can also be passed to other trace components or used to halt the processor. An external trace port analyzer can use the trigger to determine when to start and stop capture of trace.

2.2 Interfaces

The ETM-M4 has the following external interfaces:

ATB A 32-bit *Advanced Trace Bus* provides trace output from the macrocell. See the *AMBA 3 ATB Protocol Specification* for more information about this interface.

APB An *Advanced Peripheral Bus* provides the control interface for the macrocell. See the *AMBA 3 APB Protocol Specification* for more information about this interface.

CTI Your implementation can provide a *Cross Trigger Interface* to manage the interconnection of trigger and control signals between the processor core, ETM, and TPIU. The implementation of your Cortex-M4 processor determines which ETM functions are visible to the CTI. See Appendix A *Signal Descriptions* for details of recommended CTI connections for Cortex-M4 systems.

2.3 Operation

ETM-M4 implements version 3.5 of the ARM Embedded Trace Macrocell protocol. See the *Embedded Trace Macrocell Architecture Specification*.

Chapter 3
Programmers Model

This chapter describes the programmers model. It contains the following sections:

3.1 About the programmers model

This chapter describes the mechanisms for programming the registers used to set up the trace and triggering facilities of the macrocell. The programmers model enables you to use the ETM registers to control the macrocell.

3.2 Modes of operation and execution

ETM-M4 implements ETMv3.5 for tracing 16-bit and 32-bit Thumb instructions.

The *Embedded Trace Macrocell Architecture Specification* describes the features of ETMv3.5.

See *Features* on page 1-4 for information on the trace features of the ETM-M4.

When the ETM is powered up or reset, you must program all of the registers that do not have an architected reset state before you enable tracing. If you do not do so, the trace results are UNPREDICTABLE.

When programming the ETM registers you must enable all the changes at the same time. To achieve this, the Programming bit in ETMCR should be used. See *Main Control Register, ETMCR* on page 3-6.

When the Programming bit is set to 0 you must not write to registers other than ETMCR, because this can lead to UNPREDICTABLE behavior.

When setting the Programming bit, you must not change any other bits of ETMCR. You must only change the value of bits other than the Programming bit of ETMCR when bit [1] of ETMSR is set to 1. ARM recommends that you use a read-modify-write procedure when changing ETMCR.

3.3 Register summary

This section describes the ETM registers. Table 3-1 provides cross references to individual registers.

Table 3-1 ETM registers

Address	Name	Reset	Type	Description
`0xE0041000`	ETMCR	`0x00000411`	RW	*Main Control Register, ETMCR* on page 3-6
`0xE0041004`	ETMCCR	`0x8C802000`	RO	*Configuration Code Register, ETMCCR* on page 3-8
`0xE0041008`	ETMTRIGGER	-	RW	Trigger Event Register
`0xE0041010`	ETMSR	-	RW	ETM Status Register
`0xE0041014`	ETMSCR	`0x00020D09`	RO	*System Configuration Register, ETMSCR* on page 3-9
`0xE0041020`	ETMTEEVR	-	RW	TraceEnable Event Register
`0xE0041024`	ETMTECR1	-	RW	*TraceEnable Control 1 Register, ETMTECR1* on page 3-10
`0xE0041028`	ETMFFLR	-	RW	FIFOFULL Level Register
`0xE0041140`	ETMCNTRLDVR1	-	RW	Free-running counter reload value
`0xE00411E0`	ETMSYNCFR	`0x00000400`	RO	Synchronization Frequency Register
`0xE00411E4`	ETMIDR	`0x4114F250`	RO	*ID Register, ETMIDR* on page 3-11
`0xE00411E8`	ETMCCER	`0x18541800`	RO	*Configuration Code Extension Register, ETMCCER* on page 3-12
`0xE00411F0`	ETMTESSEICR	-	RW	*TraceEnable Start/Stop EmbeddedICE Control Register, ETMTESSEICR* on page 3-13
`0xE00411F8`	ETMTSEVR	-	RW	Timestamp Event Register
`0xE0041200`	ETMTRACEIDR	`0x00000000`	RW	CoreSight Trace ID Register
`0xE0041208`	ETMIDR2	`0x00000000`	RO	ETM ID Register 2
`0xE0041314`	ETMPDSR	`0x00000001`	RO	*Device Power-Down Status Register, ETMPDSR* on page 3-13
`0xE0041EE0`	ITMISCIN	-	RO	*Integration Test Miscellaneous Inputs, ITMISCIN* on page 3-14
`0xE0041EE8`	ITTRIGOUT	-	WO	*Integration Test Trigger Out, ITTRIGOUT* on page 3-15
`0xE0041EF0`	ETM_ITATBCTR2	-	RO	*ETM Integration Test ATB Control 2, ETM_ITATBCTR2* on page 3-15
`0xE0041EF8`	ETM_ITATBCTR0	-	WO	*ETM Integration Test ATB Control 0, ETM_ITATBCTR0* on page 3-16
`0xE0041F00`	ETMITCTRL	`0x00000000`	RW	Integration Mode Control Register
`0xE0041FA0`	ETMCLAIMSET	-	RW	Claim Tag Set Register
`0xE0041FA4`	ETMCLAIMCLR	-	RW	Claim Tag Clear Register
`0xE0041FB0`	ETMLAR	-	RW	Lock Access Register
`0xE0041FB4`	ETMLSR	-	RO	Lock Status Register
`0xE0041FB8`	ETMAUTHSTATUS	-	RO	Authentication Status Register
`0xE0041FCC`	ETMDEVTYPE	`0x00000013`	RO	CoreSight Device Type Register

Table 3-1 ETM registers (continued)

Address	Name	Reset	Type	Description
`0xE0041FD0`	ETMPIDR4	`0x00000004`	RO	Peripheral Identification Registers
`0xE0041FD4`	ETMPIDR5	`0x00000000`	RO	
`0xE0041FD8`	ETMPIDR6	`0x00000000`	RO	
`0xE0041FDC`	ETMPIDR7	`0x00000000`	RO	
`0xE0041FE0`	ETMPIDR0	`0x00000025`	RO	
`0xE0041FE4`	ETMPIDR1	`0x000000B9`	RO	
`0xE0041FE8`	ETMPIDR2	`0x0000000B`	RO	
`0xE0041FEC`	ETMPIDR3	`0x00000000`	RO	
`0xE0041FF0`	ETMCIDR0	`0x0000000D`	RO	Component Identification Registers
`0xE0041FF4`	ETMCIDR1	`0x00000090`	RO	
`0xE0041FF8`	ETMCIDR2	`0x00000005`	RO	
`0xE0041FFC`	ETMCIDR3	`0x000000B1`	RO	

3.4 Register descriptions

The following sections describe those registers whose implementation is specific to this product. Other registers are described in the *Embedded Trace Macrocell Architecture Specification*.

3.4.1 Main Control Register, ETMCR

The ETMCR characteristics are:

Purpose	Controls general operation of the ETM, such as whether tracing is enabled.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the ETM register summary in Table 3-1 on page 3-4.

Figure 3-1 shows the ETMCR bit assignments.

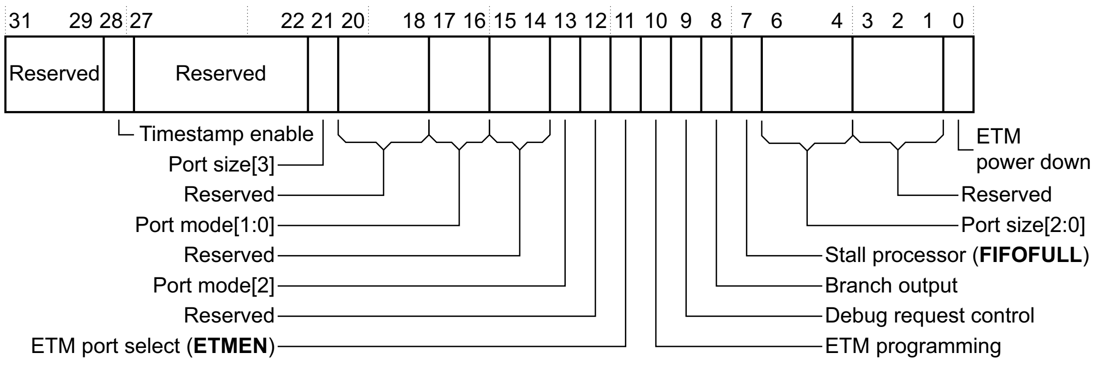

Figure 3-1 ETMCR bit assignments

Table 3-2 shows the ETMCR bit assignments.

Table 3-2 ETMCR bit assignments

Bits	Name	Function
[31:29]	-	RAZ
[28]	Timestamp enable	When set, this bit enables timestamping. An ETM reset sets this bit to 0.
[27:22]	-	RAZ
[21]	Port size[3]	This bit is implemented but has no function. An ETM reset sets this bit to 0.
[20:18]	-	Reserved
[17:16]	Port mode [1:0]	These bits are implemented but have no function. An ETM reset sets these bits to 0.
[15:14]	-	Reserved
[13]	Port mode[2]	This bit is implemented but has no function. An ETM reset sets this bit to 0.
[12]	-	Reserved

Table 3-2 ETMCR bit assignments (continued)

Bits	Name	Function
[11]	ETM port selection	This bit can be used to control other trace components in an implementation. The possible values are: **0** **ETMEN** is LOW. **1** **ETMEN** is HIGH. This bit must be set by the trace software tools to ensure that trace output is enabled from this ETM. An ETM reset sets this bit to 0.
[10]	ETM programming	This bit must be set to 1 at the start of the ETM programming sequence. Tracing is prevented while this bit is set to 1. On an ETM reset this bit is set to `0b1`.
[9]	Debug request control	When set to 1 and the trigger event occurs, the **ETMDBGRQ** output is asserted until **HALTED** is observed. This enables the ARM processor to be forced into Debug state. An ETM reset sets this bit to 0.
[8]	Branch output	When set to 1 all branch addresses are output, even if the branch was because of a direct branch instruction. Setting this bit enables reconstruction of the program flow without having access to the memory image of the code being executed. When this bit is set to 1, more trace data is generated, and this may affect the performance of the trace system. Information about the execution of a branch is traced regardless of the state of this bit. An ETM reset sets this bit to 0.
[7]	Stall processor	The **FIFOFULL** output can be used to stall the processor to prevent overflow. The **FIFOFULL** output is only enabled when the stall processor bit is set to 1. When the bit is 0 the **FIFOFULL** output remains LOW at all times and the FIFO overflows if there are too many trace packets. Trace resumes without corruption once the FIFO has drained, if overflow does occur. An ETM reset sets this bit to 0. For information about the interaction of this bit with the ETMFFLR register see the *Embedded Trace Macrocell Architecture Specification*.
[6:4]	Port size [2:0]	The ETM-M4 has no influence over the external pins used for trace. These bits are implemented but not used. On an ETM reset these bits reset to `0b001`.
[3:1]	-	Reserved
[0]	ETM power down	This bit can be used by an implementation to control if the ETM is in a low power state. This bit must be cleared by the trace software tools at the beginning of a debug session. When this bit is set to 1, writes to some registers and fields might be ignored. You can always write to the following registers and fields: • ETMCR bit [0] • ETMLAR • ETMCLAIMSET register • ETMCLAIMCLR register When the ETMCR is written with this bit set to 1, bits other than bit [0] might be ignored. On an ETM reset this bit is set to 1.

3.4.2 Configuration Code Register, ETMCCR

The ETM Configuration Code Register characteristics are:

Purpose	Enables software to read the implementation-specific configuration of the ETM.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the ETM register summary in Table 3-1 on page 3-4.

Figure 3-2 shows the ETMCCR bit assignments.

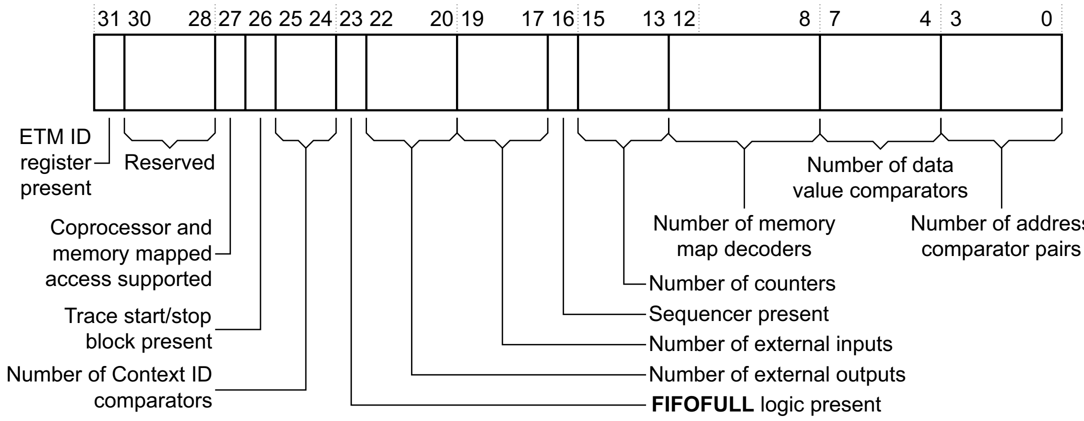
31 30 28 27 26 25 24 23 22 20 19 17 16 15 13 12 8 7 4 3 0
ETM ID register present
Reserved
Coprocessor and memory mapped access supported
Trace start/stop block present
Number of Context ID comparators
Number of data value comparators
Number of memory map decoders
Number of address comparator pairs
Number of counters
Sequencer present
Number of external inputs
Number of external outputs
FIFOFULL logic present

Figure 3-2 ETMCCR bit assignments

Table 3-3 shows the ETMCCR bit assignments.

Table 3-3 ETMCCR bit assignments

Bits	Name	Function
[31]	ETM ID register present	The value of this bit is 1, indicating that the ETMIDR, register `0x79`, is present and defines the ETM architecture version in use.
[30:28]	-	Reserved.
[27]	Coprocessor and memory access	The value of this bit is 1, indicating that memory-mapped access to registers is supported.
[26]	Trace start/stop block present	The value of this bit is 1, indicating that the Trace start/stop block is present.
[25:24]	Number of Context ID comparators	The value of these bits is `0b00`, indicating that Context ID comparators are not implemented.
[23]	**FIFOFULL** logic present	The value of this bit is 1, indicating that **FIFOFULL** logic is present in the ETM. To use FIFOFULL the system must also support the function, as indicated by bit [8] of ETMSCR, see *System Configuration Register, ETMSCR* on page 3-9.
[22:20]	Number of external outputs	The value of these bits is `0b000`, indicating that no external outputs are supported.
[19:17]	Number of external inputs	The value of these bits is between `0b000` and `0b010`, indicating the number of external inputs, from 0 to 2, implemented in the system.
[16]	Sequencer present	The value of this bit is 0, indicating that the sequencer is not implemented.

Table 3-3 ETMCCR bit assignments (continued)

Bits	Name	Function
[15:13]	Number of counters	The value of these bits is `0b001`, indicating that one counter is implemented.
[12:8]	Number of memory map decoders	The value of these bits is `0b00000`, indicating that memory map decoder inputs are not implemented.
[7:4]	Number of data value comparators	The value of these bits is `0b0000`, indicating that data value comparators are not implemented.
[3:0]	Number of address comparator pairs	The value of these bits is `0b0000`, indicating that address comparator pairs are not implemented.

3.4.3 System Configuration Register, ETMSCR

The ETMSCR characteristics are:

Purpose	Shows the ETM features supported by the implementation of the ETM macrocell.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-3 shows the ETMSCR bit assignments.

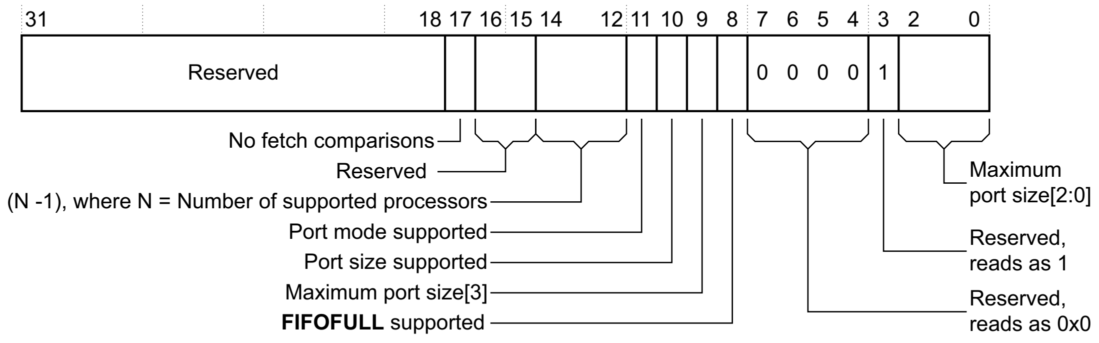

Figure 3-3 ETMSCR bit assignments

Table 3-4 shows the ETMSCR bit assignments.

Table 3-4 ETMSCR bit assignments

Bits	Name	Function
[31:18]	-	Reserved.
[17]	No Fetch comparisons	The value of this bit is 1, indicating that fetch comparisons are not implemented.
[16:15]	-	Reserved.
[14:12]	(N-1)	These bits give the number of supported processors minus 1. The value of these bits is `0b000`, indicating that there is only one processor connected.
[11]	Port mode supported	This bit reads as 1 if the currently selected port mode is supported. This has no effect on the TPIU trace port.

Table 3-4 ETMSCR bit assignments (continued)

Bits	Name	Function
[10]	Port size supported	This bit reads as 1 if the currently selected port size is supported. This has no effect on the TPIU trace port.
[9]	Maximum port size [3]	Maximum ETM port size bit [3]. This bit is used in conjunction with bits [2:0]. Its value is 0. This has no effect on the TPIU trace port.
[8]	**FIFOFULL** supported	The value of this bit is 1, indicating that **FIFOFULL** is supported. This bit is used in conjunction with bit [23] of the ETMCCR.
[7:4]	-	Reserved, Read-As-Zero.
[3]	-	Reserved, Read-As-One.
[2:0]	Maximum port size [2:0]	Maximum ETM port size bits [2:0]. These bits are used in conjunction with bit [9]. The value of these bits is `0b001`.

3.4.4 TraceEnable Control 1 Register, ETMTECR1

The ETMTECR1 characteristics are:

Purpose	Enables the start/stop logic used for trace enable.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-4 shows the ETMTECR1 bit assignments.

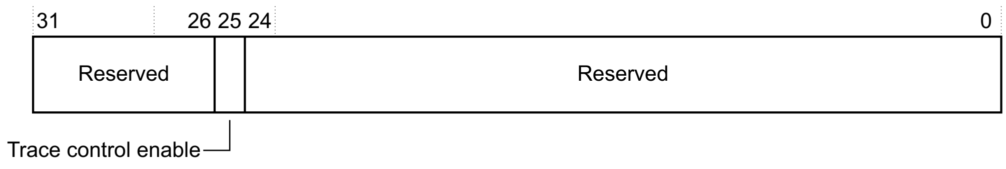

Figure 3-4 ETMTECR1 bit assignments

Table 3-5 shows the ETMTECR1 bit assignments.

Table 3-5 ETMTECR1 bit assignments

Bits	Name	Function
[31:26]	-	Reserved.
[25]	Trace control enable	Trace start/stop enable. The possible values of this bit are: **0** Tracing is unaffected by the trace start/stop logic. **1** Tracing is controlled by the trace on and off addresses configured for the trace start/stop logic. The trace start/stop resource, resource `0x5F`, is unaffected by the value of this bit.
[24:0]	-	Reserved.

3.4.5 ID Register, ETMIDR

The ETMIDR characteristics are:

Purpose	Holds the ETM architecture variant, and defines the programmers model for the ETM.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-5 shows the ETMIDR bit assignments.

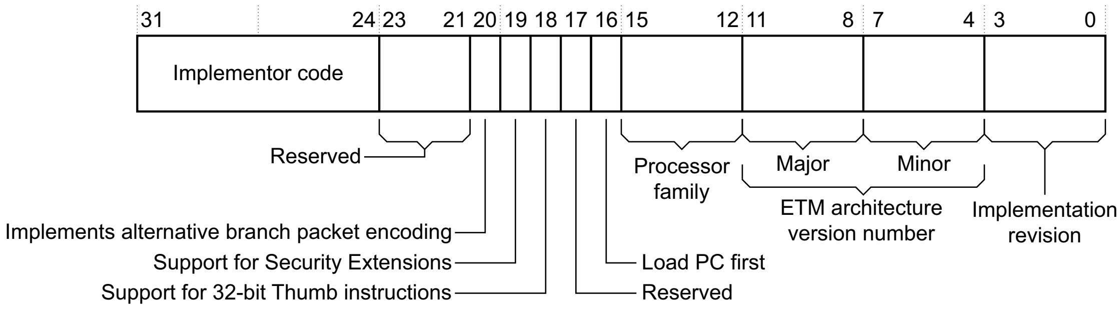
31
24 23
21 20 19 18 17 16 15
12 11
8 7
4 3
0
Implementor code
Reserved
Implements alternative branch packet encoding
Support for Security Extensions
Support for 32-bit Thumb instructions
Processor family
Major
Minor
ETM architecture version number
Implementation revision
Load PC first
Reserved

Figure 3-5 ETMIDR bit assignments

Table 3-6 shows the ETMIDR bit assignments.

Table 3-6 ETMIDR bit assignments

Bits	Name	Function
[31:24]	Implementer code	These bits identify ARM as the implementer of the processor. The value of these bits is `0b1000001`.
[23:21]	-	Reserved.
[20]	Branch packet encoding	The value of this bit is 1, indicating that alternative branch packet encoding is implemented.
[19]	Security Extensions support	The value of this bit is 0, indicating that the ETM behaves as if the processor is in Secure state at all times.
[18]	32-bit Thumb instruction tracing	The value of this bit is 1, indicating that a 32-bit Thumb instruction is traced as a single instruction.
[17]	-	Reserved.
[16]	Load PC first	The value of this bit is 0, indicating that data tracing is not supported.
[15:12]	Processor family	The value of these bits is `0b1111`, indicating that the processor family is not identified in this register.
[11:8]	Major ETM architecture version	The value of these bits is `0b0010`, indicating major architecture version number 3, ETMv3.
[7:4]	Minor ETM architecture version	The value of these bits is `0b0101`, indicating minor architecture version number 5.
[3:0]	Implementation revision	The value of these bits is `0b0000`, indicating implementation revision, 0.

3.4.6 Configuration Code Extension Register, ETMCCER

The ETMCCER characteristics are:

Purpose	Holds ETM configuration information additional to that in the ETMCCR. See *Configuration Code Register, ETMCCR* on page 3-8.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-6 shows the ETMCCER bit assignments.

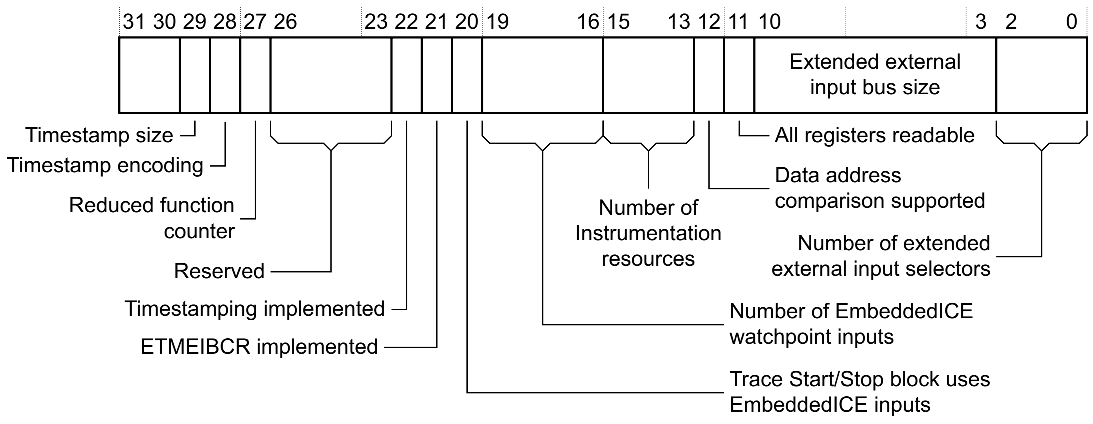
31 30 29 28 27 26 23 22 21 20 19 16 15 13 12 11 10 3 2 0
Extended external input bus size
Timestamp size
Timestamp encoding
Reduced function counter
Reserved
Timestamping implemented
ETMEIBCR implemented
Number of Instrumentation resources
All registers readable
Data address comparison supported
Number of extended external input selectors
Number of EmbeddedICE watchpoint inputs
Trace Start/Stop block uses EmbeddedICE inputs

Figure 3-6 ETMCCER bit assignments

Table 3-7 shows the ETMCCER bit assignments.

Table 3-7 ETMCCER bit assignments

Bits	Name	Function
[31:30]	-	Reserved. Read-As-Zero.
[29]	Timestamp size	Set to 0 to indicate a size of 48 bits.
[28]	Timestamp encoding	Set to 1 to indicate that the timestamp is encoded as a natural binary number.
[27]	Reduced function counter	Set to 1 to indicate that Counter 1 is a reduced function counter.
[26:23]	-	Reserved, Read-As-Zero.
[22]	Timestamping implemented	This bit is set to 1, indicating that timestamping is implemented.
[21]	EmbeddedICE behavior control implemented	The value of this bit is 0, indicating that the ETMEIBCR is not implemented. For more information on EmbeddedICE behavior see the *Embedded Trace Macrocell Architecture Specification*.
[20]	Trace Start/Stop block uses EmbeddedICE watchpoint inputs	The value of this bit is 1, indicating that the Trace Start/Stop block uses the EmbeddedICE watchpoint inputs.
[19:16]	EmbeddedICE watchpoint inputs	The value of these bits is `0b0100`, indicating that the number of EmbeddedICE watchpoint inputs implemented is four. These inputs come from the DWT.
[15:13]	Instrumentation resources	The value of these bits is `0b000`, indicating that no Instrumentation resources are supported.

Table 3-7 ETMCCER bit assignments (continued)

Bits	Name	Function
[12]	Data address comparisons	The value of this bit is 1, indicating that data address comparisons are not supported.
[11]	Readable registers	The value of this bit is 1, indicating that all registers are readable.
[10:3]	Extended external input bus	The value of these bits is 0, indicating that the extended external input bus is not implemented.
[2:0]	Extended external input selectors	The value of these bits is 0, indicating that extended external input selectors are not implemented.

3.4.7 TraceEnable Start/Stop EmbeddedICE Control Register, ETMTESSEICR

The ETMTESSEICR characteristics are:

Purpose	Specifies the EmbeddedICE watchpoint comparator inputs that are used to control the start/stop resource.
Usage constraints	There are no usage constraints.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-7 shows the ETMTESSEICR bit assignments.

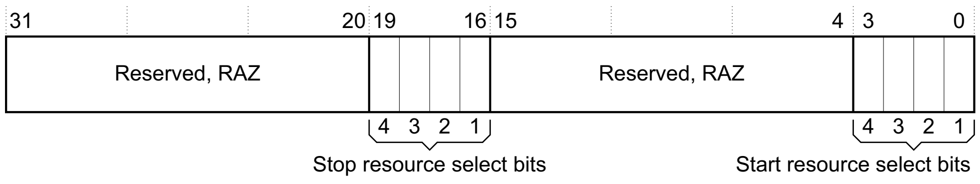

Figure 3-7 ETMTESSEICR bit assignments

Table 3-8 shows the ETMTESSEICR bit assignments.

Table 3-8 ETMTESSEICR bit assignments

Bits	Name	Function
[31:20]	-	Reserved, Read-as-zero.
[19:16]	Stop resource selection	Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a **TraceEnable** stop resource. Bit [16] corresponds to input 1, bit [17] corresponds to input 2, bit [18] corresponds to input 3, and bit [19] corresponds to input 4.
[15:4]	-	Reserved, Read-As-Zero.
[3:0]	Start resource selection	Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a **TraceEnable** start resource. Bit [0] corresponds to input 1, bit [1] corresponds to input 2, bit [2] corresponds to input 3, and bit [3] corresponds to input 4.

3.4.8 Device Power-Down Status Register, ETMPDSR

The ETMPDSR characteristics are:

Purpose	Indicates the power-down status of the ETM.

Usage constraints There are no usage constraints.

Configurations This register is only available if the processor is configured to use an ETM.

Attributes See the register summary in Table 3-1 on page 3-4

Figure 3-8 shows the ETMPDSR bit assignments.

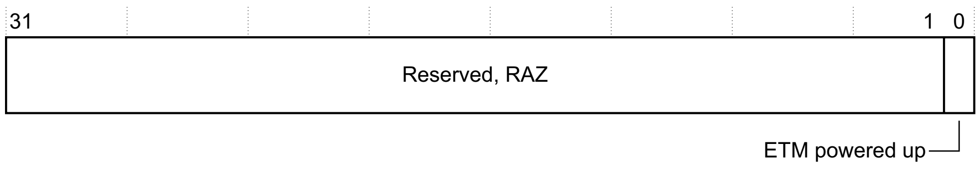

Figure 3-8 ETMPDSR bit assignments

Table 3-9 shows the ETMPDSR bit assignments.

Table 3-9 ETMPDSR bit assignments

Bits	Name	Function
[31:1]	-	Reserved, Read-As-Zero.
[0]	ETM powered up	The value of this bit indicates whether you can access the ETM Trace Registers. The value of this bit is always 1, indicating that the ETM Trace Registers can be accessed.

3.4.9 Integration Test Miscellaneous Inputs, ITMISCIN

The ITMISCIN characteristics are:

Purpose Integration test.

Usage constraints There are no usage constraints.

Configurations This register is only available if the processor is configured to use the ETM.

Attributes See the register summary in Table 3-1 on page 3-4.

Figure 3-9 shows the ITMISCIN bit assignments.

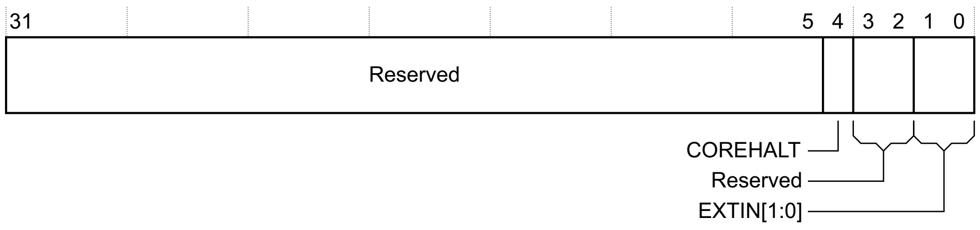

Figure 3-9 ITMISCIN bit assignments

Table 3-10 shows the ITMISCIN bit assignments.

Table 3-10 ITMISCIN bit assignments

Bits	Name	Function
[31:5]	-	Reserved.
[4]	COREHALT	A read of this bit returns the value of the **COREHALT** input pin.
[3:2]	-	Reserved.
[1:0]	EXTIN[1:0]	A read of these bits returns the value of the **EXTIN[1:0]** input pins.

3.4.10 Integration Test Trigger Out, ITTRIGOUT

The ITMISCIN characteristics are:

Purpose	Integration test.
Usage constraints	You must set bit [0] of ETMITCTRL to use this register.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4

Figure 3-10 shows the ITTRIGOUT bit assignments.

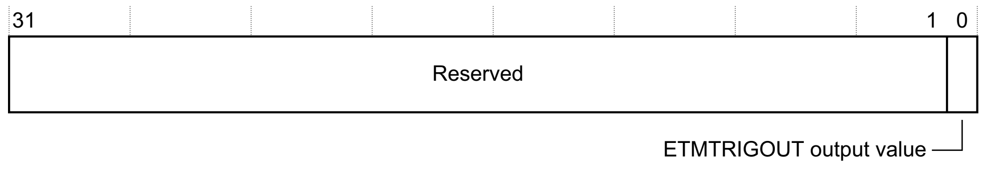

Figure 3-10 ITTRIGOUT bit assignments

Table 3-11 shows the ITTRIGOUT bit assignments.

Table 3-11 ITTRIGOUT bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	TRIGGER output value	A write to this bit sets the ETMTRIGOUT output.

3.4.11 ETM Integration Test ATB Control 2, ETM_ITATBCTR2

The ETM_ITATBCTR2 characteristics are:

Purpose	Integration test.
Usage constraints	You must set bit [0] of ETMITCTRL to use this register.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4

Figure 3-11 on page 3-16 shows the ETM_ITATBCTR2 bit assignments.

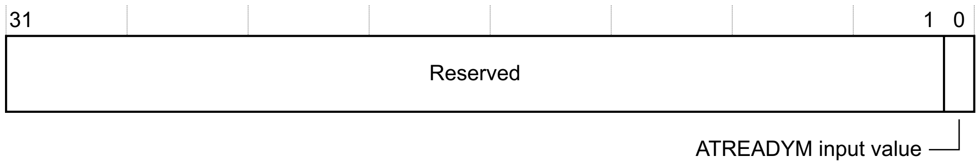

Figure 3-11 ETM_ITATBCTR2 bit assignments

Table 3-12 shows the ETM_ITATBCTR2 bit assignments.

Table 3-12 ETM_ITATBCTR2 bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	ATREADY input value	A read of this bit returns the value of the ETM ATREADYM input.

3.4.12 ETM Integration Test ATB Control 0, ETM_ITATBCTR0

The ETM_ITATBCTR0 characteristics are:

Purpose	Integration test.
Usage constraints	You must set bit [0] of ETMITCTRL to use this register.
Configurations	This register is only available if the processor is configured to use the ETM.
Attributes	See the register summary in Table 3-1 on page 3-4.

Figure 3-12 shows the ETM_ITATBCTR0 bit assignments.

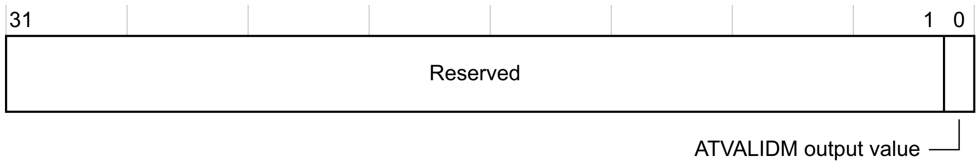

Figure 3-12 ETM_ITATBCTR0 bit assignments

Table 3-13 shows the ETM_ITATBCTR0 bit assignments.

Table 3-13 ETM_ITATBCTR0 bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	ATVALID output value	A write to this bit sets the value of the ETM ATVALIDM output.

Appendix A
Signal Descriptions

This appendix describes the signals used in the macrocell. It contains the following sections:

- *CTI signal descriptions* on page A-2.

A.1 CTI signal descriptions

Table A-1 and Table A-2 show the recommended *Cross Trigger Interface* (CTI) connections for Cortex-M4 systems.

Note

These tables show the ARM standard connections, but the actual connections are implementation-defined. Check the documentation from the supplier of your device for any changes to these connections.

Table A-1 Input connections

Trigger bit	Source signal	Source device	Comments
[7]	**ETMTRIGOUT**	ETM	Recommended if ETM is present.
[6]	**ETMTRIGGER[2]**	DWT	Recommended.
[5]	**ETMTRIGGER[1]**	DWT	Recommended.
[4]	**ETMTRIGGER[0]**	DWT	Recommended.
[3]	**ACQCOMP**	ETB	Recommended if an *Embedded Trace Buffer* (ETB) is present. If multiple cores share a single ETB, you must only connect to the CTI of one of the cores.
[2]	**FULL**	ETB	
[1]	User Defined	-	-
[0]	**HALTED**	Core	Compulsory.

Table A-2 Trigger output connections

Trigger bit	Destination signal	Destination device	Comments
[7]	User defined	-	-
[6]	User defined	-	-
[5]	**ETMEXTIN[1]**	ETM	Compulsory if ETM is present.
[4]	**ETMEXTIN[0]**	ETM	Compulsory if ETM is present.
[3]	**INTISR[y]**	NVIC	Recommended if an ETB is present. If multiple cores share a single ETB, you must only connect to the CTI of one of the cores.
[2]	**INTISR[x]**	NVIC	Compulsory. Any interrupt can be used.
[1]	User defined	-	-
[0]	**EDBGRQ**	Core	Compulsory.

Appendix B
Revisions

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

Table B-1 issue A

Change	Location	Affects
First release	-	-

Table B-2 Differences between issue A and issue B

Change	Location	Affects
No technical changes	-	-

Table B-3 Differences between issue B and issue C

Change	Location	Affects
Updated ETMCR bit assignment information.	Table 3-2 on page 3-6	All
Updated ITTRIGOUT bit assignments.	Figure 3-10 on page 3-15 Table 3-11 on page 3-15	All

Table B-3 Differences between issue B and issue C (continued)

Change	Location	Affects
Updated ETM Integration Test ATB Control2 bit assignments.	Figure 3-11 on page 3-16 Table 3-12 on page 3-16	All
Updated ETM Integration test ATB Control 0 bit assinments.	Figure 3-12 on page 3-16 Table 3-13 on page 3-16	All
Updated CTI signal descriptions Input connections.	Table A-1 on page A-2	All

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced eXtensible Interface (AXI)

A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

The AXI protocol also includes optional extensions to cover signaling for low-power operation.

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced High-performance Bus (AHB)

A bus protocol with a fixed pipeline between address/control and data phases. It only supports a subset of the functionality provided by the AMBA™ AXI protocol. The full AMBA AHB protocol specification includes a number of features that are not commonly required for master and slave IP developments and ARM recommends only a subset of the protocol is usually used. This subset is defined as the AMBA AHB-Lite protocol.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture (AMBA)

A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that describes in detail a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

See also Advanced High-performance Bus and AHB-Lite.

Advanced Peripheral Bus (APB)

A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

See also Advanced High-performance Bus.

AHB

See Advanced High-performance Bus.

AHB-Lite

A subset of the full AMBA AHB protocol specification. It provides all of the basic functions required by the majority of AMBA AHB slave and master designs, particularly when used with a multi-layer AMBA interconnect. In most cases, the extra facilities provided by a full AMBA AHB interface are implemented more efficiently by using an AMBA AXI protocol interface.

Aligned

A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA

See Advanced Microcontroller Bus Architecture.

APB

See Advanced Peripheral Bus.

Architecture

The organization of hardware and/or software that characterizes a processor and its attached components, and enables devices with similar characteristics to be grouped together when describing their behavior, for example, Harvard architecture, instruction set architecture, ARMv6 architecture.

ARM instruction

A word that specifies an operation for an ARM processor in ARM state to perform. ARM instructions are word-aligned.

See also ARM state, Thumb instruction, Thumb-2EE instruction.

ARM state

An operating state of the processor, in which it executes 32-bit ARM instructions.

See also ARM instruction, Thumb state, ThumbEE state.

AXI

See Advanced eXtensible Interface.

Big-endian

Byte ordering scheme in which bytes of decreasing significance in a data word are stored at increasing addresses in memory.

See also Little-endian and Endianness.

Branch folding

Branch folding is a technique where, on the prediction of most branches, the branch instruction is completely removed from the instruction stream presented to the execution pipeline. Branch folding can significantly improve the performance of branches, taking the CPI for branches below one.

Branch prediction

The process of predicting if conditional branches are to be taken or not in pipelined processors. Successfully predicting if branches are to be taken enables the processor to prefetch the instructions following a branch before the condition is fully resolved. Branch prediction can be done in software or by using custom hardware. Branch prediction techniques are categorized as static, in which the prediction decision is decided before run time, and dynamic, in which the prediction decision can change during program execution.

Breakpoint

A breakpoint is a mechanism provided by debuggers to identify an instruction at which program execution is to be halted. Breakpoints are inserted by the programmer to enable inspection of register contents, memory locations, variable values at fixed points in the program execution to test that the program is operating correctly. Breakpoints are removed after the program is successfully tested.

See also Watchpoint.

Burst

A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

Byte

An 8-bit data item.

Byte invariant

In a byte-invariant system, the address of each byte of memory remains unchanged when switching between little-endian and big-endian operation. When a data item larger than a byte is loaded from or stored to memory, the bytes making up that data item are arranged into the correct order depending on the endianness of the memory access.
The ARM architecture supports byte-invariant systems in ARMv6 and later versions. When byte-invariant support is selected, unaligned halfword and word memory accesses are also supported. Multi-word accesses are expected to be word-aligned.

See also Word-invariant.

Byte lane strobe

A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.

Clock gating

Gating a clock signal for a macrocell with a control signal, and using the modified clock that results to control the operating state of the macrocell.

Cold reset

Also known as power-on reset. Starting the processor by turning power on. Turning power off and then back on again clears main memory and many internal settings. Some program failures can lock up the processor and require a cold reset to enable the system to be used again. In other cases, only a warm reset is required.

See also Warm reset.

Coprocessor

A processor that supplements the main processor. It carries out additional functions that the main processor cannot perform. Usually used for floating-point math calculations, signal processing, or memory management.

Core

A core is that part of a processor that contains the ALU, the datapath, the general-purpose registers, the Program Counter, and the instruction decode and control circuitry.

Core reset

See Warm reset.

CoreSight

The infrastructure for monitoring, tracing, and debugging a complete system on chip.

CPI

See Cycles per instruction.

Cycles Per instruction (CPI)

Cycles per instruction (or clocks per instruction) is a measure of the number of computer instructions that can be performed in one clock cycle. This figure of merit can be used to compare the performance of different CPUs that implement the same instruction set against each other. The lower the value, the better the performance.

DAP

See Debug Access Port.

Debug Access Port (DAP)
A TAP block that acts as an AMBA, AHB or AHB-Lite, master for access to a system bus. The DAP is the term used to encompass a set of modular blocks that support system wide debug. The DAP is a modular component, intended to be extendable to support optional access to multiple systems such as memory mapped AHB and CoreSight APB through a single debug interface.

Doubleword A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.

EmbeddedICE logic An on-chip logic block that provides TAP-based debug support for ARM processor cores. It is accessed through the TAP controller on the ARM core using the JTAG interface.

EmbeddedICE-RT The JTAG-based hardware provided by debuggable ARM processors to aid debugging in real-time.

Embedded Trace Buffer (ETB)
The ETB provides on-chip storage of trace data using a configurable sized RAM.

Embedded Trace Macrocell (ETM)
A hardware macrocell that outputs instruction and data trace information on a trace port.

Endianness Byte ordering. The scheme that determines the order in which successive bytes of a data word are stored in memory.

See also Little-endian and Big-endian.

ETB *See* Embedded Trace Buffer.

ETM *See* Embedded Trace Macrocell.

Exception A fault or error event that is considered serious enough to require that program execution is interrupted. Examples include attempting to perform an invalid memory access, external interrupts, and undefined instructions. When an exception occurs, normal program flow is interrupted and execution is resumed at the corresponding exception vector. This contains the first instruction of the interrupt handler to deal with the exception.

Exception vector *See* Interrupt vector.

Fast context switch

In a multitasking system, the point at which the time-slice allocated to one process stops and the one for the next process starts. If processes are switched often enough, they can appear to a user to be running in parallel, in addition to being able to respond quicker to external events that might affect them.

In ARM processors, a fast context switch is caused by the selection of a non-zero PID value to switch the context to that of the next process. A fast context switch causes each Virtual Address for a memory access, generated by the ARM processor, to produce a Modified Virtual Address that is sent to the rest of the memory system to be used in place of a normal Virtual Address. For some cache control operations Virtual Addresses are passed to the memory system as data. In these cases no address modification takes place.

See also Fast Context Switch Extension.

Fast Context Switch Extension (FCSE)
An extension to the ARM architecture that enables cached processors with an MMU to present different addresses to the rest of the memory system for different software processes, even when those processes are using identical addresses.

See also Fast context switch.

FCSE *See* Fast Context Switch Extension.

Flat address mapping

A system of organizing memory in which each physical address contained within the memory space is the same as its corresponding virtual address.

Half-rate clocking

Half-rate clocking is a feature of the ETM. It means dividing the trace clock by two so that the TPA can sample trace data signals on both the rising and falling edges of the trace clock. The primary purpose of half-rate clocking is to reduce the signal transition rate on the trace clock of an ASIC for very high-speed systems.

Halting debug-mode

One of two mutually exclusive debug modes. In Halting debug-mode a *debug event*, such as a a breakpoint or watchpoint, causes the processor to enter a special Debug state. In Debug state the processor is controlled through the external debug interface. This interface also provides access to all processor state, coprocessor state, memory and input/output locations.

See also Monitor debug-mode.

High vectors

Alternative locations for exception vectors. The high vector address range is near the top of the address space, rather than at the bottom.

Implementation-defined

Behavior that is not architecturally defined, but is defined and documented by individual implementations.

Instruction cycle count

The number of cycles for which an instruction occupies the Execute stage of the pipeline.

Interrupt vector

One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.

See also High vectors.

Little-endian

Byte ordering scheme in which bytes of increasing significance in a data word are stored at increasing addresses in memory.

See also Big-endian and Endianness.

Monitor debug-mode

One of two mutually exclusive debug modes. In Monitor debug-mode a *debug event*, such as a breakpoint or a watchpoint, causes a debug exception, generating either a Prefetch Abort exception or a Data Abort exception.

See also Halting debug-mode.

Power-on reset

See Cold reset.

Prefetching

In pipelined processors, the process of fetching instructions from memory to fill up the pipeline before the preceding instructions have finished executing. Prefetching an instruction does not mean that the instruction must be executed.

Processor

A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

Read

Reads are defined as memory operations that have the semantics of a load. That is, the ARM instructions LDM, LDRD, LDC, LDR, LDRT, LDRSH, LDRH, LDRSB, LDRB, LDRBT, LDREX, RFE, STREX, SWP, and SWPB, and the Thumb instructions LDM, LDR, LDRSH, LDRH, LDRSB, LDRB, and POP.

Reserved
A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

SBO
See Should Be One.

SBZ
See Should Be Zero.

SBZP
See Should Be Zero or Preserved.

Scan chain
A scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

Should Be One (SBO)
Should be written as 1 (or all 1s for bit fields) by software. Writing 0 produces Unpredictable results.

Should Be Zero (SBZ)
Should be written as 0 (or all 0s for bit fields) by software. Writing 1 produces Unpredictable results.

Should Be Zero or Preserved (SBZP)
Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing back the same value that has been previously read from the same field on the same processor.

Synchronization primitive
The memory synchronization primitive instructions are instructions that are used to ensure memory synchronization, that is, the LDREX, STREX, SWP, and SWPB instructions.

TAP
See Test Access Port.

Test Access Port (TAP)
The collection of four mandatory and one optional terminals that form the input/output and control interface to a JTAG boundary-scan architecture. The mandatory terminals are **TDI**, **TDO**, **TMS**, and **TCK**. The optional terminal is **TRST**. This signal is mandatory in ARM cores because it resets the debug logic.

Thumb instruction
One or two halfwords that specify an operation for an ARM processor in Thumb state to perform. Thumb instructions must be halfword-aligned. In the original Thumb ISA, all instructions are 16-bit. The Thumb-2 extension of the ISA provides both 16-bit and 32-bit instructions.

See also ARM instruction, Thumb state, Thumb-2EE instruction.

Thumb state
An operating state of the processor, in which it executes 16-bit and 32-bit Thumb instructions.

See also ARM state, Thumb instruction, ThumbEE state.

Thumb-2EE instruction
One or two halfwords that specify an operation for an ARM processor in ThumbEE state to perform. Thumb-2EE instructions must be halfword-aligned.

Thumb-2EE is an extension of the Thumb-2 architecture designed as a target for dynamically generated code, that is, code compiled on the device either shortly before or during execution from a portable bytecode or other intermediate or native representation.

See also ARM instruction, Thumb instruction, ThumbEE state.

TPA

See *Trace Port Analyzer*.

Trace Port Analyzer (TPA)

A hardware device that captures trace information output on a trace port. This can be a low-cost product designed specifically for trace acquisition, or a logic analyzer.

Undefined

Indicates an instruction that generates an Undefined instruction trap. See the *ARM Architectural Reference Manual* for more information on ARM exceptions.

UNP

See Unpredictable.

Unpredictable (UNP)

For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.

In an ETM context, means that the behavior of the ETM cannot be relied on. Such conditions have not been validated. When applied to the programming of an event resource, only the output of that event resource is Unpredictable.

Unpredictable ETM behavior can affect the behavior of the entire system, because the ETM is capable of causing the core to enter debug state, and external outputs can be used for other purposes.

Warm reset

Also known as a core reset. Initializes the majority of the processor excluding the debug controller and debug logic. This type of reset is useful if you are using the debugging features of a processor.

Watchpoint

A watchpoint is a mechanism provided by debuggers to halt program execution when the data contained by a particular memory address is changed. Watchpoints are inserted by the programmer to enable inspection of register contents, memory locations, and variable values when memory is written to test that the program is operating correctly. Watchpoints are removed after the program is successfully tested.

See also Breakpoint.

Word-invariant

In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged.

The ARM architecture supports word-invariant systems in ARMv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions that are given unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. It is recommended that word-invariant systems use the endianness that produces the desired byte addresses at all times, apart possibly from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler must use only aligned word memory accesses.

See also Byte-invariant.

Write

Writes are defined as operations that have the semantics of a store. That is, the ARM instructions SRS, STM, STRD, STC, STRT, STRH, STRB, STRBT, STREX, SWP, and SWPB, and the Thumb instructions STM, STR, STRH, STRB, and PUSH.