CoreSight™ ETM9™

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0315B

CoreSight ETM9
Technical Reference Manual

Release Information

Change history

Date	Issue	Confidentiality	Change
23 March 2005	A	Non-Confidential	First release
14 June 2007	B	Non-Confidential	New issue for r0p1 release. Appendix C *Typical APB Transfers* removed. Further minor updates and corrections throughout document

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

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

Web Address

`http://www.arm.com`

Contents
CoreSight ETM9 Technical Reference Manual

List of Tables
CoreSight ETM9 Technical Reference Manual

List of Figures
CoreSight ETM9 Technical Reference Manual

Preface

This preface introduces the *CoreSight ETM9 r0p1 Technical Reference Manual* (TRM). It contains the following sections:

About this manual

This is the TRM for CoreSight ETM9.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this manual, 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 manual is written for the following target audiences:

- Designers of development tools providing support for ETM functionality. Implementation-specific behavior is described in this document. These users can also consult the *ETM Architecture Specification*, ARM IHI 0014.
- Hardware and software engineers integrating the CoreSight ETM9 into an ASIC that includes an ARM9 processor. These users can also consult the *CoreSight ETM9 Integration Manual,* ARM DII 0094.

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the functionality of CoreSight ETM9.

Chapter 2 *Implementation-defined Behavior*

Read this chapter for a description of the implementation-defined features of CoreSight ETM9.

Chapter 3 *Programmer's Model*

Read this chapter for a description of the programmer's model for CoreSight ETM9.

Chapter 4 *Blocks for Stand-alone CoreSight ETM9*

CoreSight ETM9 can be used stand-alone, that is, without a CoreSight subsystem, to trace an ARM9 family processor. This chapter describes the additional blocks provided for CoreSight ETM9 used stand-alone

Appendix A *Signals Lists*

Read this appendix for a description of the CoreSight ETM9 signals.

Appendix B *I/O Signal Timings*

Read this appendix for a description of the CoreSight ETM9 timing parameters.

Glossary Read the Glossary for definitions of terms used in this manual.

Conventions

Conventions that this manual uses are described in the following sections:

- *Typographical*
- *Timing diagrams* on page xii
- *Signal naming* on page xii
- *Numbering* on page xiii.

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.

`mono`space Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic` Denotes arguments to monospace text where the argument is to be replaced by a specific value.

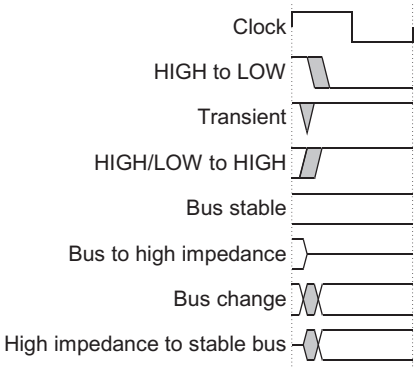
`monospace bold` Denotes language keywords when used outside example code.

< and > Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example:

- `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
- The Opcode_2 value selects which register is accessed.

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams. Shaded bus and signal areas are undefined, and the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signal naming

When a signal is described as asserted, the level depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals:

Prefix A *Advanced eXtensible Interface* (AXI) global and address channel signals are prefixed by an upper case A.

Prefix B AXI write response channel signals are prefixed by an upper case B.

Prefix C AXI low-power interface signals are prefixed by an upper case C.

Prefix H Advanced High-performance Bus (AHB) signals are prefixed by an upper case H.

Prefix n Active-low signals are prefixed by a lowercase n except in the case of AHB or *Advanced Peripheral Bus* (APB) reset signals. These are named **HRESETn** and **PRESETn** respectively.

Prefix P APB signals are prefixed by an upper case P.

Prefix R AXI read channel signals are prefixed by an upper case R.

Prefix W AXI write channel signals are prefixed by an upper case W.

Numbering

<size in bits>'<base><number>

This is a Verilog method of abbreviating constant numbers. For example:

- 'h7B4 is an unsized hexadecimal value.
- 'o7654 is an unsized octal value.
- 8'd9 is an eight-bit wide decimal value of 9.
- 8'h3F is an eight-bit wide hexadecimal value of `0x3F`. This is equivalent to b00111111.
- 8'b1111 is an eight-bit wide binary value of b00001111.

Further reading

This section lists publications by ARM Limited, and by third parties.

ARM Limited periodically provides updates and corrections to its documentation. See `http://www.arm.com` for current errata sheets, addenda, and the Frequently Asked Questions list.

ARM publications

This manual contains information that is specific to the CoreSight ETM9. See the following documents for other relevant information:

- *AMBA 3 APB Protocol Specification* (ARM IHI 0024)
- *CoreSight ETM9 Integration Manual* (ARM DII 0094)
- *CoreSight ETM9 Configuration and Sign-off Guide (*ARM DII 0191)
- *ETM Architecture Specification* (ARM IHI 0014)
- *CoreSight Architecture Specification* (ARM IHI 0029)
- *CoreSight Components TRM* (ARM DDI 0314).

In addition, see the following documentation for specific information relating to ARM products:

- *ARM Reference Peripheral Specification* (ARM DDI 0062)
- the ARM datasheet or technical reference manual for the core to which the ETM9 macrocell is to be connected.

Feedback

ARM Limited welcomes feedback on the CoreSight ETM9 and its documentation.

Feedback on the product

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

- the product name
- a concise explanation of your comments.

Feedback on this book

If you have any comments on this book, please send email to `errata@arm.com` giving:

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

ARM Limited also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces CoreSight ETM9 macrocell. It contains the following sections:

1.1 About CoreSight ETM9

CoreSight ETM9 provides instruction trace and data trace for the CoreSight-compatible ARM9 microprocessors. It provides instruction trace and data trace for:

- ARM926EJ-S processors
- ARM946E-S processors
- ARM966E-S processors
- ARM968E-S processors.

Note

CoreSight ETM9 does not support processors using the ARM9TDMI CPU core. This means it does not support the ARM920T and ARM922T processors. The ETM9 r2p2 macrocell must be used to provided trace for these products.

You can use the CoreSight ETM9 macrocell in two contexts:

- Stand-alone, as the ETM9CSSingle module. You can use the ETM9CSSingle module to trace only a single ARM9 CPU at any one time, for example an ARM946E-S processor. See Chapter 4 *Blocks for Stand-alone CoreSight ETM9* for information about using a stand-alone CoreSight ETM9.
- In a CoreSight System, as the ETM9CS module. See the *CoreSight Design Kit TRM* for information about using CoreSight ETM9 in a CoreSight system.

Figure 1-1 shows the main functional blocks and clock domains of CoreSight ETM9.

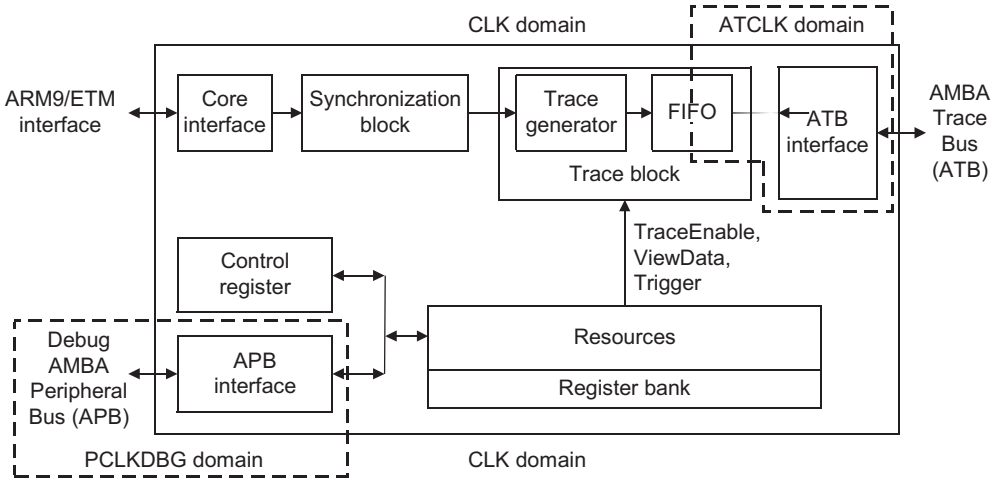


Figure 1-1 CoreSight ETM9 functional blocks and clock domains

See Appendix A *Signals Lists* for information about CoreSight ETM9 signals.

See the *Embedded Trace Macrocell Architecture Specification* for information about the trace protocol, and about controlling tracing using triggering and filtering resources.

Programming access to CoreSight ETM9 is through the APB interface:

- In a CoreSight system, the APB interface connects to the Debug APB bus.
- You program a stand-alone CoreSight ETM9 using the ETMJTAGPORT. For information about the ETMJTAGPORT see *ETMJTAGPORT* on page 4-3.

1.2 CoreSight ETM9 configuration

Table 1-1 shows the CoreSight ETM9 resources.

Table 1-1 CoreSight ETM9 resources

Resource description	Number/size
Context ID comparators	1
External inputs	4
External outputs	2
Sequencers	1
Counters	2
Memory map decoders	0
Data comparators	2
Pairs of address comparators	4
FIFO size	60 bytes
Trace port configuration	Variable

1.3 Product revisions

This is the Technical Reference Manual for CoreSight ETM9 r0p1.This section summarizes the differences in functionality between the releases of the CoreSight ETM9 macrocell.

r0p0 - r0p1 Incorporates engineering errata fixes. It contains no differences in functionality.

Chapter 2
Implementation-defined Behavior

This chapter describes the implementation-defined behavior of the CoreSight ETM9. It contains the following sections:

2.1 ETM architecture version

CoreSight ETM9 implements version 3.2 of the ETM architecture, ETMv3.2. See the *ETM Architecture Specification* for more information.

2.2 CoreSight ETM9 registers summary

Table 2-1 lists the implementation-specific CoreSight ETM9 registers.

Table 2-1 CoreSight ETM9 registers summary

Name	Base offset	Type	Reset value	Description
Configuration Code	0x004	RO	0x8D294024	See *Configuration Code Register* on page 2-6.
ID	0x1E4	RO	0x41001221	See *ETM ID Register* on page 2-5.
Configuration Code Extension	0x1E8	RO	0x000008A2	See *Configuration Code Extension Register* on page 2-8.
ITMISCOUT	0xEDC	WO	-	See *ITMISCOUT Register (miscellaneous outputs)* on page 2-13.
ITMISCIN	0xEE0	RO	-[a]	See *ITMISCIN Register (miscellaneous inputs)* on page 2-14.
ITTRIGGERACK	0xEE4	RO	-[a]	See *ITTRIGGERACK Register (trigger acknowledge)* on page 2-14.
ITTRIGGERREQ	0xEE8	WO	-	See *ITTRIGGERREQ Register (trigger request)* on page 2-15.
ITATBDATA0	0xEEC	WO	-	See *ITATBDATA0 Register (ATB data 0)* on page 2-16.
ITATBCTR2	0xEF0	RO	-[a]	See *ITATBCTR2 Register (ATB control 2)* on page 2-16.
ITATBCTR1	0xEF4	WO	-	See *ITATBCTR1 Register (ATB control 1)* on page 2-17.
ITATBCTR0	0xEF8	WO	-	See *ITATBCTR0 Register (ATB control 0)* on page 2-17.
Device Configuration	0xFC8	RO	0x00000000	Indicates no user-definable functionality. See the *ETM Architecture Specification*.
Device Type	0xFCC	RO	0x00000013	Indicates a processor trace source. See the *ETM Architecture Specification*.

Table 2-1 CoreSight ETM9 registers summary (continued)

Name	Base offset	Type	Reset value	Description
Peripheral ID4	0xFD0	RO	0x00000004	See *Peripheral Identification Registers* on page 2-9.
Peripheral ID5	0xFD4	RO	0x00000000	
Peripheral ID6	0xFD8	RO	0x00000000	
Peripheral ID7	0xFDC	RO	0x00000000	
Peripheral ID0	0xFE0	RO	0x00000010	
Peripheral ID1	0xFE4	RO	0x000000B9	
Peripheral ID2	0xFE8	RO	0x000000XB[b]	
Peripheral ID3	0xFEC	RO	0x00000000	
Component ID0	0xFF0	RO	0x0000000D	See *Component Identification Registers* on page 2-10.
Component ID1	0xFF4	RO	0x00000090	
Component ID2	0xFF8	RO	0x00000005	
Component ID3	0xFFC	RO	0x000000B1	

a. The values of these read-only registers depend on the signals on external pins of the CoreSight ETM9. Therefore it is not possible to define the register reset values.
b. See *Peripheral Identification Registers* on page 2-9 for the value of X, bits [7:4] of the register value.

Note

For all other CoreSight ETM9 registers, see the *ETM Architecture Specification*.

2.3 CoreSight ETM9 register descriptions

This section describes the following registers:

- *ETM ID Register*
- *Configuration Code Register* on page 2-6
- *Configuration Code Extension Register* on page 2-8
- *Component Identification Registers* on page 2-10
- *Peripheral Identification Registers* on page 2-9.

2.3.1 ETM ID Register

The ETM ID Register, at offset `0x1E4`, is read-only. Figure 2-1 shows the register bit assignments.

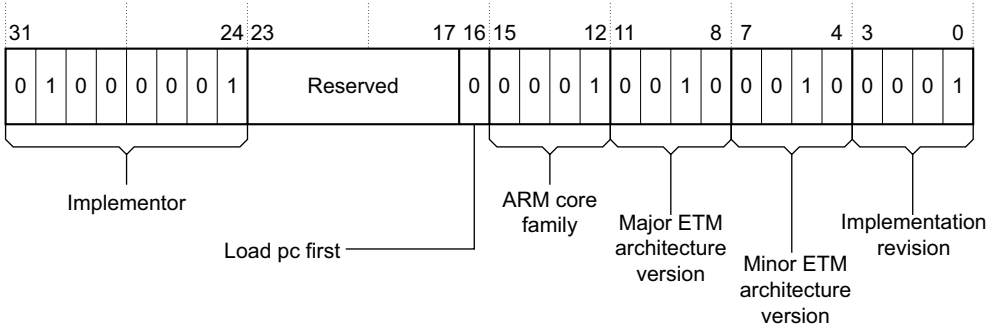


Figure 2-1 ETM ID Register bit assignments

Table 2-2 shows the value of the fields when reading the ID Register. The ID Register has the value of `0x41001220`.

Table 2-2 ETM ID Register bit assignments

Bits	Value	Meaning
[31:24]	`0x41`	Implementor = A (ARM Limited).
[23:17]	b000000	Reserved.
[16]	b0	Load pc first. Special handling is not required to reconstruct the data addresses of an LDM with the pc in the register list because ETMv3.2 requires noncontiguous data addresses to be traced. However, special handling is required to determine which transfers correspond to which register.
[15:12]	b0001	ARM core family = ARM9 processor.

Table 2-2 ETM ID Register bit assignments (continued)

Bits	Value	Meaning
[11:8]	b0010	Major ETM architecture version number = 3.
[7:4]	b0010	Minor ETM architecture version number = 2.
[3:0]	b0001	Implementation revision. Value given is for r0p1 release.

2.3.2 Configuration Code Register

The Configuration Code Register, at offset `0x004`, is read only. Figure 2-2 shows the register bit assignments.

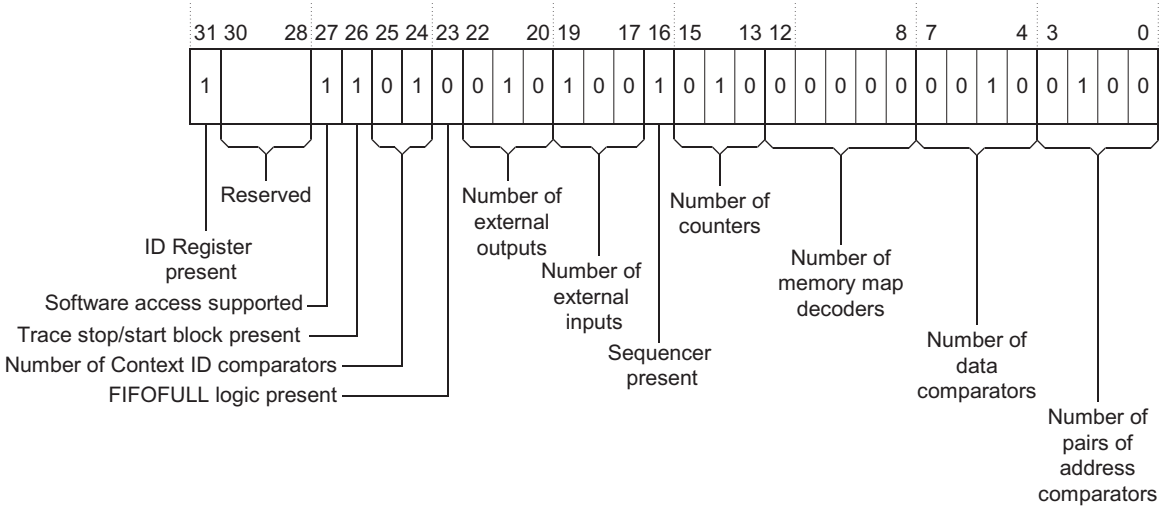


Figure 2-2 Configuration Code Register bit assignments

Table 2-3 shows the value of the fields when reading the Configuration Code Register. The Configuration Code Register has the value `0x8D294024`.

Table 2-3 Configuration Code Register bit assignments

Bits	Value	Meaning
[31]	1	CoreSight ETM9 ID Register present
[30:28]	b000	Reserved. Read-as-zero.
[27]	1	Software access is supported

Table 2-3 Configuration Code Register bit assignments (continued)

Bits	Value	Meaning
[26]	1	Trace start/stop block is present
[25:24]	1	Number of Context ID comparators
[23]	0	**FIFOFULL** logic absent
[22:20]	2	Number of external outputs
[19:17]	4	Number of external inputs
[16]	1	The sequencer is present
[15:13]	2	Number of counters
[12:8]	0	Number of memory map decoders
[7:4]	2	Number of data comparators
[3:0]	4	Number of pairs of address comparators

2.3.3 Configuration Code Extension Register

The Configuration Code Extension Register, at offset `0x1E8`, is read-only. Figure 2-3 shows the register bit assignments.

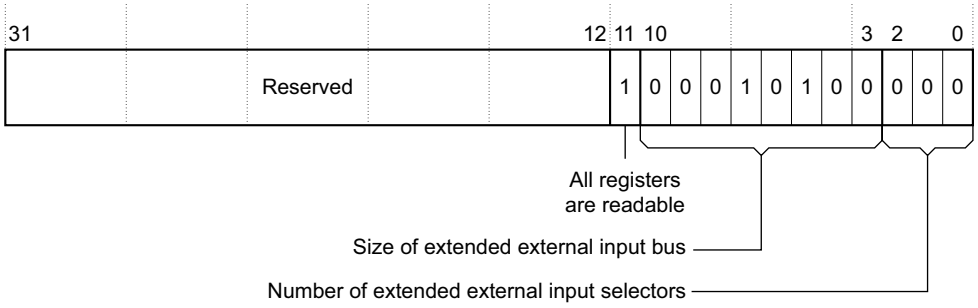


Figure 2-3 Configuration Code Extension Register bit assignments

Table 2-4 shows the value of the fields when reading the Configuration Code Extension Register.

Table 2-4 Configuration Code Extension Register bit assignments

Bits	Value	Meaning
[31:12]	0	Reserved, RAZ
[11]	1	All registers, except some integration test registers, are readable. See Table 2-1 on page 2-3 for details of the access allowed to integration test registers. Registers with names which start with IT are the Integration Test Registers, for example ITATBCTR1.
[10:3]	00	Size of extended external input bus
[2:0]	0	Number of extended external input selectors

2.3.4 Peripheral Identification Registers

The CoreSight ETM9 Peripheral Identification Registers are a set of eight read-only registers, PeripheralID7 to PeripheralID0. These registers are defined in the *ETM Architecture Specification*, which specifies that only bits[7:0] of each register are used. Table 2-5 shows the values of the fields when reading this set of registers. The *ETM Architecture Specification* gives a more detailed description of many of these fields.

Table 2-5 Peripheral Identification Registers, bit assignments

Register	Register offset	Bit	Value	Description
PeripheralID7	0xFDC	[31:8]	-	Unused, read undefined
		[7:0]	0x00	Reserved for future use, RAZ
PeripheralID6	0xFD8	[31:8]	-	Unused, read undefined
		[7:0]	0x00	Reserved for future use, RAZ
PeripheralID5	0xFD4	[31:8]	-	Unused, read undefined
		[7:0]	0x00	Reserved for future use, RAZ
PeripheralID4	0xFD0	[31:8]	-	Unused, read undefined
		[7:4]	0x0	n, where 2^n is number of 4KB blocks used
		[3:0]	0x4	JEP106 continuation code [3:0]
PeripheralID3	0xFEC	[31:8]	-	Unused, read undefined
		[7:4]	0x0	RevAnd (at top level)
		[3:0]	0x0	Customer Modified0x00 indicates from ARM
PeripheralID2	0xFE8	[31:8]	-	Unused, read undefined
		[7:4]	0x1	Revision Number of Peripheral. Value given is for r0p1 release.
		[3]	0x1	Indicates that a JEDEC assigned value is used
		[2:0]	0x3	JEP106 identity code [6:4]
PeripheralID1	0xFE4	[31:8]	-	Unused, read undefined
		[7:4]	0xB	JEP106 identity code [3:0]

Table 2-5 Peripheral Identification Registers, bit assignments (continued)

Register	Register offset	Bit	Value	Description
		[3:0]	0x9	Part Number 1 Upper *Binary Coded Decimal* (BCD) value of Device Number
PeripheralID0	0xFE0	[31:8]	-	Unused, read undefined
		[7:0]	0x10	Part Number 0 Middle and Lower BCD value of Device Number

Note

In Table 2-5 on page 2-9, the *Peripheral Identification Registers* on page 2-9 are listed in order of register name, from most significant (ID7) to least significant (ID0). This does not match the order of the register offsets. Similarly, in Table 2-6 on page 2-11 the *Component Identification Registers* are listed in order of register name, from most significant (ID3) to least significant (ID0).

2.3.5 Component Identification Registers

There are four read-only Component Identification Registers, ComponentID3 to ComponentID0. Although these are implemented as standard 32-bit registers:

- The most significant 24 bits of each register are not used and Read-As-Zero
- The least significant 8 bits of each register together make up the *component ID*.

This concept of a single 32-bit component ID, obtained from the four Component Identification Registers, is shown in Figure 2-4:

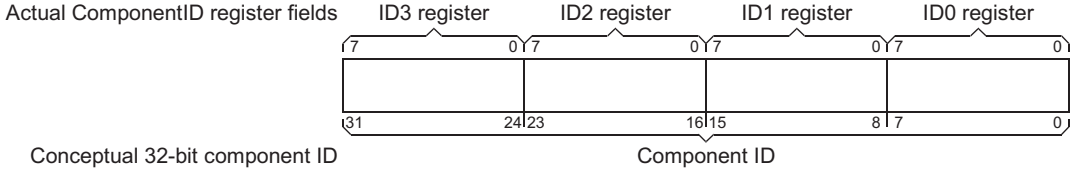


Figure 2-4 Mapping between the Component ID Registers and the component ID value

Table 2-6 lists the values of the fields when reading the CoreSight ETM9 Component Identification Registers. This, again, shows how the valid fields combine to give the component ID. This register structure is as defined in the *CoreSight Architecture Specification*.

Table 2-6 Component Identification Registers, bit assignments

Register	Register offset	Bit	Value	Description
ComponentID3	0xFFC	[31:8]	-	Unused, read undefined
		[7:0]	0xB1	Component identifier, bits[31:24]
ComponentID2	0xFF8	[31:8]	-	Unused, read undefined
		[7:0]	0x05	Component identifier, bits[23:16]
ComponentID1	0xFF4	[31:8]	-	Unused, read undefined
		[7:4]	0x9	Component class (component identifier, bits[15:12])
		[3:0]	0x0	Component identifier, bits[11:8]
ComponentID0	0xFF0	[31:8]	-	Unused, read undefined
		[7:0]	0x0D	Component identifier, bits[7:0]

2.3.6 Integration Test Registers

The following sub-sections describe the Integration Test Registers. If you want to access these registers you must first set bit[0] of the Integration Mode Control Register to 1.

- You can use the write-only Integration Test Registers to set the outputs of some of the ETM signals. Table 2-7 on page 2-12 lists the signals which can be controlled in this way.
- You can use the read-only Integration Test Registers to read the state of some of the ETM input signals. Table 2-8 on page 2-12 lists the signals which can be read in this way.

Table 2-7 Output signals that can be controlled by the Integration Test Registers

Signal	Register	Bit	Register description
AFREADYM[a]	ITATBCTR0	[1]	See *ITATBCTR0 Register (ATB control 0)* on page 2-17
ATBYTESM[1:0][a]	ITATBCTR0	[9:8]	See *ITATBCTR0 Register (ATB control 0)* on page 2-17
ATDATAM[31, 23, 15, 7, 0][a]	ITATBDATA0	[4:0]	See *ITATBDATA0 Register (ATB data 0)* on page 2-16
ATIDM[6:0][a]	ITATBCTR1	[6:0]	See *ITATBCTR1 Register (ATB control 1)* on page 2-17
ATVALIDM[a]	ITATBCTR0	[0]	See *ITATBCTR0 Register (ATB control 0)* on page 2-17
EXTINACK[3:0]	ITMISCOUT	[3:0]	See *ITMISCOUT Register (miscellaneous outputs)* on page 2-13
EXTOUT[1:0]	ITMISCOUT	[9:8]	See *ITMISCOUT Register (miscellaneous outputs)* on page 2-13
TRIGOUT[a]	ITTRIGGERREQ	[0]	See *ITTRIGGERREQ Register (trigger request)* on page 2-15

a. These signals are only available with ETM9CS. Other signals are available with both ETM9CSSingle and ETM9CS

Table 2-8 Input signals that can be read by the Integration Test Registers

Signal	Register	Bit	Register description
AFVALIDM[a]	ITATBCTR2	[1]	See *ITATBCTR2 Register (ATB control 2)* on page 2-16
ATREADYM[a]	ITATBCTR2	[0]	See *ITATBCTR2 Register (ATB control 2)* on page 2-16
DBGACK	ITMISCIN	[4]	See *ITMISCIN Register (miscellaneous inputs)* on page 2-14
EXTIN[3:0]	ITMISCIN	[3:0]	See *ITMISCIN Register (miscellaneous inputs)* on page 2-14
EXTOUTACK[1:0]	ITMISCIN	[9:8]	See *ITMISCIN Register (miscellaneous inputs)* on page 2-14
TRIGOUTACK[a]	ITTRIGGERACK	[0]	See *ITTRIGGERACK Register (trigger acknowledge)* on page 2-14

a. These signals are only available with ETM9CS. Other signals are available with both ETM9CSSingle and ETM9CS

Using the Integration Test Registers

The *CoreSight ETM9 Integration Manual* gives a full description of the use of the Integration Test Registers to check integration. In brief:

- When bit 1 of the Integration Mode Control Register is set, values written to the write-only integration test registers map onto the specified outputs of CoreSight ETM9. For example, writing `0x3` to ITMISCOUT[9:8] causes **EXTOUT[1:0]** to take the value `0x3`.
- When bit 1 of the Integration Mode Control Register is set, values read from the read-only integration test registers correspond to the values of the specified inputs of CoreSight ETM9. For example, if you read ITMISCIN[9:8] you obtain the value of **EXTOUTACK[1:0]**.

ITMISCOUT Register (miscellaneous outputs)

The ITMISCOUT Register, at offset `0xEDC`, is write-only. Figure 2-5 shows the register bit assignments.

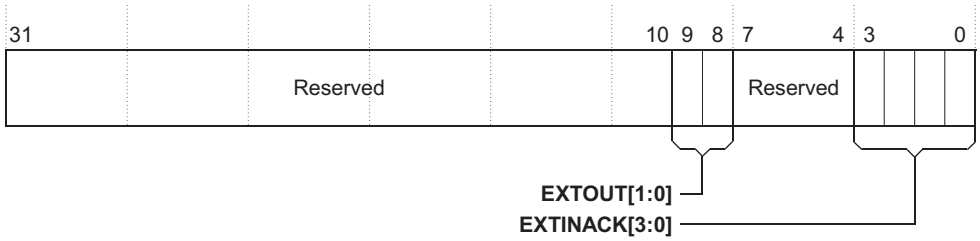


Figure 2-5 ITMISCOUT Register bit assignments

Table 2-9 lists the register bit assignments for the ITMISCOUT Register.

Table 2-9 ITMISCOUT Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Write as zero.
[9:8]	EXTOUT	Drives the **EXTOUT[1:0]** external outputs.
[7:4]	-	Reserved. Write as zero.
[3:0]	EXTINACK	Drives the **EXTINACK[3:0]** external outputs.

ITMISCIN Register (miscellaneous inputs)

The ITMISCIN Register. at offset `0xEE0`, is read-only. Figure 2-6 shows the register bit assignments.

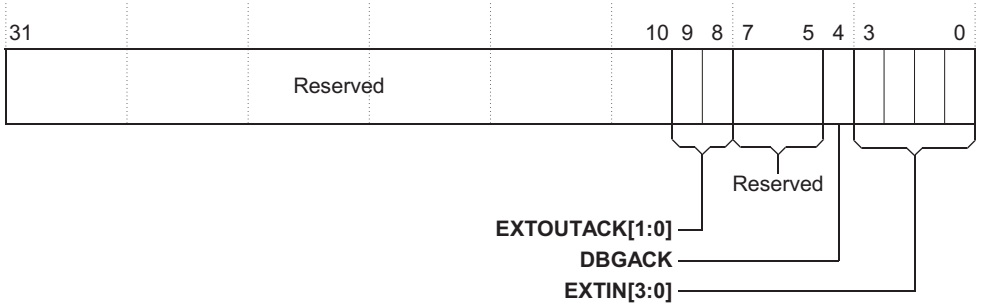


Figure 2-6 ITMISCIN Register bit assignments

Table 2-10 lists the fields when reading the ITMISCIN Register. The value of these fields depend on the signals on the input pins when the register is read.

Table 2-10 ITMISCIN Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Read undefined.
[9:8]	EXTOUTACT	Returns the value of the **EXTOUTACK[1:0]** external inputs.
[7:5]	-	Reserved. Read undefined.
[4]	DBGACK	Returns the value of the **DBGACK** external input.
[3:0]	EXTIN	Returns the value of the **EXTIN[3:0]** external inputs.

ITTRIGGERACK Register (trigger acknowledge)

The ITTRIGGERACK Register, at offset `0xEE4`, is read-only. Figure 2-7 on page 2-15 shows the register bit assignments.

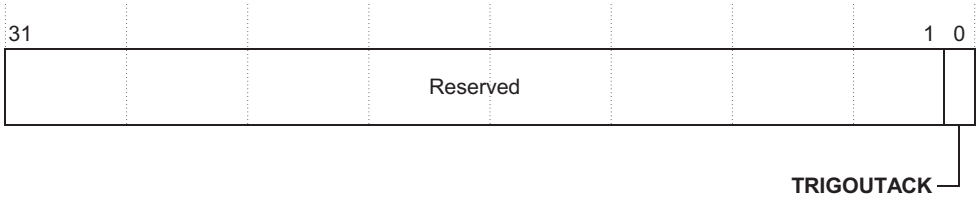


Figure 2-7 ITTRIGGERACK Register bit assignments

Table 2-11 describes the field when reading the ITTRIGGERACK Register. The value of this field depend on the signal on the input pins when the register is read.

Table 2-11 ITTRIGGERACK Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved. Read undefined.
[0]	TRIGOUTACK	Returns the value of the **TRIGOUTACK** external input.

ITTRIGGERREQ Register (trigger request)

The ITTRIGGERREQ Register, at offset `0xEE8`, is write-only. Figure 2-8 shows the register bit assignments.

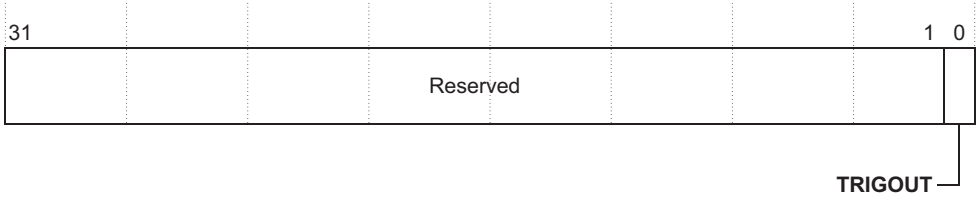


Figure 2-8 ITTRIGGERREQ Register bit assignments

Table 2-12 lists the register bit assignments for the ITTRIGGERREQ Register.

Table 2-12 ITTRIGGERREQ Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved. Write as zero.
[0]	TRIGOUT	Drives the **TRIGOUT** external output.

ITATBDATA0 Register (ATB data 0)

The ITATBDATA0 Register, at offset `0xEEC`, is write-only. Figure 2-9 shows the register bit assignments.

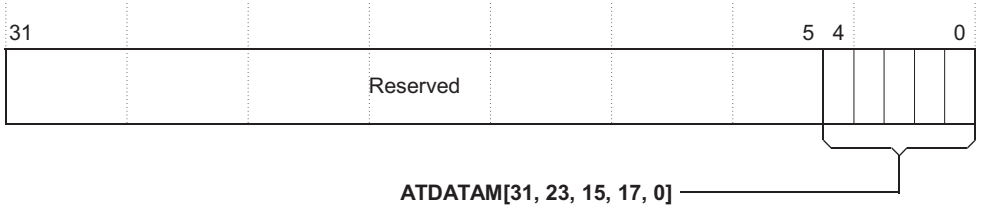


Figure 2-9 ITATBDATA0 Register bit assignments

Table 2-13 lists the register bit assignments for the ITATBDATA0 Register.

Table 2-13 ITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	-	Reserved. Write as zero.
[4:0]	ATDATAM	Drives the **ATDATAM[31, 23, 15, 7, 0]** external outputs.

ITATBCTR2 Register (ATB control 2)

The ITATBCTR2 Register, at offset `0xEF0`, is read-only. Figure 2-10 shows the register bit assignments.

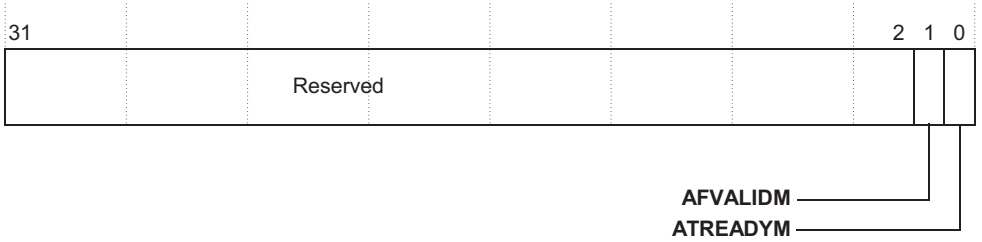


Figure 2-10 ITATBCTR2 Register bit assignments

Table 2-14 lists the fields when reading the ITATBCTR2 Register. The value of these fields depend on the signals on the input pins when the register is read.

Table 2-14 ITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved. Read undefined.
[1]	AFVALIDM	Returns the value of the **AFVALIDM** external input.
[0]	ATREADYM	Returns the value of the **ATREADYM** external input.

ITATBCTR1 Register (ATB control 1)

The ITATBCTR1 Register, at offset `0xEF4`, is write-only. Figure 2-11 shows the register bit assignments.

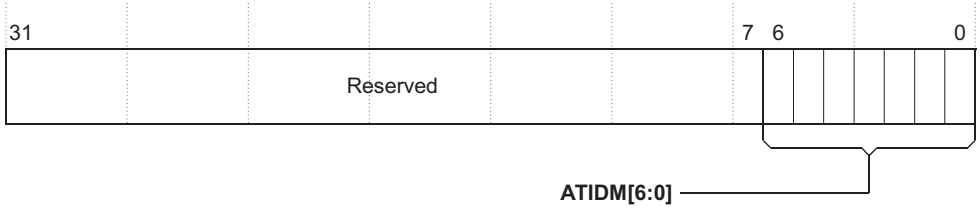


Figure 2-11 ITATBCTR1 Register bit assignments

Table 2-15 lists the register bit assignments for the ITATBCTR1 Register.

Table 2-15 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	-	Reserved. Write as zero.
[6:0]	ATIDM	Drives the **ATIDM[6:0]** external outputs.

ITATBCTR0 Register (ATB control 0)

The ITATBCTR0 Register, at offset `0xEF8`, is write-only. Figure 2-12 on page 2-18 shows the register bit assignments.

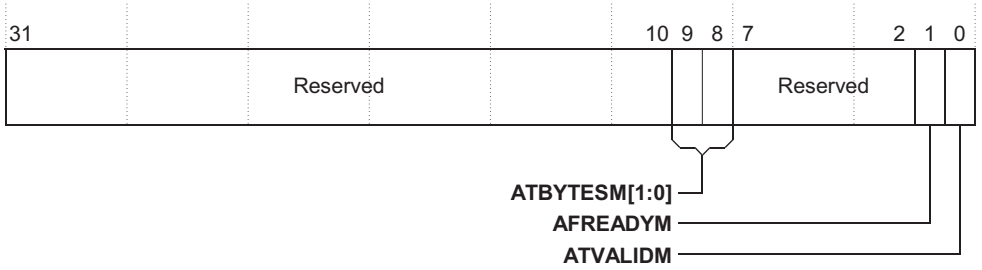


Figure 2-12 ITATBCTR0 Register bit assignments

Table 2-16 lists the register bit assignments for the ITATBCTR0 Register.

Table 2-16 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Write as zero.
[9:8]	ATBYTESM	Drives the **ATBYTESM[1:0]** external outputs.
[7:2]	-	Reserved. Write as zero.
[1]	AFREADYM	Drives the **AFREADYM** external output.
[0]	ATVALIDM	Drives the **ATVALIDM** external output.

2.4 Precise TraceEnable events

The *ETM Architecture Specification* states that **TraceEnable** is Imprecise under certain conditions, with some implementation-defined exceptions. Selection of the following resources by the enabling event does not cause **TraceEnable** to be Imprecise, provided that the resources are themselves precise:

- single address comparators
- address range comparators.

2.5 Context ID tracing

CoreSight ETM9 detects the MCR instruction that changes the context ID and traces the appropriate number of bytes as a context ID packet instead of a normal data packet. As a result, if context ID tracing is enabled, an MCR instruction that changes the context ID does not have its data traced separately.

2.6 ETM9CSSingle clocks

This section describes ETM9CSSingle clocks and clock enables:

- *ETM9CSSingle clock signals*
- *ETM9CSSingle clock enable signals* on page 2-22.

2.6.1 ETM9CSSingle clock signals

ETM9CSSingle contains the following clocks:

CLK This is the main clock for the ETM9CSSingle block and must be the same clock as that wired to the **CLK** input of the ARM9 processor. It clocks all registers in the core-facing logic of ETM9CSSingle. It can be asynchronous to **PCLK and ATCLK**.

PCLK This is the system APB interface clock in ETM9CSSingle. It can be asynchronous to **CLK and ATCLK**.

ATCLK This is the *AMBA Trace Bus* (ATB) interface clock. It can be asynchronous to **CLK and PCLK**.

Note

The effective **PCLK** frequency must be the same or slower than the effective **ATCLK** frequency:

- the effective **PCLK** frequency is obtained from **PCLK** and **PCLKEN**
- the effective **ATCLK** frequency is obtained from **ATCLK** and **ATCLKEN**.

TRACECLK Trace clock generated in ETMTRACEPORT for use off-chip

TRACECLK is generated from **ATCLK**. It is exported off-chip for the trace capture device to clock in the trace data. It is not used anywhere else in ETM9CSSingle. During layout, the rising edge of **TRACECLK** must be delayed to the middle of the **ATCLK** cycle to provide maximum set-up time for trace capture devices to sample the trace data off TRACECLK. See *CoreSight ETM9 Configuration and Sign-off Guide* for details.

2.6.2 ETM9CSSingle clock enable signals

ETM9CSSingle has the following clock enable signals:

PCLKEN This is the system APB interface clock enable. It can be used to slow down **PCLK**.

ATCLKEN This is the ATB interface clock enable. It can be used to slow down **ATCLK**.

Note

ETM9CSSingle has a **CLKEN** input. You must not use **CLKEN** to slow down **CLK**. You must connect **CLKEN** to the appropriate processor output. See *CoreSight ETM9 Integration Manual* for details.

2.7 ETM9CSSingle resets

ETM9CSSingle contains the following resets:

nPORESET This signal is the main power-on reset. It resets registers in the CLK domain of ETM9CSSingle. It is active LOW.

PRESETn This signal is the system APB reset. It resets registers in the PCLK domain of ETM9CSSingle except the TAP controller registers in ETMJTAGPORT. It is active LOW.

ATRESETn This signal is the ATB interface reset. It resets registers in the ATCLK domain of ETM9CSSingle. It is active LOW.

nTRST This signal is the JTAG reset for ETM9CSSingle. It resets registers in the TAP controller of ETMJTAGPORT. It is active LOW.

2.8 ETM9CS clocks

This section describes ETM9CS clocks and clock enables:

- *ETM9CS clock signals*
- *ETM9CS clock enable signals*.

2.8.1 ETM9CS clock signals

ETM9CS contains the following clocks:

CLK This is the main clock for the ETM9CS block and must be the same clock as that wired to the **CLK** input of the ARM9 processor. It can be asynchronous to PCLKDBG and **ATCLK**.

PCLKDBG This is the Debug APB interface clock in ETM9CS. It can be asynchronous to **CLK** and **ATCLK**.

ATCLK This is the ATB interface clock. It can be asynchronous to **CLK** and **PCLKDBG**.

2.8.2 ETM9CS clock enable signals

ETM9CSSingle has the following clock enable signals:

ATCLKEN This is the ATB clock enable. It can be used to slow down **ATCLK**.

PCLKENDBG This is the Debug APB interface clock enable. It can be used to slow down PCLKDBG.

Note

ETM9CS has a **CLKEN** input. You must not use **CLKEN** to slow down **CLK**. You must connect **CLKEN** to the appropriate processor output. See *CoreSight ETM9 Integration Manual* for details.

2.9 ETM9CS resets

ETM9CS contains the following resets:

nPORESET This signal is the main power-on reset. It resets all the registers in the CLK domain of ETM9CS. It is active LOW.

PRESETDBGn This signal is the Debug APB interface reset. It resets all the registers in the PCLKDBG domain of ETM9CS. It is active LOW.

ATRESETn This signal is the ATB interface reset. It resets all the registers in the ATCLK domain. It is active LOW.

2.10 PORTMODE, PORTSIZE, and MAXPORTSIZE

PORTMODE and **PORTSIZE** can be set by writing to the ETM Control Register, `0x00`. See *ETM Architecture Specification* for more details.

2.10.1 PORTMODE and PORTSIZE in ETM9CS

In a CoreSight system **PORTMODE** and **PORTSIZE** are not used.

2.10.2 PORTMODE and PORTSIZE in ETM9CSSingle

In ETM9CSSingle, **PORTMODE** and **PORTSIZE** are used to control ETMTRACEPORT functionality.

ETM9CSSingle supports the following port modes and given combinations of **PORTMODE** and **PORTSIZE**:

Dynamic mode, PORTMODE[2:0] = b000

ETMTRACEPORT functionality is disabled. **PORTSIZE** must be set to 32 bits.

Asynchronous mode, PORTMODE[2:0] = b100

ETMTRACEPORT functionality is enabled.

Table 2-17 lists the supported **PORTMODE** and **PORTSIZE** combinations.

Table 2-17 Supported PORTMODE and PORTSIZE combinations

PORTMODE[2:0] value	PORTSIZE[3:0] value	Mode
b000	b0100 (32-bit port)	Dynamic
b100	b0100 (32-bit port)	Asynchronous
b100	b0010 (16-bit port)	Asynchronous
b100	b0001 (8-bit port)	Asynchronous
b100	b0000 (4-bit port)	Asynchronous
b100	b1001 (2-bit port)	Asynchronous

2.10.3 MAXPORTSIZE in ETM9CS and ETM9CSSingle

MAXPORTSIZE can be programmed to any architecturally defined value.

MAXPORTSIZE appears in the System Configuration Register. If CoreSight ETM9 is programmed with a port size greater than that specified by MAXPORTSIZE, the port size supported bit, bit 10 of the ETM System Configuration Register, is LOW. The debugger must check that this bit is HIGH before a debugging session to ensure that a valid port size has been programmed.

2.11 Data instructions in Java state

Table 2-18 gives the amount of trace expected for each data bytecode, which is implementation defined.

Table 2-18 Bytecodes and amounts of trace

Bytecodes	Amount of trace
baload, bastore	Byte
caload, castore, saload, sastore	Halfword
iload, aload, fload, iaload, aaload, faload, istore, astore, fstore, iastore, aastore, fastore, net, getfield_a, putfield_a, ldc_a, ldc_w_a, getstatic_a (SP=1), putstatic_a (SP=1),	Word
dload, lload, daload, laload, dstore, lstore, dastore, lastoregetfield2_a, putfield2_a, ldc2_w_a, getstatic_a (SP=0), putstatic_a (SP=0)	Two words
all other bytecodes	no data

Table 2-18 assumes the bytecode is implemented in hardware. The Jazelle specification allows any bytecode to be implemented in software, and normal ARM state data trace is output instead in this case. This applies to aastore in current implementations.

2.12 Restrictions and limitations

See the TRM for your ARM9 processor for any restrictions or limitations on what can be traced.

Chapter 3
Programmer's Model

This chapter describes the mechanisms for programming the registers used to set up the trace and triggering facilities of CoreSight ETM9. It contains the following sections:

3.1 About the programmer's model

When programming the ETM registers you must enable all the changes at the same time. For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up.

You can use the ETM programming bit in the ETM Control Register to disable all trace operations during programming. To do this follow the procedure shown in Figure 3-1 on page 3-3.

The individual registers are not described here. For more information, see the *ETM Architecture Specification*.

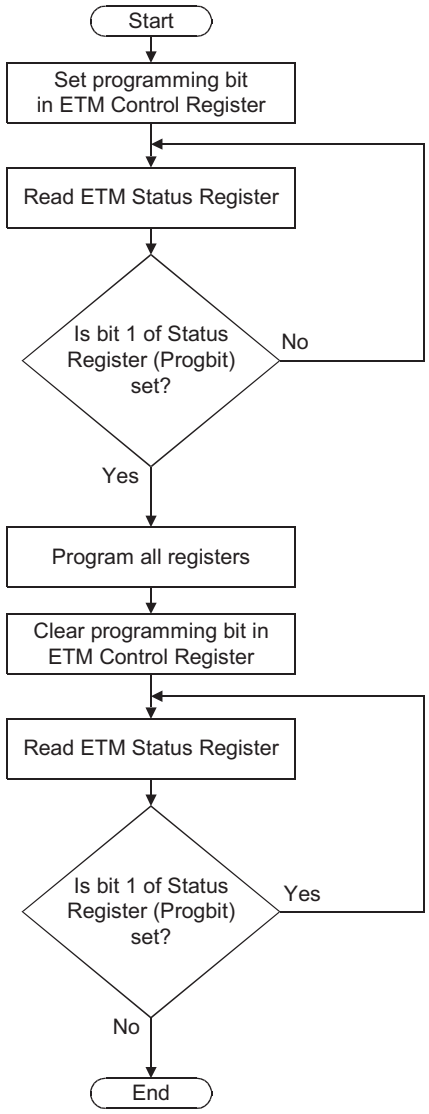


Figure 3-1 Programming ETM registers

The processor does not need to be in the debug state while the ETM registers are being programmed.

3.2 Programming and reading ETM registers

There are two methods of access to the CoreSight ETM9 registers:

- *Software access using APB*
- *JTAG access through ETMJTAGPORT (JTAG-APB bridge) in ETM9CSSingle*.

3.2.1 Software access using APB

APB provides a direct method of programming stand-alone CoreSight ETM9 and CoreSight ETM9 in a CoreSight system.

APB is a simple low cost interface used to provide access to the programmable control registers of peripheral devices. It has the following features:

- Unpipelined protocol, that is, a second transfer cannot start before the first transfer completes.
- Every transfer takes at least two cycles.

For more information on APB transfers, see the *AMBA 3 APB Protocol Specification*.

3.2.2 JTAG access through ETMJTAGPORT (JTAG-APB bridge) in ETM9CSSingle

JTAG programming is through the ETMJTAGPORT described in *ETMJTAGPORT* on page 4-3. The interface is an extension of the ARM TAP controller, and is assigned scan chain number 6. The scan chain consists of a 40-bit shift register comprising:

- A 32-bit data field
- A 7-bit address field
- A read/write bit.

Some registers in CoreSight ETM9 are outside the address range of the JTAG interface. These include the CoreSight Management Registers and the CoreSight ETM9 Integration Registers, which are accessible through APB. The CoreSight Management Registers are only used in a CoreSight system. The general arrangement of the ETM JTAG registers is shown in Figure 3-2 on page 3-5.

When writing a register, the data to be written is scanned into the 32-bit data field, the address of the register into the 7-bit address field and a 1 into the read/write bit. A register is read by scanning its address into the address field and a 0 into the read/write bit. The 32-bit data field is ignored.

The read data is transferred into the lower 32-bit data field when the TAP state machine transitions through CAPTURE-DR.

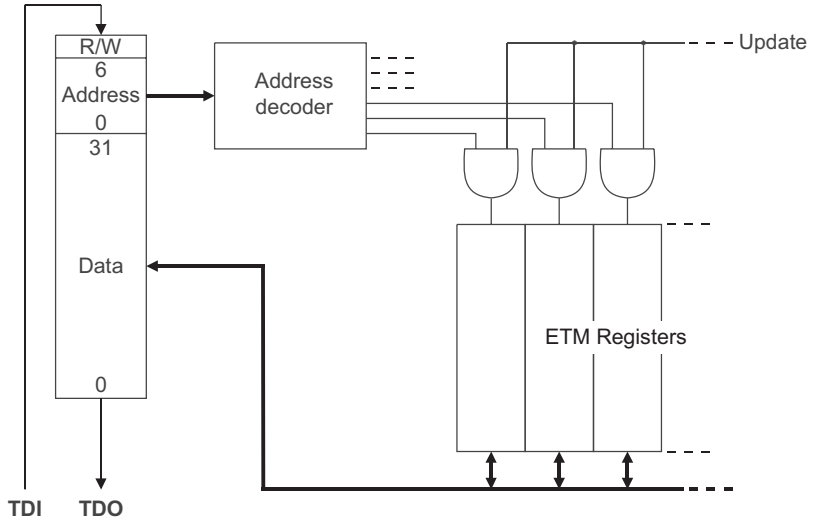


Figure 3-2 ETM JTAG structure

Note

A read or write takes place when the TAP controller enters the UPDATE-DR state.

Chapter 4
Blocks for Stand-alone CoreSight ETM9

This chapter describes the additional blocks for a stand-alone CoreSight ETM9. It contains the following section:

4.1 About additional blocks for stand-alone CoreSight ETM9

Two additional components are available for stand-alone CoreSight ETM9:

ETMJTAGPORT

A JTAG to APB bridge that supports the JTAG programming model for access to the ETM.

ETMTRACEPORT

A Trace Port Interface Unit for tracing a single trace source that supports multiple trace port sizes.

Figure 4-1 shows CoreSight ETM9 with an ETMJTAGPORT and ETMTRACEPORT.

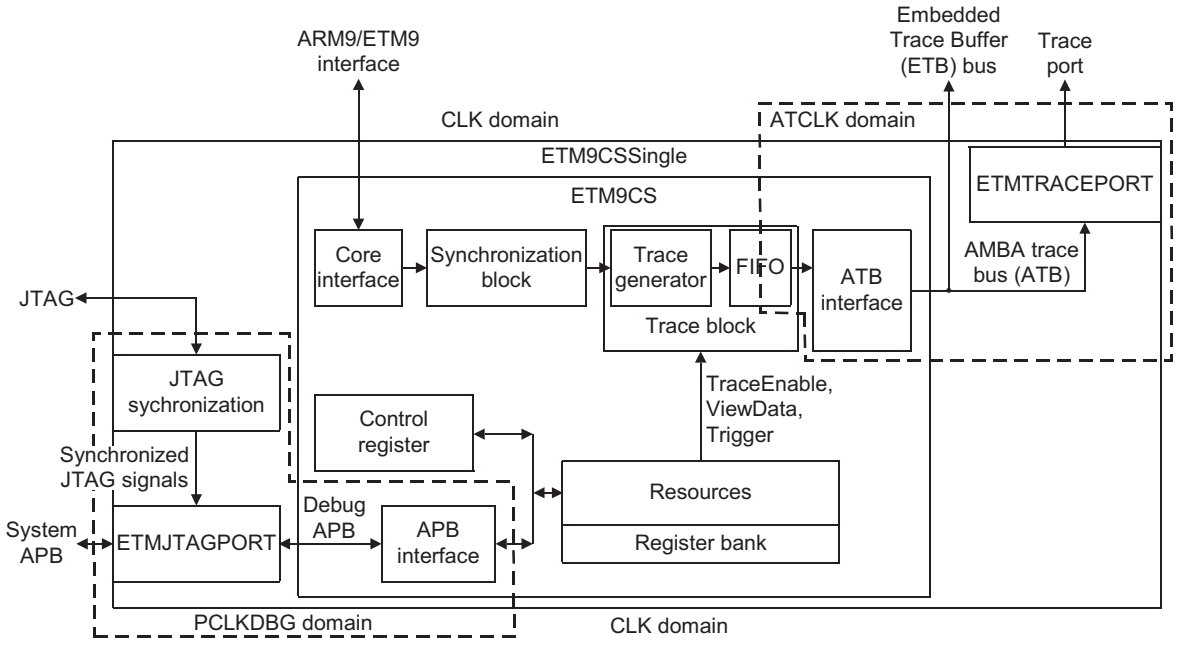


Figure 4-1 CoreSight ETM9 with ETMJTAGPORT and ETMTRACEPORT

4.1.1 ETMJTAGPORT

ETMJTAGPORT converts synchronized JTAG programming commands to APB accesses. ETMJTAGPORT arbitrates between synchronized JTAG and system APB. If a JTAG access is in progress, the system **PREADY** is held LOW for the duration of the JTAG access. This is because APB access can be stalled but JTAG access cannot be stalled.

ETMJTAGPORT contains a TAP controller that must be placed in parallel with the core TAP controller when ETMJTAGPORT is connected to a CoreSight ETM9, so that both TAP controllers are in step with one another. Figure 4-2 shows an example of TAP controllers in step and an optional ETB.

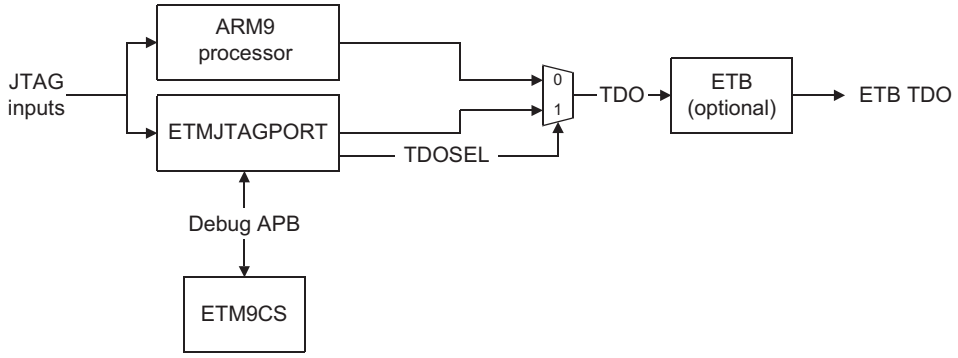


Figure 4-2 TAP controllers in step

4.1.2 JTAG synchronization

The registers in ETMJTAGPORT are clocked by the effective **PCLK** (**PCLK** + **PCLKEN**). Therefore its JTAG interface inputs must be synchronized to the effective **PCLK**. This is done in the JTAG synchronization module, `ETMJTAGSync`, of ETM9CSSingle as shown in Figure 4-3 on page 4-4.

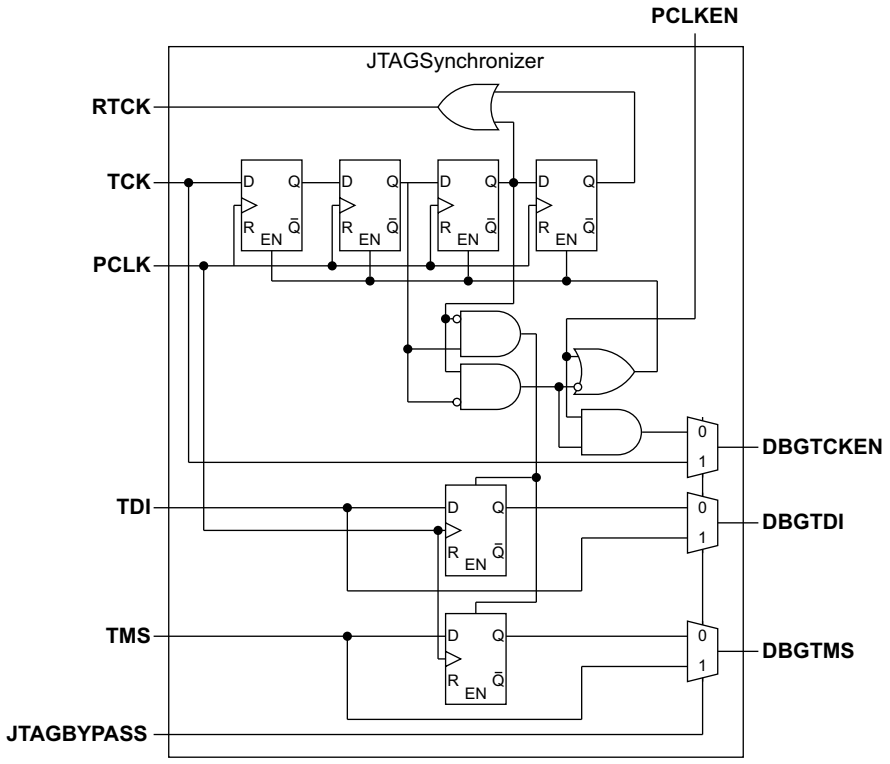


Figure 4-3 JTAG synchronization block

Note

All registers in Figure 4-3 are reset by **nTRST**.

To set **JTAGSBYPASS** to 0:

- The effective **PCLK** must be the same or slower than the effective **ATCLK**.

To set **JTAGSBYPASS** to 1:

- **PCLK** must be identical to **CLK**.
- The effective **PCLK** must be the same or slower than the effective **ATCLK**.
- Either:
  - **PCLKEN** must be set to 1
  - **TCKEN** from the external JTAG synchronizer must be a pure subset of **PCLKEN**.

Note

See the *CoreSight ETM9 Integration Manual* for details of JTAG scan chain connections.

4.1.3 Programmer's model for stand-alone systems

This section describes:

- *The Scan Chain 6 Register*.

The Scan Chain 6 Register

To write an ETM register, bit[39] must be set and the address field, bits[38:32], must contain a valid ETM register address. The data in bits[31:0] are then written into the corresponding ETM register when the TAP controller goes through the UpdateDR state.

To read an ETM register, bit[39] must be clear and the address field must contain a valid ETM register address. The register data is fetched from the ETM when the TAP controller passes through the UpdateDR state, and loaded into the Scan Chain 6 Register when the TAP controller passes through the CaptureDR state. This data can then be read out through **TDO** by putting the TAP controller into the ShiftDR state.

Table 4-1 describes the format of the 40-bit Scan Chain 6 Register

Table 4-1 Scan Chain 6 Register bit assignments

Bits	Description
[39]	Write / !Read Access
[38:32]	Register Address
[31:0]	Data

4.1.4 ETMTRACEPORT

ETMTRACEPORT takes the ETM trace output stream and converts it to a form suitable for taking off-chip. It also creates and exports a clock, **TRACECLK**, used by the off-chip capture device to sample the converted trace data.

TRACECLK and its associated data are generated from **ATCLK**. CoreSight ETM9CSSingle supports asynchronous CLK, PCLK, and ATCLK clock domains. Therefore, it is not necessary for the trace port clock frequency to be an integer division of the main clock.

ETMTRACEPORT sets its trace port configuration according to the values of **PORTSIZE[3:0]** output from CoreSight ETM9. ETMTRACEPORT only supports trace port sizes of 2, 4, 8, 16 and 32 bits, because ETMTRACEPORT only receives 32-bit data packets from the ETM, and all these port sizes wholly divide into 32.

See *CoreSight ETM9 Configuration and Sign-off Guide* for implementation details about delaying **TRACECLK**.

Appendix A
Signals Lists

This appendix describes the signals used in the CoreSight ETM9. It contains the following sections:

A.1 ETM9CSSingle Signals

Table A-1 lists the ETM9CSSingle signals in alphabetical order. Clock domains, where specified, give the clock on which input signals must be generated and output signals sampled. See the *CoreSight ETM9 Integration Manual* for information about signals and connectivity.

Table A-1 ETM9CSSingle signals

Signal name	Input/Output	Description	Clock domains
ASICCTL[7:0]	Output	Contents of the ASIC control register.	**CLK**
ATCLK	Input	ATB interface clock.	-
ATCLKEN	Input	Enable signal for **ATCLK.**	**ATCLK**
ATRESETn	Input	Internal trace bus reset. Resets all registers in the ATCLK domain. Active LOW.	Internally synchronized
BIGEND	Input	When HIGH indicates that the processor is operating in big-endian mode.	**CLK**
CHSD[1:0]	Input	Coprocessor handshake decode signals.	**CLK**
CHSE[1:0]	Input	Coprocessor handshake execute signals.	**CLK**
CLK	Input	This is the main clock for the ETM9CSSingle, and must be the same as the processor clock.	-
CLKEN	Input	Indicates when the ARM9 processor stalls waiting for memory accesses to complete. This signal is connected to an output from the ARM9 processor.	**CLK**
CORESELECT[2:0]	Output	Where an ETM is shared between multiple cores, this signal specifies which core to trace.	-
DA[31:0]	Input	Data address.	**CLK**
DABORT	Input	This signal indicates that the data transfer requested by the processor has been aborted.	**CLK**
DBGACK	Input	Indicates that the core is in debug state. This is connected to the core general purpose **DBGACK** output, so that it can be used to determine when **ETMDBGRQ** can be deasserted. It is also used for other purposes in the ETM, and care must be taken to ensure the timing of this signal is appropriate because it does not come through the main core/ETM interface.	**CLK**

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
DBGEN	Input	When HIGH indicates that invasive debug is enabled.	-
DD[31:0]	Input	Write data value.	**CLK**
DDIN[31:0]	Input	Read data value.	**CLK**
DMAS[1:0]	Input	Data memory access size.	**CLK**
DnMREQ	Input	If this signal is LOW at the end of a cycle then the processor requires a data memory access in the following cycle.	**CLK**
DnRW	Input	If this signal is LOW at the end of a cycle then any data memory access in the following cycle is a read. If this signal is HIGH then it is a write.	**CLK**
DSEQ	Input	If this signal is HIGH at the end of the cycle then any data memory access in the following cycle is sequential from the last data memory access.	**CLK**
ETBTRACEDATA[31:0]	Output	Provides trace data for ETB capture.	**ATCLK**
ETBTRACEVALID	Output	Indicates that trace data is valid.	**ATCLK**
ETBTRIGGER	Output	Indicates trigger condition.	**ATCLK**
ETMDBGRQ	Output	Request from the CoreSight ETM9 for the core to enter debug state. This must be ORed with any ASIC-level **DBGRQ** signals before being connected to the core **EDBGRQ** input.	**CLK**
ETMEN	Output	When HIGH, ETMTRACEPORT is enabled. When LOW, logic related to the to the ETMTRACEPORT can be clock-gated.	**CLK**
ETMPWRUP	Output	When HIGH, indicates that CoreSight ETM9 is in use. When LOW: • external logic supporting CoreSight ETM9 can be clock-gated to conservepower. • the ARM9 processor disables the CoreSight ETM9 interface • logic within CoreSight ETM9 is clock-gated to conserve power.	**CLK**

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
EXTIN[3:0]	Input	External input resources.	See footnote [a]
EXTINACK[3:0]	Output	Acknowledge signal for the **EXTIN[3:0]** bus. When **EXTSBYPASS** is HIGH, **EXTINACK[3:0]** are not valid and can be ignored.	-
EXTOUT[1:0]	Output	External outputs.	See footnote [a]
EXTOUTACK[1:0]	Input	Acknowledge signals for the **EXTOUT[1:0]** bus. When **EXTSBYPASS** is HIGH, these signals must be tied LOW.	-
EXTSBYPASS	Input	**EXTSBYPASS** is a single bit input to the ETM that is used to bypass the synchronization of the external inputs **EXTIN[3:0]** and external outputs **EXTOUT[1:0]**. When EXTSBYPASS is LOW: • a HIGH output on any bit of **EXTOUT[1:0]** is held until the corresponding bit of **EXTOUTACK[1:0]** is asserted HIGH. • a HIGH on any bit of **EXTIN[3:0]** must remain held until the ETM asserts the corresponding bit of **EXTINACK[3:0]** HIGH. When **EXTSBYPASS** is HIGH, **EXTOUT[1:0]** outputs are valid on the rising edge of **CLK** and **EXTIN[3:0]** inputs must be valid on the rising edge of **CLK**.	-
FIFOPEEK[8:0]	Output	For validation purposes only. Indicates when various events occur before being written to the FIFO.	**CLK**
HIVECS	Input	When this signal is LOW the exception vectors start at address 0x0000 0000. When HIGH the exception vectors start at address 0xFFFF 0000.	**CLK**
IA[31:0]	Input	Instruction address.	**CLK**
ID15To11[15:11]	Input	Instruction data field.	**CLK**
ID31To25[31:25]	Input	Instruction data field.	**CLK**

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
IJBIT	Input	When this signal is HIGH it indicates the ARM processor is in JAVA state. This signal is valid with the address.	**CLK**
JTAGSBYPASS	Input	**JTAGSBYPASS** is a single bit input to the ETM that is used to bypass the JTAG synchronization.	-
InMREQ	Input	If this signal is LOW at the end of the cycle then the processor requires an instruction memory access in the following cycle.	**CLK**
INSTREXEC	Input	This signal indicates the instruction in the Execute stage of the pipeline follower of the ETM has been executed.	**CLK**
INSTRVALID	Input	This signal indicates the instruction in the Execute stage is valid and has not been flushed.	**CLK**
ISEQ	Input	If this signal is HIGH at the end of the cycle then any instruction memory access in the following cycle is sequential from the last instruction memory access.	**CLK**
ITBIT	Input	When HIGH this signal indicates the ARM processor is in Thumb state. When this signal is LOW the processor is in ARM state. This signal is valid with the address.	**CLK**
LATECANCEL	Input	Cancel coprocessor instruction. If this signal is HIGH during the first memory cycle of a coprocessor instruction, then the coprocessor must cancel the instruction without changing any internal state. This signal is only asserted in cycles where the previous instruction accessed memory and a Data Abort occurred.	**CLK**
MAXCORES[2:0]	Input	Where a CoreSight ETM9 is shared between multiple cores, this signal specifies the number of cores the ETM can trace. It must be tied to the number of cores sharing the ETM minus 1.	-
MAXEXTIN[2:0]	Input	External inputs supported by the ASIC (maximum 4). This appears in the configuration code register.	-
MAXEXTOUT[1:0]	Input	External outputs supported by the ASIC (maximum 2). This appears in the configuration code register.	-

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
MAXPORTSIZE[3:0]	Input	Maximum port size supported. See *MAXPORTSIZE in ETM9CS and ETM9CSSingle* on page 2-27.	-
NIDEN	Input	Non-invasive debug enable.	Internally synchronized
nPORESET	Input	Main power on reset. Resets all registers in the CLK domain.	Internally synchronized
nTRST	Input	JTAG interface reset. Active LOW.	-
PADDR[11:2]	Input	System APB Address Bus.	**PCLK**
PASS	Input	This signal indicates that the coprocessor instruction in the Execute stage of the pipeline follower of the ETM has been executed.	**CLK**
PCLK	Input	System APB interface clock	-
PCLKEN	Input	System APB clock enable.	**PCLK**
PENABLE	Input	The system APB interface is enabled for a transfer.	**PCLK**
PORTMODE[2:0]	Output	Currently requested port mode.	**CLK**
PORTSIZE[3:0]	Output	Currently requested port size.	**CLK**
PRDATA[31:0]	Output	System APB read data.	**PCLK**
PREADY	Output	Used to extend transfers, and to arbitrate between JTAG and APB accesses.	**PCLK**
PRESETn	Input	System APB interface reset. Active LOW.	Internally synchronized
PROCID[31:0]	Input	Process ID.	**CLK**
PROCIDWR	Input	This signal must be asserted whenever the PROCID bus changes. This causes the ETM to output the new context ID at the next available opportunity.	**CLK**
PSEL	Input	System APB select signal.	**PCLK**
PSLVERR	Output	System APB error signal	**PCLK**
PWDATA[31:0]	Input	System APB write data.	**PCLK**

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
PWRITE	Input	System APB transfer direction (!Read/Write).	**PCLK**
RANGEOUT0	Input	This signal indicates that the corresponding watchpoint unit has matched the conditions currently present on the address, control and data buses. This signal is independent of the state of the enable control bit of the watchpoint unit.	**CLK**
RANGEOUT1	Input	This signal indicates that the corresponding watchpoint unit has matched the conditions currently present on the address, control and data buses. This signal is independent of the state of the enable control bit of the watchpoint unit.	**CLK**
RSTBYPASS	Input	Reset synchronization bypass.	-
RTCK	Output	JTAG interface return clock	-
SE	Input	Scan enable.	-
TCK	Input	JTAG interface clock .	-
TDI	Input	Test data in. Must be connected to **TDI** input to the core.	See footnote [b]
TDO	Output	Test data out. Must be multiplexed externally with TDO from the core, controlled by **TDOSEL**.	**PCLK**
TDOSEL	Output	Selects between core and ETM **TDO**.	**PCLK**
TMS	Input	Test mode select. Must be connected to the **TMS** input to the core.	See footnote [b]
TRACECLK	Output	Exported clock for **TRACEDATA[31:0]** and **TRACECTL**. Data is valid on both edges of this clock for maximum integrity.	-
TRACECTL	Output	Used by trace capture devices. This signal is valid for the same time as **TRACEDATA**.Trigger = **TRACECTL** & !**TRACEDATA[0]** TraceDisabled = **TRACECTL** & **TRACEDATA[0]**.	**TRACECLK**

Table A-1 ETM9CSSingle signals (continued)

Signal name	Input/Output	Description	Clock domains
TRACEDATA[31:0]	Output	32-bit trace port. Only data on this bus must be captured.	**TRACECLK**
ZIFIRST	Input	When the ARM9 processor is in Java state this signal is asserted on the first ARM instruction to be traced. (No more than two instructions are ever traced for a bytecode.) If only one ARM instruction is traced in Java state, both **ZIFIRST** and **ZILAST** are asserted.	**CLK**
ZILAST	Input	When the ARM9 processor is in Java state this signal is asserted on the last ARM instruction to be traced. (No more than two instructions are ever traced for a bytecode.) If only one ARM instruction is traced in Java state, both **ZIFIRST** and **ZILAST** are asserted	**CLK**

a. If **EXTSBYPASS** is 1, then **EXTIN[3:0]** must be generated on CLK and **EXTOUT[1:0]** must be sampled on **CLK**. If **EXTSBYPASS** is 0, then there is no such restriction. See *CoreSight ETM9 Integration Manual* for more details.
b. If **JTAGSBYPASS** is 1, then the JTAG interface inputs must be synchronous to PCLK. If **JTAGSBYPASS** is 0, then the JTAG inputs are synchronized internally.

A.2 ETM9CS Signals

Table A-2 ETM9CS signals in alphabetical order. Clock domains, where specified, give the clock on which input signals must be generated and output signals sampled. See the *CoreSight ETM9 Integration Manual* for information about signals and connectivity.

Table A-2 ETM9CS signals

Signal name	Input/output	Description	Clock domains
AFREADYM	Output	ATB interface FIFO flush finished.	**ATCLK**
AFVALIDM	Input	ATB interface FIFO flush request.	**ATCLK**
ASICCTL[7:0]	Output	Contents of the ASIC control register.	**CLK**
ATBYTESM[1:0]	Output	Size of **ATDATAM**.	**ATCLK**
ATCLK	Input	ATB interface clock.	-
ATCLKEN	Input	Enable signal for **ATCLK**.	**ATCLK**
ATDATAM[31:0]	Output	ATB interface data.	**ATCLK**
ATIDM[6:0]	Output	ATB interface trace source ID.	**ATCLK**
ATREADYM	Input	**ATDATA** can be accepted.	**ATCLK**
ATRESETn	Input	ATB interface reset.	Internally synchronized
ATVALIDM	Output	ATB interface data valid.	**ATCLK**
BIGEND	Input	When HIGH indicates that the processor is operating in big-endian mode.	**CLK**
CHSD[1:0]	Input	Coprocessor handshake decode signals.	**CLK**
CHSE[1:0]	Input	Coprocessor handshake execute signals.	**CLK**
CLK	Input	This is the main clock for the ETM9CS, and must be the same as the processor clock.	-
CLKEN	Input	Indicates when the ARM9 processor stalls waiting for memory accesses to complete. This signal is connected to an output from the ARM9 processor.	**CLK**
CORESELECT[2:0]	Output	Where an ETM is shared between multiple cores, this signal specifies which core to trace.	-
DA[31:0]	Input	Data address.	**CLK**

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
DABORT	Input	This signal indicates that the data transfer requested by the processor has been aborted.	**CLK**
DBGACK	Input	Indicates that the core is in debug state. This is connected to the core general purpose **DBGACK** output, so that it can be used to determine when **ETMDBGRQ** can be deasserted. It is also used for other purposes in the ETM, and care must be taken to ensure the timing of this signal is appropriate because it does not come through the main core/ETM interface.	**CLK**
DBGEN	Input	When HIGH indicates that invasive debug is enabled.	-
DD[31:0]	Input	Write data value	**CLK**
DDIN[31:0]	Input	Read data value.	**CLK**
DMAS[1:0]	Input	Data memory access size.	**CLK**
DnMREQ	Input	If this signal is LOW at the end of a cycle then the processor requires a data memory access in the following cycle.	**CLK**
DnRW	Input	If this signal is LOW at the end of a cycle then any data memory access in the following cycle is a read. If this signal is HIGH then it is a write.	**CLK**
DSEQ	Input	If this signal is HIGH at the end of the cycle then any data memory access in the following cycle is sequential from the last data memory access.	**CLK**
ETMDBGRQ	Output	Request from the CoreSight ETM9 for the core to enter debug state. This must be ORed with any ASIC-level **DBGRQ** signals before being connected to the core **EDBGRQ** input.	**CLK**
ETMEN	Output	When HIGH, the ATB interface is enabled. When LOW, logic related to the to the ATB interface can be clock-gated.	**CLK**

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
ETMPWRUP	Output	When HIGH, indicates that CoreSight ETM9 is in use. When LOW: • external logic supporting CoreSight ETM9 can be clock-gated to conservepower. • the ARM9 processor disables the CoreSight ETM9 interface • logic within CoreSight ETM9 is clock-gated to conserve power.	**CLK**
ETPSUP	Input	Indicates a trace port is connected.	-
EXTIN[3:0]	Input	External input resources.	See footnote [a]
EXTINACK[3:0]	Output	Acknowledge signal for the **EXTIN[3:0]** bus. When **EXTSBYPASS** is HIGH, **EXTINACK[3:0]** are not valid and can be ignored.	-
EXTOUT[1:0]	Output	External outputs.	See footnote [a]
EXTOUTACK[1:0]	Input	Acknowledge signals for the **EXTOUT[1:0]** bus. When **EXTSBYPASS** is HIGH, these signals must be tied LOW.	-
EXTSBYPASS	Input	EXTSBYPASS is a single bit input to the ETM that is used to bypass the synchronization of the external inputs **EXTIN[3:0]** and external outputs **EXTOUT[1:0]**. When EXTSBYPASS is LOW: • a HIGH output on any bit of **EXTOUT[1:0]** is held until the corresponding bit of **EXTOUTACK[1:0]** is asserted HIGH. • a HIGH on any bit of **EXTIN[3:0]** must remain held until the ETM asserts the corresponding bit of **EXTINACK[3:0]** HIGH. When **EXTSBYPASS** is HIGH, **EXTOUT[1:0]** outputs are valid on the rising edge of **CLK** and **EXTIN[3:0]** inputs must be valid on the rising edge of **CLK**.	-

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
FIFOPEEK[8:0]	Output	For validation purposes only. Indicates when various events occur before being written to the FIFO.	**CLK**
HIVECS	Input	When this signal is LOW the exception vectors start at address `0x0000 0000`. When HIGH the exception vectors start at address `0xFFFF 0000`.	**CLK**
IA[31:0]	Input	Instruction address.	**CLK**
ID15To11[15:11]	Input	Instruction data field.	**CLK**
ID31To25[31:25]	Input	Instruction data field.	**CLK**
IJBIT	Input	When this signal is HIGH it indicates the ARM processor is in JAVA state. This signal is valid with the address.	**CLK**
InMREQ	Input	If this signal is LOW at the end of the cycle then the processor requires an instruction memory access in the following cycle.	**CLK**
INSTREXEC	Input	This signal indicates the instruction in the Execute stage of the pipeline follower of the ETM has been executed.	**CLK**
INSTRVALID	Input	This signal indicates the instruction in the Execute stage is valid and has not been flushed.	**CLK**
ISEQ	Input	If this signal is HIGH at the end of the cycle then any instruction memory access in the following cycle is sequential from the last instruction memory access.	**CLK**
ITBIT	Input	When HIGH this signal indicates the ARM processor is in Thumb state. When this signal is LOW the processor is in ARM state. This signal is valid with the address.	**CLK**
LATECANCEL	Input	Cancel coprocessor instruction. If this signal is HIGH during the first memory cycle of a coprocessor instruction, then the coprocessor must cancel the instruction without changing any internal state. This signal is only asserted in cycles where the previous instruction accessed memory and a Data Abort occurred.	**CLK**

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
MAXCORES[2:0]	Input	Where an ETM is shared between multiple cores, this signal specifies the number of cores the ETM can trace. It must be tied to the number of cores sharing the ETM minus 1.	-
MAXEXTIN[2:0]	Input	External inputs supported by the ASIC (maximum 4). This appears in the configuration code register.	-
MAXEXTOUT[1:0]	Input	External outputs supported by the ASIC (maximum 2). This appears in the configuration code register.	-
MAXPORTSIZE[3:0]	Input	Maximum port size supported. See *MAXPORTSIZE in ETM9CS and ETM9CSSingle* on page 2-27.	-
NIDEN	Input	Non-invasive debug enable.	-
nPORESET	Input	Main reset. Resets all registers in the CLK domain.	Internally synchronized
PADDRDBG[11:2]	Input	Debug APB Address Bus.	**PCLKDBG**
PADDRDBG31	Input	Originates as an output signal from either the ETMJTAGPORT or *Debug Access Por*t (DAP). **PADDRDBG31** at logic 1 indicates an access from hardware (JTAG). **PADDRDBG31** at logic 0 indicates an access from software.	**PCLKDBG**
PASS	Input	This signal indicates that the coprocessor instruction in the Execute stage of the pipeline follower of the ETM has been executed.	**CLK**
PCLKDBG	Input	Debug APB clock.	-
PCLKENDBG	Input	Debug APB clock enable.	**PCLKDBG**
PENABLEDBG	Input	The Debug APB interface is enabled for a transfer.	**PCLKDBG**
PORTMODE[2:0]	Output	Currently requested port mode. Not used in a CoreSight system.	**CLK**
PORTSIZE[3:0]	Output	Currently requested port size. Not used in a CoreSight system	**CLK**
PRDATADBG[31:0]	Output	Debug APB read data.	**PCLKDBG**

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
PREADYDBG	Output	Used to extend Debug APB transfers.	**PCLKDBG**
PRESETDBGn	Input	Debug APB interface reset.	Internally synchronized
PROCID[31:0]	Input	Process ID.	**CLK**
PROCIDWR	Input	This signal must be asserted whenever the PROCID bus changes. This causes the ETM to output the new context ID at the next available opportunity.	**CLK**
PSELDBG	Input	Debug APB Slave select signal.	**PCLKDBG**
PSLVERRDBG	Output	Debug APB error signal	**PCLKDBG**
PWDATADBG[31:0]	Input	Debug APB write data.	**PCLKDBG**
PWRITEDBG	Input	Debug APB transfer direction (!Read/Write).	**PCLKDBG**
RANGEOUT0	Input	This signal indicates that the corresponding watchpoint unit has matched the conditions currently present on the address, control and data buses. This signal is independent of the state of the enable control bit of the watchpoint unit.	**CLK**
RANGEOUT1	Input	This signal indicates that the corresponding watchpoint unit has matched the conditions currently present on the address, control and data buses. This signal is independent of the state of the enable control bit of the watchpoint unit.	**CLK**
RSTBYPASS	Input	Reset synchronization bypass.	-
SE	Input	Scan enable.	-
TRIGOUT	Output	Trigger request status signal.	See footnote [b]
TRIGOUTACK	Input	ATB trigger acknowledge.	-

Table A-2 ETM9CS signals (continued)

Signal name	Input/output	Description	Clock domains
TRIGSBYPASS	Input	Trigger synchronization bypass.	-
ZIFIRST	Input	When the ARM9 processor is in Java state this signal is asserted on the first ARM instruction to be traced. (No more than two instructions are ever traced for a bytecode.) If only one ARM instruction is traced in Java state, both **ZIFIRST** and **ZILAST** are asserted.	**CLK**
ZILAST	Input	When the ARM9 processor is in Java state this signal is asserted on the last ARM instruction to be traced. (No more than two instructions are ever traced for a bytecode.) If only one ARM instruction is traced in Java state, both **ZIFIRST** and **ZILAST** are asserted	**CLK**

a. If **EXTSBYPASS** is 1, then **EXTIN[3:0]** must be generated on CLK and **EXTOUT[1:0]** must be sampled on **CLK**. If **EXTSBYPASS** is 0, then there is no such restriction. See *CoreSight ETM9 Integration Manual* for further details.
b. If **TRIGSBYPASS** is 1, then **TRIGOUT** must be sampled on ATCLK. If **TRIGSBYPASS** is 0, then there is no such restriction.

Appendix B
I/O Signal Timings

This appendix lists and describes the CoreSight ETM9 I/O timing. It contains the following sections:

- *ETM9CSSingle I/O timing parameters* on page B-2
- *ETM9CS I/O timing parameters* on page B-6.

B.1 ETM9CSSingle I/O timing parameters

Signals are classified according to the percentage of the clock period taken up by internal logic.

- For inputs this is the delay between the input port and the first register.
- For outputs this is the delay between the last register and the output port.

The timing classifications used are:

Early Less than 20% of the period described above.

Middle Between 20% and 80% of the period described above.

Late Greater than 80% of the period described above.

Table B-1 describes the ETM9CSSingle signal timing parameters.

Table B-1 ETM9CSSingle signal timing parameters

Signal name	Timing classification	Input/ Output
ASICCTL[7:0]	Middle	Output
ATCLK	-	Input
ATCLKEN	Late	Input
ATRESETn	Late	Input
BIGEND	Middle	Input
CHSD[1:0]	Middle	Input
CHSE[1:0]	Middle	Input
CLK	-	Input
CORESELECT[2:0]	Middle	Output
CLKEN	Middle	Input
DA[31:0]	Middle	Input
DABORT	Middle	Input
DBGACK	Middle	Input
DBGEN	Late	Input
DD[31:0]	Middle	Input
DDIN[31:0]	Middle	Input

Table B-1 ETM9CSSingle signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
DMAS[1:0]	Middle	Input
DnMREQ	Middle	Input
DnRW	Middle	Input
DSEQ	Middle	Input
ETBTRACEDATA[31:0]	Middle	Output
ETBTRACEVALID	Middle	Output
ETBTRIGGER	Middle	Output
ETMDBGRQ	Middle	Output
ETMEN	Middle	Output
ETMPWRUP	Late	Output
EXTIN[3:0]	Middle	Input
EXTINACK[3:0]	Middle	Output
EXTOUT[1:0]	Middle	Output
EXTOUTACK[1:0]	Middle	Input
EXTSBYPASS	Middle	Input
FIFOPEEK[8:0]	Middle	Output
HIVECS	Middle	Input
IA[31:0]	Middle	Input
ID15To11[15:11]	Middle	Input
ID31To25[31:25]	Middle	Input
IJBIT	Middle	Input
JTAGSBYPASS	Late	Input
InMREQ	Middle	Input
INSTREXEC	Middle	Input

Table B-1 ETM9CSSingle signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
INSTRVALID	Middle	Input
ISEQ	Middle	Input
ITBIT	Middle	Input
LATECANCEL	Middle	Input
MAXCORES[2:0]	Middle	Input
MAXEXTIN[2:0]	Middle	Input
MAXEXTOUT[1:0]	Middle	Input
MAXPORTSIZE[3:0]	Middle	Input
NIDEN	Late	Input
nPORESET	Late	Input
nTRST	Middle	Input
PADDR[11:2]	Late	Input
PASS	Middle	Input
PCLK	-	Input
PCLKEN	Late	Input
PENABLE	Late	Input
PORTMODE[2:0]	Middle	Output
PORTSIZE[3:0]	Middle	Output
PRDATA[31:0]	Late	Output
PREADY	Late	Output
PRESETn	Middle	Input
PROCID[31:0]	Middle	Input
PROCIDWR	Middle	Input
PSEL	Late	Input

Table B-1 ETM9CSSingle signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
PSLVERR	Late	Output
PWDATA[31:0]	Late	Input
PWRITE	Late	Input
RANGEOUT0	Middle	Input
RANGEOUT1	Middle	Input
RSTBYPASS	Late	Input
RTCK	Middle	Output
SE	Late	Input
TCK	Late	Input
TDI	Late	Input
TDO	Middle	Output
TDOSEL	Middle	Output
TMS	Late	Input
TRACECLK	Middle	Output
TRACECTL	Middle	Output
TRACEDATA[31:0]	Middle	Output
ZIFIRST	Middle	Input
ZILAST	Middle	Input

Note

Actual clock frequencies and input and output timing constraints vary according to application requirements and the silicon process technologies used. The maximum operating clock frequencies change according to the constraints and the process technology you use.

B.2 ETM9CS I/O timing parameters

Signals are classified according to the percentage of the clock period taken up by internal logic.

- For inputs this is the delay between the input port and the first register.
- For outputs this is the delay between the last register and the output port.

The timing classifications used are:

Early Less than 20% of the period described above.

Middle Between 20% and 80% of the period described above.

Late Greater than 80% of the period described above.

Table B-2 describes the ETM9CS signal timing parameters.

Table B-2 ETM9CS signal timing parameters

Signal name	Timing classification	Input/ Output
AFREADYM	Middle	Output
AFVALIDM	Middle	Input
ASICCTL[7:0]	Middle	Output
ATBYTESM[1:0]	Middle	Output
ATCLK	-	Input
ATCLKEN	Middle	Input
ATDATAM[31:0]	Middle	Output
ATIDM[6:0]	Middle	Input
ATREADYM	Middle	Input
ATRESETn	Late	Input
ATVALIDM	Middle	Output
BIGEND	Middle	Input
CHSD[1:0]	Middle	Input
CHSE[1:0]	Middle	Input
CLK	-	Input
CLKEN	Middle	Input

Table B-2 ETM9CS signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
CORESELECT[2:0]	Middle	Output
DA[31:0]	Middle	Input
DABORT	Middle	Input
DBGACK	Middle	Input
DBGEN	Late	Input
DD[31:0]	Middle	Input
DDIN[31:0]	Middle	Input
DMAS[1:0]	Middle	Input
DnMREQ	Middle	Input
DnRW	Middle	Input
DSEQ	Middle	Input
ETMDBGRQ	Middle	Output
ETMEN	Middle	Output
ETMPWRUP	Late	Output
ETPSUP	Late	Input
EXTIN[3:0]	Middle	Input
EXTINACK[3:0]	Middle	Output
EXTOUT[1:0]	Middle	Output
EXTOUTACK[1:0]	Middle	Input
EXTSBYPASS	Middle	Input
FIFOPEEK[8:0]	Middle	Output
HIVECS	Middle	Input
IA[31:0]	Middle	Input
ID15To11[15:11]	Middle	Input

Table B-2 ETM9CS signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
ID31To25[31:25]	Middle	Input
IJBIT	Middle	Input
InMREQ	Middle	Input
INSTREXEC	Middle	Input
INSTRVALID	Middle	Input
ISEQ	Middle	Input
ITBIT	Middle	Input
LATECANCEL	Middle	Input
MAXCORES[2:0]	Middle	Input
MAXEXTIN[2:0]	Middle	Input
MAXEXTOUT[1:0]	Middle	Input
MAXPORTSIZE[3:0]	Middle	Input
NIDEN	Late	Input
nPORESET	Late	Input
PADDRDBG[11:2]	Middle	Input
PADDRDBG31	Middle	Input
PASS	Middle	Input
PCLKDBG	-	Input
PCLKENDBG	Middle	Input
PENABLEDBG	Middle	Input
PORTMODE[2:0]	Middle	Output
PORTSIZE[3:0]	Middle	Output
PRDATADBG[31:0]	Middle	Output
PREADYDBG	Middle	Output

Table B-2 ETM9CS signal timing parameters (continued)

Signal name	Timing classification	Input/ Output
PRESETDBGn	Late	Input
PROCID[31:0]	Middle	Input
PROCIDWR	Middle	Input
PSELDBG	Middle	Input
PSLVERRDBG	Middle	Output
PWDATADBG[31:0]	Middle	Input
PWRITEDBG	Middle	Input
RANGEOUT0	Middle	Input
RANGEOUT1	Middle	Input
RSTBYPASS	Late	Input
SE	Late	Input
TRIGOUT	Middle	Output
TRIGOUTACK	Middle	Input
TRIGSBYPASS	Middle	Input
ZIFIRST	Middle	Input
ZILAST	Middle	Input

Note

Actual clock frequencies and input and output timing constraints vary according to application requirements and the silicon process technologies used. The maximum operating clock frequencies change according to the constraints and the process technology you use.

Glossary

This glossary describes some of the terms used in this manual. Where terms can have several meanings, the meaning presented here is intended.

Advanced High-performance Bus (AHB)

The AMBA Advanced High-performance Bus system connects embedded processors such as an ARM core to high-performance peripherals, DMA controllers, on-chip memory, and interfaces. It is a high-speed, high-bandwidth bus that supports multi-master bus management to maximize system performance.

See also Advanced Microcontroller Bus Architecture and AHB-Lite.

Advanced Microcontroller Bus Architecture(AMBA)

AMBA is the ARM open standard for multi-master on-chip buses, capable of running with multiple masters and slaves. It is an on-chip bus specification that details 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. AHB conforms to this standard.

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

Advanced Peripheral Bus (APB)
: The AMBA Advanced Peripheral Bus is a simpler bus protocol than 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
: AHB-Lite is a subset of the full AHB specification. It is intended for use in designs where only a single AHB master is used. This can be a simple single AHB master system or a multi-layer AHB system where there is only one AHB master on a layer.

Aligned
: Refers to data items stored so that their address is divisible by the highest power of two that divides their size. Aligned words and halfwords therefore have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore refer to addresses that are divisible by four and two respectively. The terms byte-aligned and doubleword-aligned are defined similarly.

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 state
: A processor that is executing ARM (32-bit) instructions is operating in ARM state.

  See also Thumb state.

Big-endian
: Memory organization in which the least significant byte of a word is at a higher address than the most significant byte.

  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 1.

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 halted unconditionally. Breakpoints are inserted by programmers to allow inspection of register contents, memory locations, and/or 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 AHB buses are controlled using the **HBURST** signals to specify if transfers are single, four-beat, eight-beat, or 16-beat bursts, and to specify how the addresses are incremented.

Byte invariant Refers to the way of switching between little-endian and big-endian operation that leaves byte accesses entirely unchanged. Accesses to other data sizes are necessarily affected by such endianness switches.

Byte lane strobe An AHB signal, **HBSTRB**, that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of **HBSTRB** corresponds to eight bits of the data bus.

Central Processing Unit (CPU)
The part of a processor that contains the ALU, the registers, and the instruction decode logic and control circuitry. Also commonly known as the processor core.

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

Core reset *See* Warm reset.

CPI *See* Cycles per instruction.

CPRT Coprocessor Register Transfer

CPU *See* Central Processing Unit.

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 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.

ETM *See* Embedded Trace Macrocell.

Exception An event that occurs during program operation that makes continued normal operation inadvisable or impossible, and so makes it necessary to change the flow of control in a program. Exceptions can be caused by error conditions in hardware or software. The processor can respond to exceptions by running appropriate exception handler code that attempts to remedy the error condition, and either restarts normal execution or ends the program in a controlled way.

Exception 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 service routine.

Fast Context Switch Extension (FCSE)
This 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.

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 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.

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
A feature that is not architecturally defined, and which might vary between implementations. The feature is defined and documented for each individual implementation.

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

Little-endian Memory organization where the least significant byte of a word is at a lower address than the most significant byte.

See also Big-endian and Endianness.

Monitor mode One of two mutually exclusive debug modes. In monitor mode the ARM9 processor enables a software abort handler provided by the debug monitor or operating system debug task. When a breakpoint or watchpoint is encountered, this enables vital system interrupts to continue to be serviced while normal program execution is suspended.

See also Halt 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 contraction of microprocessor. A processor includes the CPU or core, plus additional components such as memory, and interfaces. These are combined as a single macrocell, that can be fabricated on an integrated circuit.

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. Java instructions that are accelerated by hardware can cause a number of reads to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

Reduced Instruction Set Computer (RISC)
A computer architecture that reduces chip complexity by limiting the complexity of instructions that can be executed. In RISC computers, there is no microcode layer, and instruction size is fixed.

Register A temporary storage location used to hold binary data until it is ready to be used.

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 zero and are read as zero.

RISC *See* Reduced Instruction Set Computer.

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 a 0 produces Unpredictable results.

Should Be Zero (SBZ)
Should be written as 0 (or all 0s for bit fields) by software. Writing a 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 the same value back 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 terminals and one optional terminal 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**.

TPA *See* Test Port Analyzer.

Trace Port Analyzer 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.

Thumb instruction A halfword that specifies an operation for an ARM processor in Thumb state to perform. Thumb instructions must be halfword-aligned.

Thumb state A processor that is executing Thumb (16-bit) halfword aligned instructions is operating in Thumb state.

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

Unpredictable 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.

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.

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. Java instructions that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.