CoreSight™ ETM™-R4

Revision: r1p0

Technical Reference Manual

ARM®

ARM DDI 0367B

CoreSight ETM-R4
Technical Reference Manual

Release Information

Change History

Date	Issue	Confidentiality	Change
18 January 2006	A	Confidential	First release, for r0p0.
23 November 2007	B	Non-Confidential	First release for r1p0.

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 ETM-R4 Technical Reference Manual

List of Tables
CoreSight ETM-R4 Technical Reference Manual

List of Figures
CoreSight ETM-R4 Technical Reference Manual

Preface

This preface introduces the *CoreSight™ ETM™-R4 Technical Reference Manual*. It contains the following sections:

- *About this manual* on page x
- *Feedback* on page xv.

About this manual

This is the *Technical Reference Manual* (TRM) for the *CoreSight Embedded Trace Macrocell*™ for the Cortex-R4 and Cortex-R4F processors, the CoreSight ETM-R4 macrocell.

You implement the ETM-R4 macrocell with either the Cortex-R4 processor or the Cortex-R4F processor. In this manual, in general:

- any reference to the processor applies to either the Cortex-R4 processor or the Cortex-R4F processor, as appropriate
- any reference to the Cortex-R4 processor applies also to the Cortex-R4F 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*m* identifier indicates the revision status of the product described in this manual, where:

r*n* Identifies the major revision of the product.

p*m* 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. You can find complementary information in the *ETM Architecture Specification* (ARM IHI 0014).
- Hardware and software engineers integrating the macrocell into an ASIC that includes a Cortex™-R4 processor. You can find complementary information in the *CoreSight ETM-R4 Integration Manual* (ARM DII 0133).

Using this manual

This manual is organized into the following chapters:

Chapter 1 *Introduction*

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

Chapter 2 ***Implementation-defined Behavior***

Read this chapter for a description of the Implementation-defined features of the macrocell.

Chapter 3 ***Programmer's Model***

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

Appendix A ***Signals Lists***

Read this appendix for a description of all signals.

Appendix B ***I/O Signal Timings***

Read this appendix for a description of the macrocell timing parameters.

Appendix C ***Typical APB Transfers***

Read this appendix for a description of APB read and write transfers.

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

Conventions

Conventions that this manual can use are described in:

- *Typographical*
- *Timing diagrams* on page xii
- *Signals* on page xii
- *Numbering* on page xiii.

Typographical

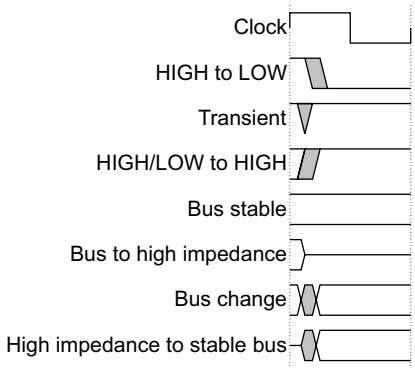
This manual uses the following typographical conventions:

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 >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

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

Signals

The signal conventions are:

Signal level	The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means: • HIGH for active-HIGH signals • LOW for active-LOW signals.
Lower-case n	At the start or end of a signal name denotes an active-LOW signal.
Prefix A	Denotes global *Advanced eXtensible Interface* (AXI) signals:

Prefix AF Denotes Advanced Trace Bus (ATB) Flush signals.

Prefix AR Denotes AXI read address channel signals.

Prefix AT Denotes Advanced Trace Bus (ATB) signals, other than ATB flush signals.

Prefix AW Denotes AXI write address channel signals.

Prefix B Denotes AXI write response channel signals.

Prefix C Denotes AXI low-power interface signals.

Prefix H Denotes *Advanced High-performance Bus* (AHB) signals.

Prefix P Denotes *Advanced Peripheral Bus (*APB) signals.

Prefix R Denotes AXI read data channel signals.

Prefix W Denotes AXI write data channel signals.

Numbering

The numbering convention is:

<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.

This section lists publications by ARM. You can access ARM documentation at:

`http://infocenter.arm.com/help/index.jsp`

ARM publications

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

- *CoreSight ETM-R4 Configuration and Sign-off Guide (*ARM DII 0132)
- *CoreSight ETM-R4 Integration Manual* (ARM DII 0133)
- *ETM Architecture Specification* (ARM IHI 0014)
- *CoreSight Design Kit R4 Technical Reference Manual* (ARM DDI 0314)
- *CoreSight Design Kit R4 Integration Manual* (ARM DII 0134)
- *ARM Reference Peripheral Specification* (ARM DDI 0062)
- *Cortex™-R4 Technical Reference Manual* (ARM DDI 0363).

Feedback

ARM Limited welcomes feedback on the macrocell and its documentation.

Feedback on this 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 manual

If you have any comments on this manual, send your emails 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 the macrocell. It contains the following sections:

1.1 About the macrocell

This macrocell provides instruction trace and data trace for the Cortex-R4 microprocessor. It is compatible with Thumb® and Thumb-2.

It functions as the ETMR4 module in a CoreSight System. See the *CoreSight Design Kit R4 Technical Reference Manual* for more information about how a CoreSight system uses the ETMR4 module.

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

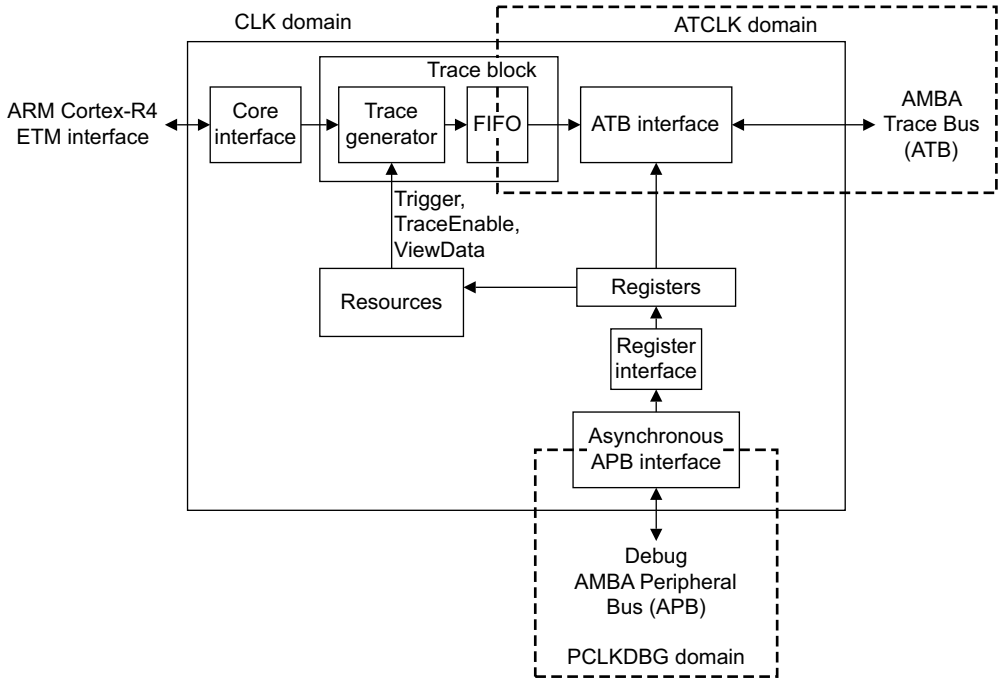


Figure 1-1 Macrocell functional blocks and clock domains

See Appendix A *Signals Lists* for information about the macrocell signals.

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

Programming access to the macrocell is through the APB interface. In a CoreSight system, the APB interface connects to the Debug APB bus.

1.2 Macrocell configuration

Table 1-1 shows the macrocell resources.

Table 1-1 Macrocell resources

Resource description	Number or size
Context ID comparators	1
External inputs	4
External outputs	2
Sequencers	1
Counters	2
Memory map decoders	0
Data value comparators	2
Pairs of address comparators	4
Extended external inputs	47
Extended external input selectors	2
FIFO size	72 bytes
Trace port configuration	32-bit ATB
Trace Start/Stop block	1
ASICCTL general-purpose I/O bus	8 bit

1.3 Product revision information

This manual is for the r1p0 revision of the ETM-R4 macrocell. This revision of the ETM is for major revision r1 of the Cortex-R4 or Cortex-R4F processor. See *Product revision status* on page x for more information about revision numbering.

The major changes in the r1p0 release of the ETM are:

- the **EVNTBUS** input signal width increases from 29 to 47 bits
- the number of valid bits in each Selection value field in the Extended External Input Selection Register, register `0x07B`, increases from 5 to 6
- an extra bit is defined in the ITETMIF Integration Test Register, register `0x3B6`, to return the value of the **EVNTBUS[46]** signal
- the value of the Size of extended external input bus field in the Configuration Code Extension Register, register `0x07A`, changes from 29 to 47
- the revision fields in the ID registers change, to indicate the r1p0 revision:
  - the Implementation revision field of the ETM ID Register, register `0x079`, changes to b0001
  - the Revision field of the PeripheralID2 Register, register `0x3FA`, changes to b0001
- file, module and macro names are changed to make them unique, to avoid clashes with the names used on other ETMs.

Chapter 2
Implementation-defined Behavior

This chapter describes information that is implementation-specific and that relates to the macrocell. It contains the following sections:

2.1 ETM architecture version

The macrocell implements version 3.3 of the ETM architecture, ETMv3.3. See the *ETM Architecture Specification* for more information.

2.2 Implementation-defined registers

In terms of how they are defined, there are two groups of ETM registers:

- registers that are completely defined by the *ETM Architecture Specification*
- registers that are at least partly Implementation-defined.

Chapter 3 *Programmer's Model* gives more information about the ETM registers, in the sections:

- *Summary of ETM registers* on page 3-6
- *Descriptions of Implementation-defined registers* on page 3-17.

In Chapter 3, the following sections describe each of the Implementation-defined registers:

- *ETM Control Register* on page 3-17
- *Configuration Code Register* on page 3-22
- *ASIC Control Register* on page 3-24
- *ETM ID Register* on page 3-25
- *Configuration Code Extension Register* on page 3-26
- *Extended External Input Selection Register* on page 3-28
- *Power-Down Status register* on page 3-29
- The Integration test registers:
  - *ITETMIF Register, ETM interface* on page 3-35
  - *ITMISCOUT Register, miscellaneous outputs* on page 3-37
  - *ITMISCIN Register, miscellaneous inputs* on page 3-38
  - *ITTRIGGERACK Register, trigger acknowledge* on page 3-39
  - *ITTRIGGERREQ Register, trigger request* on page 3-40
  - *ITATBDATA0 Register, ATB data 0* on page 3-41
  - *ITATBCTR0 Register, ATB control 0* on page 3-44
  - *ITATBCTR1 Register, ATB control 1* on page 3-43
  - *ITATBCTR2 Register, ATB control 2* on page 3-42
- *Peripheral Identification Registers* on page 3-29
- *Component Identification Registers* on page 3-32.

2.3 Precise TraceEnable events

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

- single address comparators
- address range comparators.

2.4 Parallel instruction execution

The Cortex-R4 processor supports parallel instruction execution. This means the macrocell is capable of tracing two instructions per cycle, although only the first instruction can have data associated with it.

Although the trace start/stop block is evaluated for each instruction as required, the macrocell cannot trace one instruction without the other. In other words, if one instruction is traced, the instruction it is paired with is always traced as well. If **ViewData** is active, any data associated with the paired instruction is also traced.

2.5 Context ID tracing

The macrocell 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. This means that if context ID tracing is enabled, an MCR instruction that changes the context ID does not have its data traced separately.

2.6 Trace and Comparator features

In ETM Architecture v3.3, it is Implementation-defined whether an ETM macrocell supports a number of Trace and Comparator features. This section specifies the implementation of these features on the CoreSight ETM-R4 macrocell:

- *Trace features*
- *Comparator features*.

2.6.1 Trace features

It is Implementation-defined whether an ETM macrocell supports the following features:

- data value and data address tracing
- data suppression
- cycle-accurate tracing.

All of these features are implemented on the CoreSight ETM-R4 macrocell.

For descriptions of these features see the *ETM Architecture Specification*.

2.6.2 Comparator features

It is Implementation-defined whether an ETM macrocell supports data address comparison.

Data address comparison is implemented on the CoreSight ETM-R4 macrocell.

For a description of data address comparison see the *ETM Architecture Specification*.

2.7 Interaction with the Performance Monitoring Unit, PMU

The Cortex-R4 processor includes a PMU that enables events, such as cache misses and instructions executed, to be counted over a period of time. The macrocell can still use these events by means of the extended external input facility. Each bit in the **EVNTBUS[46:0]** input is mapped to the corresponding extended external input. See the *Cortex-R4 Technical Reference Manual* for details of the mapping of events to bits within this bus.

Some events use two bits. Two of these events can occur in a cycle. They must be dealt with separately if they are to be properly counted.

The Cortex-R4 PMU can count the two external outputs as additional events. These events are not provided back to the macrocell as extended external inputs.

These facilities enable additional filtering of the system events using ETM resources, such as instruction address ranges or the start/stop resource, before they are passed back to the PMU for counting. To do this:

- Configure the ETM extended external input selectors to the system events you want to count.
- Configure the required ETM filtering resource as appropriate.
- Configure the ETM external outputs to extended external input selector AND the required ETM filtering resource.
- Select the ETM external outputs as the events to be counted in the Cortex-R4 PMU.

2.8 ETMR4 clocks

This section describes ETMR4 clocks and clock enables:

- *ETMR4 clock signals*
- *ETMR4 clock enable signals*.

2.8.1 ETMR4 clock signals

ETMR4 contains the following clocks:

CLK This is the main clock for the ETMR4 block and must be the same clock as that wired to the **CLK** input of the Cortex-R4 processor. It can be asynchronous to **PCLKDBG** and **ATCLK**.

PCLKDBG This is the Debug APB interface clock for ETMR4. 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 ETMR4 clock enable signals

ETMR4 has the following clock enable signals:

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

PCLKENDBG This is the Debug APB interface clock enable. It can slow down **PCLKDBG**.

2.9 ETMR4 resets

ETMR4 contains the following resets:

nSYSPORESET This signal is the main power-on reset. It resets all the registers in the ETMR4. It is active LOW.

PRESETDBGn This signal is the Debug APB interface reset. It resets all the registers in the ETMR4. It is active LOW.

2.10 PortMode and PortSize

The macrocell only supports a 32-bit port size, and only supports the dynamic port mode.

In the ETM Control Register, at offset `0x0`, from reset:

- the PortSize bits, bits [21,6:4], take the value b0100, indicating a 32-bit port
- the PortMode bits, bits [13, 17:16], take the value b000, indicating dynamic port mode.

2.11 Other Implementation-defined features of the macrocell

The following Implementation-defined features of the macrocell do not affect the descriptions of the features given in the ETM Architecture Specification:

- Bits [1:0] of the Synchronization Frequency Register are reserved, RAZ, WI.
- Value Not Traced packets are not output in data-only mode. When data address tracing is enabled in data-only mode an address packet is output for each traced data transfer for which the data address is not sequential to the previously traced data transfer.

2.12 Restrictions and limitations

See the *Cortex-R4 Technical Reference Manual* 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 the macrocell. It contains the following sections:

3.1 About the programmer's model

The programmer's model enables you to use the ETM registers to control the macrocell. The following sections describe the programmer's model:

- *Controlling ETM programming* on page 3-3
- *Programming and reading ETM registers* on page 3-5
- *Summary of ETM registers* on page 3-6
- *Descriptions of Implementation-defined registers* on page 3-17.

3.2 Controlling ETM programming

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.

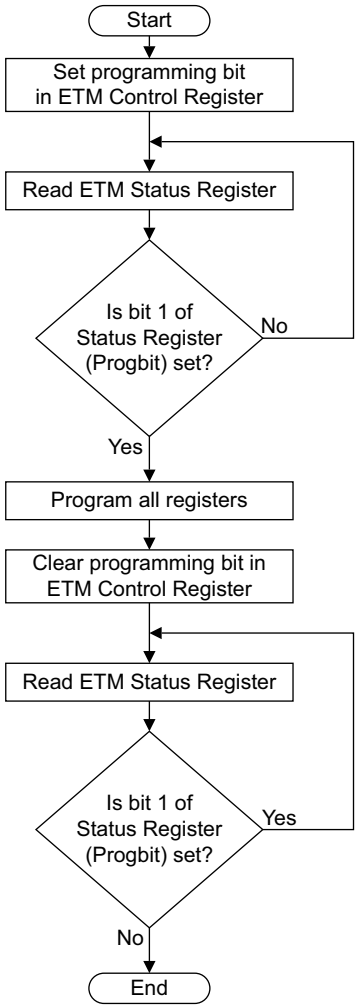


Figure 3-1 Programming ETM registers

The processor does not have to be in the debug state while you program the ETM registers.

3.3 Programming and reading ETM registers

You program and read the ETM registers using the debug APB interface. This is described in the following section.

3.3.1 Software access using APB

The APB interface provides a direct method of programming:

- a stand-alone macrocell
- a macrocell in a CoreSight system.

The APB interface is a simple low cost interface that provides 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.

See Appendix C *Typical APB Transfers* for more information on APB transfers.

3.4 Summary of ETM registers

This section summarizes the ETM registers. For full descriptions of the ETM registers, see:

- *Descriptions of Implementation-defined registers* on page 3-17, for the Implementation-defined registers.
- The *ETM Architecture Specification*, for the other registers.

3.4.1 List of registers in numerical order

Table 3-1 on page 3-7 shows all of the registers, and tells you where each register is described in detail. The registers are listed in register number order.

The macrocell registers are listed by functional group in the section *Functional grouping of registers* on page 3-11. The functional group register tables include additional information about each register:

- The register access type. This is read-only, write-only or read/write.
- The clock domain of the register.
- The base offset address of the register. The base address of a register is always four times its register number.
- Additional information about the implementation of the register, where appropriate.

Note

- Registers not listed here are not implemented. Reading a non-implemented register address returns 0. Writing to a non-implemented register address has no effect.
- In Table 3-1 on page 3-7:
  - the Default value column shows the value of the register immediately after an ETM reset. For read-only registers, every read of the register returns this value.
  - the listed Functional group table gives more information about the register, including whether it is read-only, read/write, or write-only,

Table 3-1 ETM registers summary

Name	Register number	Access	Default value	Group[a]	Description, see
ETM Control	0x000	R/W	0x00000441	1	*ETM Control Register* on page 3-17
Configuration Code	0x001	RO	0x8D014024[b]	1	*Configuration Code Register* on page 3-22
Trigger Event	0x002	R/W	-[c]	4	*ETM Architecture Specification*
ASIC Control	0x003	R/W	-[c]	1	*ASIC Control Register* on page 3-24
ETM Status	0x004	R/W	-[c]	1	*ETM Architecture Specification*
System Configuration	0x005	RO	0x00020C0C[d]	1	*ETM Architecture Specification*
TraceEnable Start/Stop Resource control	0x006	R/W	-[c]	2	*ETM Architecture Specification*
TraceEnable Control 2	0x007	R/W	-[c]	2	*ETM Architecture Specification*
TraceEnable Event	0x008	R/W	-[c]	2	*ETM Architecture Specification*
TraceEnable Control 1	0x009	R/W	-[c]	2	*ETM Architecture Specification*
FIFOFULL Level[e]	0x00B	R/W	-[c]	1	*ETM Architecture Specification*
ViewData Event	0x00C	R/W	-[c]	2	*ETM Architecture Specification*
ViewData Control 1	0x00D	R/W	-[c]	2	*ETM Architecture Specification*
ViewData Control 3	0x00F	R/W	-[c]	2	*ETM Architecture Specification*
Address Comparator Value 1-8	0x010 to 0x017	R/W	-[c]	3	*ETM Architecture Specification*
Address Comparator Access Type 1-8	0x020 to 0x027	R/W	-[c]	3	*ETM Architecture Specification*
Data Comparator Value 1[f]	0x030[f]	R/W	-[c]	3	*ETM Architecture Specification*
Data Comparator Value 3[f]	0x032[f]	R/W	-[c]	3	*ETM Architecture Specification*
Data Comparator Mask 1[f]	0x040[f]	R/W	-[c]	3	*ETM Architecture Specification*

Table 3-1 ETM registers summary (continued)

Name	Register number	Access	Default value	Group[a]	Description, see
Data Comparator Mask 3[f]	0x042[f]	R/W	-[c]	3	*ETM Architecture Specification*
Counter Reload Value 1-2	0x050, 0x051	R/W	-[c]	4	*ETM Architecture Specification*
Counter Enable Event 1-2	0x054, 0x055	R/W	-[c]	4	*ETM Architecture Specification*
Counter Reload Event 1-2	0x058, 0x059	R/W	-[c]	4	*ETM Architecture Specification*
Counter Value 1-2	0x05C, 0x05D	R/W	-[c]	4	*ETM Architecture Specification*
Sequencer State Transition Events	0x060 to 0x065	R/W	-[c]	4	*ETM Architecture Specification*
Current Sequencer State	0x067	R/W	-[c]	4	*ETM Architecture Specification*
External Output Event 1-2	0x068, 0x069	R/W	-[c]	4	*ETM Architecture Specification*
Context ID Comparator Value	0x06C	R/W	-[c]	3	*ETM Architecture Specification*
Context ID Comparator Mask	0x06F	R/W	-[c]	3	*ETM Architecture Specification*
Synchronization Frequency	0x078	R/W	0x00000400	1	*ETM Architecture Specification*
ETM ID	0x079	RO	0x4104F23x[g]	1	*ETM ID Register* on page 3-25
Configuration Code Extension	0x07A	RO	0x0000097A	1	*Configuration Code Extension Register* on page 3-26
Extended External Input Selector	0x07B	R/W	-[c]	4	*Extended External Input Selection Register* on page 3-28
CoreSight Trace ID	0x080	R/W	0x00000000	1	See *ETM Architecture Specification*
Power-Down Status register	0x0C5	RO	-[c]	1	*Power-Down Status register* on page 3-29

Table 3-1 ETM registers summary (continued)

Name	Register number	Access	Default value	Group[a]	Description, see
ITETMIF	0x3B6	RO[h]	-[i]	6	*ITETMIF Register, ETM interface* on page 3-35
ITMISCOUT	0x3B7	WO	n/a[j]	6	*ITMISCOUT Register, miscellaneous outputs* on page 3-37
ITMISCIN	0x3B8	RO[h]	-[i]	6	*ITMISCIN Register, miscellaneous inputs* on page 3-38
ITTRIGGERACK	0x3B9	RO[h]	-[i]	6	*ITTRIGGERACK Register, trigger acknowledge* on page 3-39
ITTRIGGERREQ	0x3BA	WO	n/a[j]	6	*ITTRIGGERREQ Register, trigger request* on page 3-40
ITATBDATA0	0x3BB	WO	n/a[j]	6	*ITATBDATA0 Register, ATB data 0* on page 3-41
ITATBCTR2	0x3BC	RO[h]	-[i]	6	*ITATBCTR2 Register, ATB control 2* on page 3-42
ITATBCTR1	0x3BD	WO	n/a[j]	6	*ITATBCTR1 Register, ATB control 1* on page 3-43
ITATBCTR0	0x3BE	WO	n/a[j]	6	*ITATBCTR0 Register, ATB control 0* on page 3-44
Integration Mode Control	0x3C0	R/W	0x00000000	5	*ETM Architecture Specification*
Claim Tag Set	0x3E8	R/W	0x000000FF	5	*ETM Architecture Specification*
Claim Tag Clear	0x3E9	R/W	0x00000000	5	*ETM Architecture Specification*
Lock Access	0x3EC	WO	n/a[j]	5	*ETM Architecture Specification*
Lock Status	0x3ED	RO	-[i]	5	*ETM Architecture Specification*
Authentication Status	0x3EE	RO	-[i]	5	*ETM Architecture Specification*
Device Configuration	0x3F2	RO	0x00000000	5	*ETM Architecture Specification*
Device Type	0x3F3	RO	0x00000013	5	*ETM Architecture Specification*

Table 3-1 ETM registers summary (continued)

Name	Register number	Access	Default value	Group[a]	Description, see
Peripheral ID4 to 7	0x3F4 to 0x3F7	RO	-[i]	5	*Peripheral Identification Registers* on page 3-29
Peripheral ID0 to 3	0x3F8 to 0x3FB	RO	-[i]		
Component ID0 to 3	0x3FC to 0x3FF	RO	-[i]	5	*Component Identification Registers* on page 3-32

a. Functional group. For more information, see:
for Group 1, *General Control and ID registers* on page 3-11, Table 3-2 on page 3-11
for Group 2, *TraceEnable and ViewData registers* on page 3-12, Table 3-3 on page 3-12
for Group 3, *Comparator registers* on page 3-13, Table 3-4 on page 3-13
for Group 4, *Counter, Sequencer and other resource registers* on page 3-14, Table 3-5 on page 3-14
for Group 5, *CoreSight Management registers* on page 3-15, Table 3-6 on page 3-15
for Group 5, *Integration Test registers* on page 3-16, Table 3-7 on page 3-16.

b. Default value when **MAXEXTOUT[1:0]** and **MAXEXTIN[2:0]** are all tied LOW (0), see the register description for more information.

c. These registers are not reset by a reset of the macrocell. Therefore, they do not have specific default values.

d. Bits [14:12] of the System Configuration Register are tied to the **MAXCORES[2:0]** signals. If a **MAXCORES** bit is High then the corresponding bit in the System Configuration Register is set to 1, for example if **MAXCORES[0]** is tied HIGH then bit [12] is set to 1. The default value given is for all **MAXCORES** signals tied LOW, bits [14:12] = b000.
For more information about the **MAXCORES[2:0]** signals, see *ETMR4 Signals* on page A-2

e. Although the macrocell does not include FIFOFULL logic, the FIFOFULL Level Register controls the FIFO level at which data suppression occurs. For more information see the *ETM Architecture Specification*.

f. In the Data Comparator register area, even number registers are reserved. For the CoreSight ETM-R4, reserved areas are:
Register 0x031, Data Comparator Value 1, at offset 0x0C4 Register 0x033, Data Comparator Value 3, at offset 0x0CC
Register 0x041, Data Comparator Mask 1, at offset 0x104 Register 0x043, Data Comparator Mask 3, at offset 0x10C.
You must not write to these reserved register addresses. Reads from these addresses are Unpredictable.

g. The value of bits [3:0] of the ETM ID Register depend on the macrocell revision, see the register description for more information.

h. The values of the read-only Integration Test registers are valid only when the macrocell is in Integration Test mode. If you read one of these registers when the macrocell is in normal operating mode the result returned is Unknown.

i. See the register description for details.

j. Not applicable. These are write-only registers.

3.4.2 Functional grouping of registers

This section lists the macrocell registers by functional group, as follows:

- *General control and ID registers*
- *TraceEnable and ViewData registers* on page 3-12
- *Comparator registers* on page 3-13
- *Counter, Sequencer and other resource registers* on page 3-14
- *CoreSight Management registers* on page 3-15
- *Integration Test registers* on page 3-16.

These functional groups include all of the registers.

General control and ID registers

Table 3-2 shows the General Control and ID registers.

Table 3-2 General Control and ID registers

Name	Register number	Base offset	Clock domain	Description, see
ETM Control	0x000	0x000	**CLK**	*ETM Control Register* on page 3-17
Configuration Code	0x001	0x004	**CLK**	*Configuration Code Register* on page 3-22
ASIC Control	0x003	0x00C	**CLK**	*ASIC Control Register* on page 3-24
ETM Status	0x004	0x010	**CLK**	*ETM Architecture Specification.*
System Configuration	0x005	0x014	**CLK**	*ETM Architecture Specification.*
FIFOFULL Level[a]	0x00B	0x02C	**CLK**	*ETM Architecture Specification.*
Synchronization Frequency	0x078	0x1E0	**CLK**	*ETM Architecture Specification.*[b]
ETM ID	0x079	0x1E4	**CLK**	*ETM ID Register* on page 3-25
Configuration Code Extension	0x07A	0x1E8	**CLK**	*Configuration Code Extension Register* on page 3-26
CoreSight Trace ID	0x080	0x200	**CLK**	*ETM Architecture Specification.*
Power-Down Status register	0x0C5	0x314	**CLK**	*Power-Down Status register* on page 3-29

a. Although the macrocell does not include FIFOFULL logic, the FIFOFULL Level Register controls the FIFO level at which data suppression occurs. For more information see the *ETM Architecture Specification*.
b. Only bits [11:2] of the Synchronization Frequency Register are implemented. Bits [1:0] Read-As-Zero.

TraceEnable and ViewData registers

Table 3-3 shows the TraceEnable and ViewData registers.

Table 3-3 TraceEnable and ViewData registers

Name	Register number	Base offset	Clock domain	Description, see
TraceEnable Start/Stop Resource control	0x006	0x018	**CLK**	*ETM Architecture Specification*.
TraceEnable Control 2	0x007	0x01C	**CLK**	*ETM Architecture Specification*.
TraceEnable Event	0x008	0x020	**CLK**	*ETM Architecture Specification*.
TraceEnable Control 1	0x009	0x024	**CLK**	*ETM Architecture Specification*.
ViewData Event	0x00C	0x030	**CLK**	*ETM Architecture Specification*.
ViewData Control 1	0x00D	0x034	**CLK**	*ETM Architecture Specification*.
ViewData Control 3	0x00F	0x03C	**CLK**	*ETM Architecture Specification*.

Comparator registers

Table 3-4 shows the Comparator registers. These control the Address, Data and Context ID comparators.

Table 3-4 Comparator registers

Name	Register number	Base offset	Clock domain	Description, see
Address Comparator Value 1-8	0x010 to 0x017	0x040 to 0x05F	**CLK**	*ETM Architecture Specification*.
Address Comparator Access Type 1-8	0x020 to 0x027	0x080 to 0x09F	**CLK**	*ETM Architecture Specification*.[a]
Data Comparator Value 1[b]	0x030[b]	0x0C0[b]	**CLK**	*ETM Architecture Specification*.
Data Comparator Value 3[b]	0x032[b]	0x0C8[b]	**CLK**	*ETM Architecture Specification*.
Data Comparator Mask 1[b]	0x040[b]	0x100[b]	**CLK**	*ETM Architecture Specification*.
Data Comparator Mask 3[b]	0x042[b]	0x108[b]	**CLK**	*ETM Architecture Specification*.
Context ID Comparator Value	0x06C	0x1B0	**CLK**	*ETM Architecture Specification*.
Context ID Comparator Mask	0x06F	0x1BC	**CLK**	*ETM Architecture Specification*.

a. Because the Cortex-R4 processor does not implement the Security Extensions, only bits [9:0] of the Address Comparator Access Type Registers are implemented.

b. In the Data Comparator register area, even number registers are reserved. For the CoreSight ETM-R4, reserved areas are:
Register 0x031, Data Comparator Value 1, at offset 0x0C4 Register 0x033, Data Comparator Value 3, at offset 0x0CC
Register 0x041, Data Comparator Mask 1, at offset 0x104 Register 0x043, Data Comparator Mask 3, at offset 0x10C.
You must not write to these reserved register addresses. The value of a reads from these addresses is Unknown.

Counter, Sequencer and other resource registers

Table 3-5 shows the Counter, Sequencer and other resource registers. These control:

- the two Counters, and associated events
- the Sequencer, and associated state change events
- Trigger events
- EXTOUT (External Output) events
- Extended External Input selection.

Table 3-5 Counter, Sequencer and other resource registers

Name	Register number	Base offset	Clock domain	Description, see
Trigger Event	0x002	0x008	**CLK**	*ETM Architecture Specification*.
Counter Reload Value 1-2	0x050, 0x051	0x140, 0x144	**CLK**	*ETM Architecture Specification*.
Counter Enable Event 1-2	0x054, 0x055	0x150, 0x154	**CLK**	*ETM Architecture Specification*.
Counter Reload Event 1-2	0x058, 0x059	0x160, 0x164	**CLK**	*ETM Architecture Specification*.
Counter Value 1-2	0x05C, 0x05D	0x170, 0x174	**CLK**	*ETM Architecture Specification*.
Sequencer State Transition Events	0x060 to 0x065	0x180 to 0x194	**CLK**	*ETM Architecture Specification*.
Current Sequencer State	0x067	0x19C	**CLK**	*ETM Architecture Specification*.
External Output Event 1-2	0x068, 0x069	0x1A0, 0x1A4	**CLK**	*ETM Architecture Specification*.
Extended External Input Selector	0x07B	0x1EC	**CLK**	*Extended External Input Selection Register* on page 3-28.

CoreSight Management registers

Table 3-6 shows the CoreSight Management registers.

Table 3-6 CoreSight Management registers

Name	Register number	Base offset	Clock domain	Description, see
Integration Mode Control	0x3C0	0xF00	**PCLKDBG**	*ETM Architecture Specification*
Claim Tag Set	0x3E8	0xFA0	**PCLKDBG**	*ETM Architecture Specification*
Claim Tag Clear	0x3E9	0xFA4	**PCLKDBG**	*ETM Architecture Specification*
Lock Access	0x3EC	0xFB0	**PCLKDBG**	*ETM Architecture Specification*
Lock Status	0x3ED	0xFB4	**PCLKDBG**	*ETM Architecture Specification*
Authentication Status	0x3EE	0xFB8	**PCLKDBG**	*ETM Architecture Specification*
Device Configuration	0x3F2	0xFC8	**PCLKDBG**	*ETM Architecture Specification*
Device Type	0x3F3	0xFCC	**PCLKDBG**	*ETM Architecture Specification*
Peripheral ID4 to 7	0x3F4 to 0x3F7	0xFD0 to 0xFDC	**PCLKDBG**	*Peripheral Identification Registers* on page 3-29
Peripheral ID0 to 3	0x3F8 to 0x3FB	0xFE0 to 0xFEC	**PCLKDBG**	
Component ID0 to 3	0x3FC to 0x3FF	0xFF0 to 0xFFC	**PCLKDBG**	*Component Identification Registers* on page 3-32

Integration Test registers

Table 3-7 shows the Integration Test registers.

Table 3-7 Integration Test registers

Name	Register number	Base offset	Clock domain	Description, see
ITETMIF	0x3B6	0xED8	**CLK**	*ITETMIF Register, ETM interface* on page 3-35
ITMISCOUT	0x3B7	0xEDC	**CLK**	*ITMISCOUT Register, miscellaneous outputs* on page 3-37
ITMISCIN	0x3B8	0xEE0	**CLK**	*ITMISCIN Register, miscellaneous inputs* on page 3-38
ITTRIGGERACK	0x3B9	0xEE4	**ATCLK**	*ITTRIGGERACK Register, trigger acknowledge* on page 3-39
ITTRIGGERREQ	0x3BA	0xEE8	**ATCLK**	*ITTRIGGERREQ Register, trigger request* on page 3-40
ITATBDATA0	0x3BB	0xEEC	**ATCLK**	*ITATBDATA0 Register, ATB data 0* on page 3-41
ITATBCTR2	0x3BC	0xEF0	**ATCLK**	*ITATBCTR2 Register, ATB control 2* on page 3-42
ITATBCTR1	0x3BD	0xEF4	**ATCLK**	*ITATBCTR1 Register, ATB control 1* on page 3-43
ITATBCTR0	0x3BE	0xEF8	**ATCLK**	*ITATBCTR0 Register, ATB control 0* on page 3-44

3.5 Descriptions of Implementation-defined registers

This section describes the following registers:

- *ETM Control Register*
- *Configuration Code Register* on page 3-22
- *ASIC Control Register* on page 3-24
- *ETM ID Register* on page 3-25
- *Configuration Code Extension Register* on page 3-26
- *Power-Down Status register* on page 3-29
- *Peripheral Identification Registers* on page 3-29
- *Component Identification Registers* on page 3-32
- *Integration Test Registers* on page 3-33
- *Extended External Input Selection Register* on page 3-28.

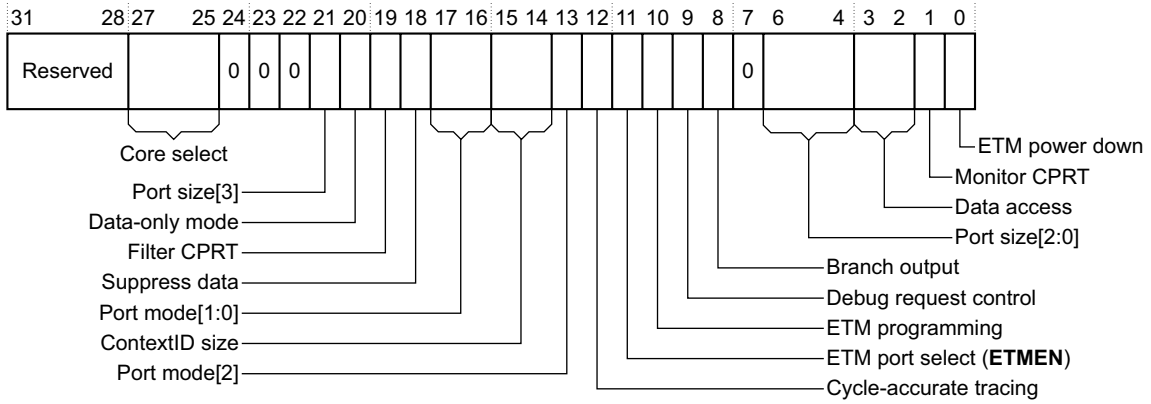
All other CoreSight ETM-R4 registers are described in the *ETM Architecture Specification*.

3.5.1 ETM Control Register

The ETM Control Register controls general operation of the ETM, such as whether tracing is enabled or coprocessor data is traced. It is:

- register `0x00`, at offset `0x00`
- a read/write register, with some bits that are read-only.

Figure 3-2 shows the bit assignments for the ETM Control Register:



See the register bit assignments table for more information about the RAZ bits, shown as 0.

Figure 3-2 ETM Control Register bit assignments

Table 3-8 shows the bit assignments for the ETM Control Register:

Table 3-8 ETM Control Register bit assignments

Bit	Function	Access	Description
[31:28]	Reserved	R/W	Must be written as 0.
[27:25]	Core select	R/W	If an ETM is shared between multiple cores, selects which core to trace. For the maximum value permitted, see bits [14:12] of the System Configuration Register. See the *ETM Architecture Specification* for more information. To guarantee that the ETM is correctly synchronized to the new core, you must update these bits as follows: 1. Set bit [10], ETM programming, and bit [0], ETM power down, to 1. 2. Change the core select bits. 3. Clear bit [0], ETM power down, to 0. 4. Perform other programming required as normal. On an ETM reset this field is zero.
[24]	Instrumentation resources access control	RO	ETM-R4 does not implement any instrumentation resources and therefore this bit is RAZ.
[23]	Disable software writes	RO	ETM-R4 does not support this feature and therefore this bit is RAZ.
[22]	Disable register writes from the debugger	RO	ETM-R4 does not support this feature and therefore this bit is RAZ.
[21]	Port size[3]	R/W	Use this bit in conjunction with bits [6:4]. On an ETM reset this bit is 0, corresponding to the 32-bit port size.
[20]	Data-only mode	R/W	The possible values of this bit are: **0** Instruction trace enabled. **1** Instruction trace disabled. Data-only tracing is possible in this mode. On an ETM reset this bit is 0.
[19]	Filter (CPRT)	R/W	Use this bit in conjunction with bit [1], the MonitorCPRT bit. For details see *Filter Coprocessor Register Transfers (CPRT) in ETMv3.0 and later* in the *ETM Architecture Specification*. On an ETM reset this bit is 0.
[18]	Suppress data	R/W	Use this bit with bit [7] to suppress data. For details see *Data suppression* in the *ETM Architecture Specification*. On an ETM reset this bit is 0.

Table 3-8 ETM Control Register bit assignments (continued)

Bit	Function	Access	Description
[17:16]	Port mode [1:0]	R/W	These bits are used, in conjunction with bit [13], to set the trace port clocking mode. ETM-R4 supports only dynamic mode, corresponding to the value b000, but you can write other values to these bits, and a read of the register returns the value written. Writing another value to these bits has no effect on the ETM. Bit [11] of the System Configuration Register indicates if these bits are set to select a supported clocking mode. On an ETM reset these bits are zero. For more information about trace port clocking modes see the *ETM Architecture Specification*.
[15:14]	ContextIDsize	R/W	The possible values of this field are: **b00** No Context ID tracing. **b01** Context ID bits [7:0] traced. **b10** Context ID bits [15:0] traced. **b11** Context ID bits [31:0] traced. **Note** Only the number of bytes specified is traced even if the new value is larger than this. On an ETM reset this field is zero.
[13]	Port mode[2]	R/W	See the description of bits [17:16]. On an ETM reset this bit is 0.
[12]	Cycle-accurate tracing	R/W	Set this bit to 1 if you want the trace to include a precise cycle count of executed instructions. This is achieved by adding extra information into the trace, giving cycle counts even when **TraceEnable** is inactive. On an ETM reset this bit is 0.
[11]	ETM port selection	R/W	This bit controls an external output, **ETMEN**. The possible values are: **0** **ETMEN** is LOW. **1** **ETMEN** is HIGH. You can use the **ETMEN** signal to control the routing of trace port signals to shared GPIO pins on your SoC, under the control of logic external to the ETM. Trace software tools must set this bit to 1 to ensure that trace output is enabled from this ETM. On an ETM reset this bit is 0.

Table 3-8 ETM Control Register bit assignments (continued)

Bit	Function	Access	Description
[10]	ETM programming	R/W	When set to 1, the ETM is being programmed. For more information, see *ETM Programming bit and associated state* in the *ETM Architecture Specification*. On an ETM reset this bit is set to b1.
[9]	Debug request control	R/W	If you set this bit to 1, when the trigger event occurs, the **DBGRQ** output is asserted until **DBGACK** is observed. This enables the ARM processor to be forced into Debug state. On an ETM reset this bit is 0.
[8]	Branch output	R/W	Set this bit to 1 if you want the ETM to output all branch addresses, even if the branch is because of a direct branch instruction. Setting this bit to 1 enables reconstruction of the program flow without having access to the memory image of the code being executed. On an ETM reset this bit is 0.
[7]	Stall processor	RO	ETM-R4 does not implement FIFOFULL stalling of the processor, and therefore this bit is RAZ.
[6:4]	Port size [2:0]	R/W	Use this field with bit [21] to specify the port size. The port size determines how many external pins are available to output the trace information on **ATDATA[31:0]**. ETM-R4 supports only the 32-bit port size, corresponding to a Port size[3:0] value of b0100, but you can write other values to these bits, and a read of the register returns the value written. Writing another value to these bits has no effect on the ETM. Bit [10] of the System Configuration Register indicates if these bits are set to select an unsupported port size. For more information see the *ETM Architecture Specification*. On an ETM reset this field is b100, corresponding to the 32-bit port size.
[3:2]	Data access	R/W	This field configures the data tracing mode. The possible values are: **b00** No data tracing. **b01** Trace only the data portion of the access. **b10** Trace only the address portion of the access. **b11** Trace both the address and the data of the access. On an ETM reset this field is zero.

Table 3-8 ETM Control Register bit assignments (continued)

Bit	Function	Access	Description
[1]	MonitorCPRT	R/W	This field controls whether CPRTs are traced. The possible values are: **0** CPRTs not traced. **1** CPRTs traced. This bit is used with bit [19]. For details see *Filter Coprocessor Register Transfers (CPRT) in ETMv3.0 and later* in the *ETM Architecture Specification*. On an ETM reset this bit is 0.
[0]	ETM power down	R/W	A pin controlled by this bit enables the ETM power to be controlled externally, see *Control of ETM power down*. The sense of this bit is inverted, and drives the **ETMPWRUP** signal. This bit must be cleared by the trace software tools at the beginning of a debug session. When this bit is set to 1, ETM tracing is disabled and accesses to any registers other than this register and the Lock Access Register are ignored. On an ETM reset this bit is set to 1.

Control of ETM power down

You can use the **ETMPWRUP** signal, controlled by the ETM power down bit of the ETM Control Register, to gate the clock to the logic in the ETM interface of the processor, to save power. Also, when you set the ETM power down bit to 1, the clock to most of the logic in the ETM macrocell is gated, disabling ETM tracing and leaving the ETM block operating in a low-power mode.

Note

You must not use the **ETMEN** signal to gate the ETM clock or any other functionality required for basic operation. You can use the ETMEN signal to control functionality that is required only for off-chip tracing, such as multiplexing between two ETMs. Use the **ETMPWRUP** signal to control basic operation of the ETM.

3.5.2 Configuration Code Register

The Configuration Code Register, `0x001` at offset `0x004`, is read-only. Figure 3-3 shows the arrangement of bits in the register.

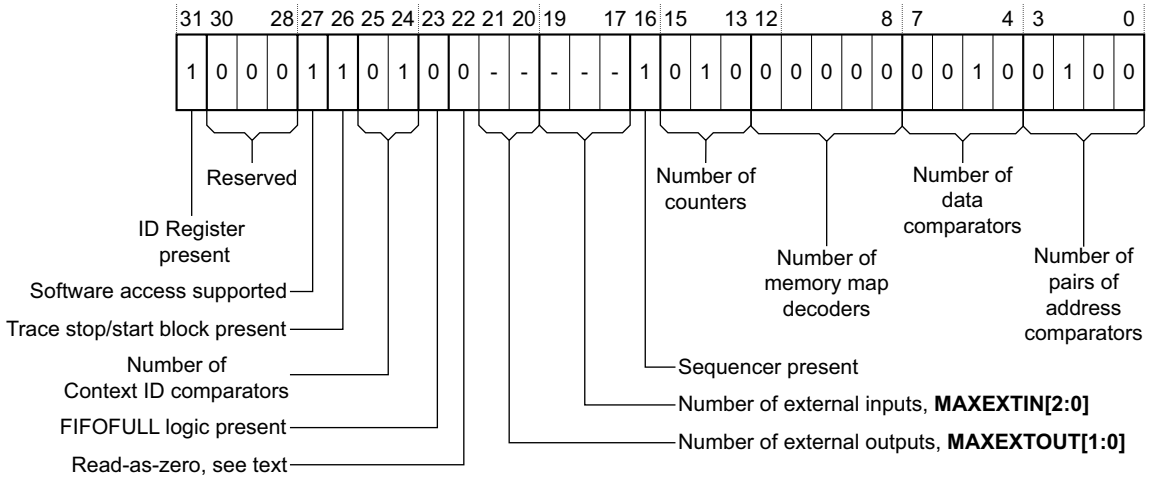


Figure 3-3 Configuration Code Register bit assignments

Table 3-9 lists how the bit values correspond with the register functions. If the **MAXEXTOUT[1:0]** and **MAXEXTIN[2:0]** signals are all tied LOW (0) the Configuration Code Register has the value `0x8D014024`.

Table 3-9 Configuration Code Register bit assignments

Bits	Value	Function
[31]	1	ETM ID Register present.
[30:28]	b000	Reserved. *Read-As-Zero* (RAZ).
[27]	1	Software access is supported.
[26]	1	Trace start/stop block is present.
[25:24]	1	Number of Context ID comparators.
[23]	0	**FIFOFULL** logic absent.
[22]	0	Reserved, Read-As-Zero. The *ETM Architecture Specification* defines this as the most significant bit of the Number of external outputs field, see the description of bits [21:20].

Table 3-9 Configuration Code Register bit assignments (continued)

Bits	Value	Function
[21:20]	-	Number of external outputs. Determined by the **MAXEXTOUT[1:0]** inputs. The value of these bits is the minimum of **MAXEXTOUT[1:0]** and 2, because CoreSight ETM-R4 supports a maximum of 2 external outputs.
[19:17]	-	Number of external inputs. Determined by the **MAXEXTIN[2:0]** inputs. The value of these bits is the minimum of **MAXEXTIN[2:0]** and 4, because CoreSight ETM-R4 supports a maximum of 4 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.

3.5.3 ASIC Control Register

The ASIC Control Register, `0x003` at offset `0x00C`, is a read/write register that is eight bits wide.

It is eight bits wide, and register bits [7:0] drive the **ASICCTL[7:0]** signal. Figure 3-4 shows the arrangement of bits in the register.

Figure 3-4 ASIC Control Register bit assignments

Table 3-10 lists how the bit values correspond with the register functions. After a reset the value of the register is Undefined.

Table 3-10 ASIC Control Register bit assignments

Bits	Function
[31:8]	Reserved.
[7:0]	**ASICCTL[7:0]**: when a bit in this field is set to 0 the corresponding bit of **ASICCTL[7:0]** is LOW when a bit in this field is set to 1 the corresponding bit of **ASICCTL[7:0]** is HIGH.

3.5.4 ETM ID Register

The ETM ID Register has value `0x4104F231`, and is read-only. Figure 3-5 shows the arrangement of bits in the register.

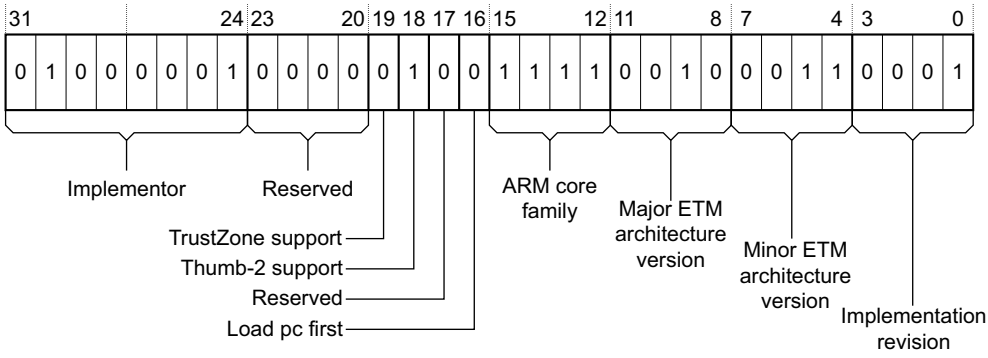


Figure 3-5 ETM ID Register bit assignments

Table 3-11 lists how the bit values correspond with the register functions in the macrocell. The ETM ID Register has the value `0x4104F23x`, where x depends on the release version of the macrocell, see the Implementation revision field description in the table for more information.

Table 3-11 ETM ID Register bit assignments

Bit numbers	Value	Function
[31:24]	0x41	Implementor = A (for ARM).
[23:20]	b0000	Reserved.
[19]	0	Security Extensions support. This bit is set to 1 if the processor supports the ARMv7 architecture Security Extensions. On the macrocell, this bit is not set (=0), meaning that the ETM behaves as if the processor is in Secure state at all times.
[18]	1	Thumb-2 support. This bit is set to 1 if the processor supports the Thumb-2 architectural extensions. On the macrocell, this bit is set to 1, meaning that all 32-bit Thumb instructions are traced as a single instruction, including BL and BLX immediate.
[17]	0	Reserved.

Table 3-11 ETM ID Register bit assignments (continued)

Bit numbers	Value	Function
[16]	0	If set to 1, load PC first. On the macrocell, this bit is not set (=0), meaning that on an LSM[a] load operation with the PC included in the load list, the PC is *not* loaded first.
[15:12]	b1111	ARM processor family. The value of b1111 means that the processor family is defined elsewhere.
[11:8]	b0010	Major ETM architecture version number. A value of 0 in this field indicates ETMv1. For ETM v3.x, this field = 2.
[7:4]	b0011	Minor ETM architecture version number. For ETM vx.3, this field = 3.
[3:0]	b0001	Implementation revision. Value given is for the r1p0 release of the macrocell. For release r0p0 the value is b0000.

a. See the *ETM Architecture Specification* for a definition and list of LSM operations.

3.5.5 Configuration Code Extension Register

The Configuration Code Extension Register, register `0x07A` at offset `0x1E8`, is read-only. Figure 3-6 shows the arrangement of bits in the register.

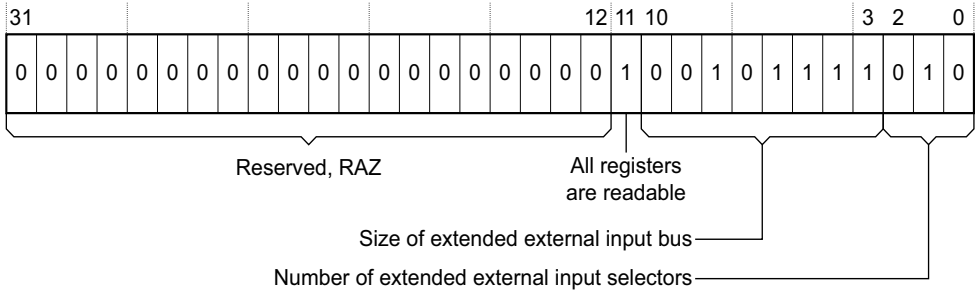


Figure 3-6 Configuration Code Extension Register bit assignments

Table 3-12 lists how the bit values correspond with the register functions. The Configuration Code Extension Register has the value `0x0000097A`.

Table 3-12 Configuration Code Extension Register bit assignments

Bit numbers	Value	Function
[31:12]	0	Reserved, RAZ.
[11]	1	All registers, except some integration test registers, are readable. See Table 3-7 on page 3-16 for details of the access to integration test registers[a].
[10:3]	47	Size of extended external input bus.
[2:0]	2	Number of extended external input selectors.

a. Registers with names that start with IT are the Integration Test Registers, for example ITATBCTR1.

3.5.6 Extended External Input Selection Register

The Extended External Input Selection Register specifies the extended external inputs, see the *ETM Architecture Specification* for more information.

It is:

- register `0x7B`, at offset `0x1EC`
- a read/write register.

Figure 3-7 shows the bit assignments for the Extended External Input Selection Register:

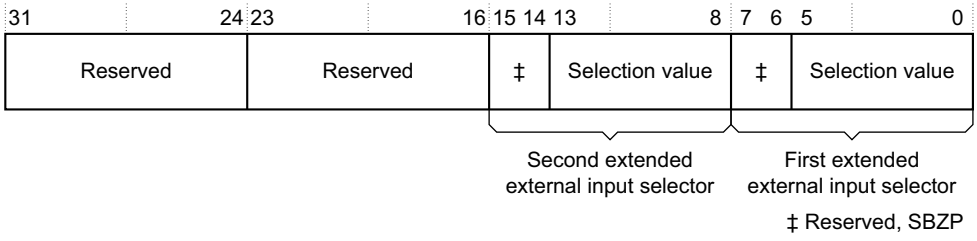


Figure 3-7 Extended External Input Selection Register bit assignments

Table 3-13 shows the bit assignments for the Extended External Input Selection Register:

Table 3-13 Extended External Input Selection Register bit assignments

Bits	Description
[31:15]	Reserved, SBZP.
[15:8]	Second extended external input selector: **Bits [15,14]** Reserved, SBZP. **Bits [13:8]** Selection value for second external input.
[7:0]	First extended external input selector: **Bits [7,6]** Reserved, SBZP. **Bits [5:0]** Selection value for first external input.

3.5.7 Power-Down Status register

The *Power-Down Status register* (PDSR) indicates the power-down status of the ETM. It is:

- register `0xC5`, at offset `0x314`
- a read-only register.

Figure 3-8 shows the arrangement of bits in the register.

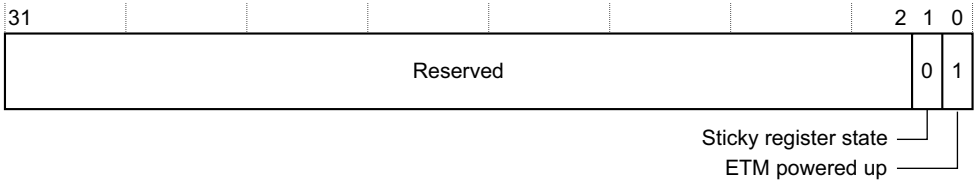


Figure 3-8 Power-Down Status register bit assignments

Table 3-14 shows the PDSR bit assignments:

Table 3-14 Power-Down Status register bit assignments

Bit numbers	Value	Function
[31:2]	0	Reserved, RAZ
[1]	0	Sticky Register State. ETM-R4 does not support multiple power domains so this bit is RAZ.
[0]	1	ETM Powered Up. The ETM Trace registers are accessible. ETM-R4 does not support multiple power domains so this bit is RAO.

3.5.8 Peripheral Identification Registers

The Peripheral Identification Registers are a set of eight read-only registers, PeripheralID7 to PeripheralID0. These registers are defined in the *ETM Architecture Specification*. This specifies that only bits [7:0] of each register are used. This means that the eight Peripheral Identification Registers can be considered to define a single 64-bit *Peripheral ID*, as Figure 3-9 on page 3-30 shows.

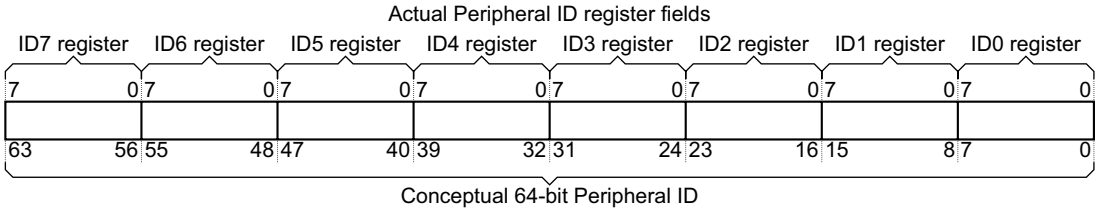


Figure 3-9 Mapping between the Peripheral ID Registers and the Peripheral ID value

Figure 3-10 shows the Peripheral ID fields in the single conceptual Peripheral ID register.

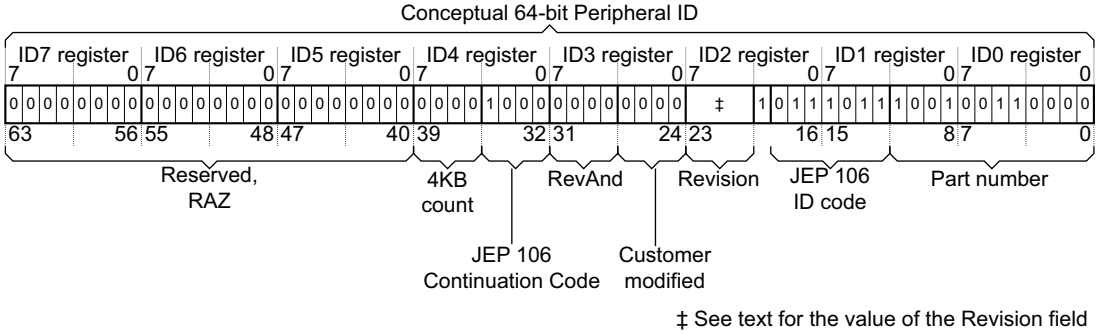


Figure 3-10 Peripheral ID fields

Table 3-15 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 3-15 Peripheral Identification Registers, bit assignments

Register	Register number	Register offset	Bit	Value	Description
PeripheralID7	0x3F7	0xFDC	[31:8]	-	Unused, read undefined.
			[7:0]	0x00	Reserved for future use, RAZ.
PeripheralID6	0x3F6	0xFD8	[31:8]	-	Unused, read undefined.
			[7:0]	0x00	Reserved for future use, RAZ.
PeripheralID5	0x3F5	0xFD4	[31:8]	-	Unused, read undefined.
			[7:0]	0x00	Reserved for future use, RAZ.

Table 3-15 Peripheral Identification Registers, bit assignments (continued)

Register	Register number	Register offset	Bit	Value	Description
PeripheralID4	0x3F4	0xFD0	[31:8]	-	Unused, read undefined.
			[7:4]	0x0	n, where 2^n is number of 4KB blocks used.
			[3:0]	0x4	JEP 106 continuation code.
PeripheralID3	0x3FB	0xFEC	[31:8]	-	Unused, read undefined.
			[7:4]	0x0	RevAnd (at top level). Manufacturer revision number.
			[3:0]	0x0	Customer Modified. 0x0 indicates from ARM.
PeripheralID2	0x3FA	0xFE8	[31:8]	-	Unused, read undefined.
			[7:4]	a	Revision Number of Peripheral. This value is the same as the Implementation revision field of the ETM ID register, see *ETM ID Register* on page 3-25.
			[3]	1	Always 1. Indicates that a JEDEC assigned value is used.
			[2:0]	b011	JEP 106 identity code [6:4].
PeripheralID1	0x3F9	0xFE4	[31:8]	-	Unused, read undefined.
			[7:4]	b0001	JEP 106 identity code [3:0]
			[3:0]	0x9	Part Number [11:8]. Upper *Binary Coded Decimal* (BCD) value of Device Number.
PeripheralID0	0x3F8	0xFE0	[31:8]	-	Unused, read undefined.
			[7:0]	0x30	Part Number [7:0]. Middle and Lower BCD value of Device Number.

a. See the Description column for details.

Note

In Table 3-15 on page 3-30, the *Peripheral Identification Registers* on page 3-29 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 3-16 on page 3-32 the *Component Identification Registers* on page 3-32 are listed in order of register name, from most significant (ID3) to least significant (ID0).

3.5.9 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*.

Figure 3-11 shows this concept of a single 32-bit component ID, obtained from the four Component Identification Registers.

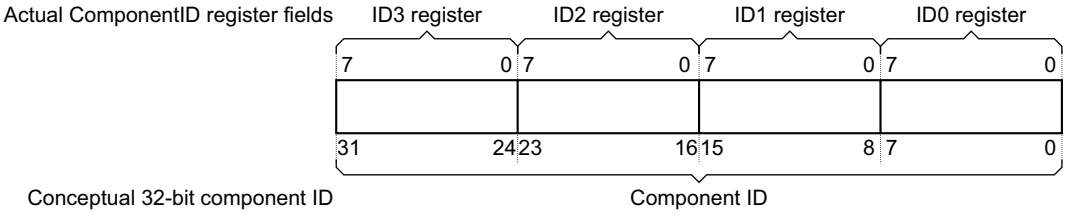


Figure 3-11 Mapping between the Component ID Registers and the Component ID value

Table 3-16 shows the values of the fields when reading the Component Identification Registers. This, again, shows how the valid fields combine to give the component ID. This register structure is defined in the *ETM Architecture Specification*.

Table 3-16 Component Identification Registers, bit assignments

Register	Register number	Register offset	Bit	Value	Description
ComponentID3	0x3FF	0xFFC	[31:8]	-	Unused, read undefined.
			[7:0]	0xB1	Component identifier, bits [31:24].
ComponentID2	0x3FE	0xFF8	[31:8]	-	Unused, read undefined.
			[7:0]	0x05	Component identifier, bits [23:16].
ComponentID1	0x3FD	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	0x3FC	0xFF0	[31:8]	-	Unused, read undefined.
			[7:0]	0x0D	Component identifier, bits [7:0].

3.5.10 Integration Test Registers

The following subsections 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 3-17 shows the signals that 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 3-18 on page 3-34 shows the signals that can be read in this way.

See the *ETM Architecture Specification* for more information. The Integration Mode Control Register is described in the *ETM Architecture Specification*.

Table 3-17 Output signals that the Integration Test Registers can control

Signal	Register	Bit	Register description
AFREADY	ITATBCTR0	[1]	See *ITATBCTR0 Register, ATB control 0* on page 3-44
ATBYTES[1:0]	ITATBCTR0	[9:8]	See *ITATBCTR0 Register, ATB control 0* on page 3-44
ATDATA[31, 23, 15, 7, 0]	ITATBDATA0	[4:0]	See *ITATBDATA0 Register, ATB data 0* on page 3-41
ATID[6:0]	ITATBCTR1	[6:0]	See *ITATBCTR1 Register, ATB control 1* on page 3-43
ATVALID	ITATBCTR0	[0]	See *ITATBCTR0 Register, ATB control 0* on page 3-44
ETMDBGRQ	ITMISCOUT	[4]	See *ITMISCOUT Register, miscellaneous outputs* on page 3-37
EXTOUT[1:0]	ITMISCOUT	[9:8]	See *ITMISCOUT Register, miscellaneous outputs* on page 3-37
nETMWFIREADY	ITMISCOUT	[5]	See *ITMISCOUT Register, miscellaneous outputs* on page 3-37
TRIGGER	ITTRIGGERREQ	[0]	See *ITTRIGGERREQ Register, trigger request* on page 3-40

Table 3-18 Input signals that the Integration Test Registers can read

Signal	Register	Bit	Register description
AFVALID	ITATBCTR2	[1]	See *ITATBCTR2 Register, ATB control 2* on page 3-42
ATREADY	ITATBCTR2	[0]	See *ITATBCTR2 Register, ATB control 2* on page 3-42
DBGACK	ITMISCIN	[4]	See *ITMISCIN Register, miscellaneous inputs* on page 3-38
ETMCID[31, 0]	ITETMIF	[11:10]	See *ITETMIF Register, ETM interface* on page 3-35
ETMDA[31, 0]	ITETMIF	[7:6]	See *ITETMIF Register, ETM interface* on page 3-35
ETMDCTL[11, 0]	ITETMIF	[5:4]	See *ITETMIF Register, ETM interface* on page 3-35
ETMDD[63, 0]	ITETMIF	[9:8]	See *ITETMIF Register, ETM interface* on page 3-35
ETMIA[31, 1]	ITETMIF	[3:2]	See *ITETMIF Register, ETM interface* on page 3-35
ETMICTL[13, 0]	ITETMIF	[1:0]	See *ITETMIF Register, ETM interface* on page 3-35
ETMWFIPENDING	ITMISCIN	[5]	See *ITMISCIN Register, miscellaneous inputs* on page 3-38
EVNTBUS[46, 28, 0]	ITETMIF	[14:12]	See *ITETMIF Register, ETM interface* on page 3-35
EXTIN[3:0]	ITMISCIN	[3:0]	See *ITMISCIN Register, miscellaneous inputs* on page 3-38
TRIGGERACK	ITTRIGGERACK	[0]	See *ITTRIGGERACK Register, trigger acknowledge* on page 3-39

Using the Integration Test Registers

The *CoreSight ETM-R4 Integration Manual* gives a full description of the use of the Integration Test Registers to check integration. In brief:

- When bit [0] of the Integration Mode Control Register is set to 1, values written to the write-only integration test registers map onto the specified outputs of the macrocell. For example, writing `0x3` to ITMISCOUT[9:8] causes **EXTOUT[1:0]** to take the value `0x3`.
- When bit [0] of the Integration Mode Control Register is set to 1, values read from the read-only integration test registers correspond to the values of the specified inputs of the macrocell. For example, if you read ITMISCIN[3:0] you obtain the value of **EXTIN[3:0]**.

When bit [0] of the Integration Mode Control Register is set to 0:

- Reading an Integration Test Register returns an Unpredictable value.
- The effect of attempting to write to an Integration Test Register, other than the read-only Integration Test Registers, is Unpredictable.

Note

You must not attempt to write to an Integration Test Register unless you have set bit [0] of the Integration Mode Control Register to 1.

See the *ETM Architecture Specification* for details of the Integration Mode Control Register.

ITETMIF Register, ETM interface

The ITETMIF Register, `0x3B6` at offset `0xED8`, is read-only. Figure 3-12 shows the assignment of bits in the register.

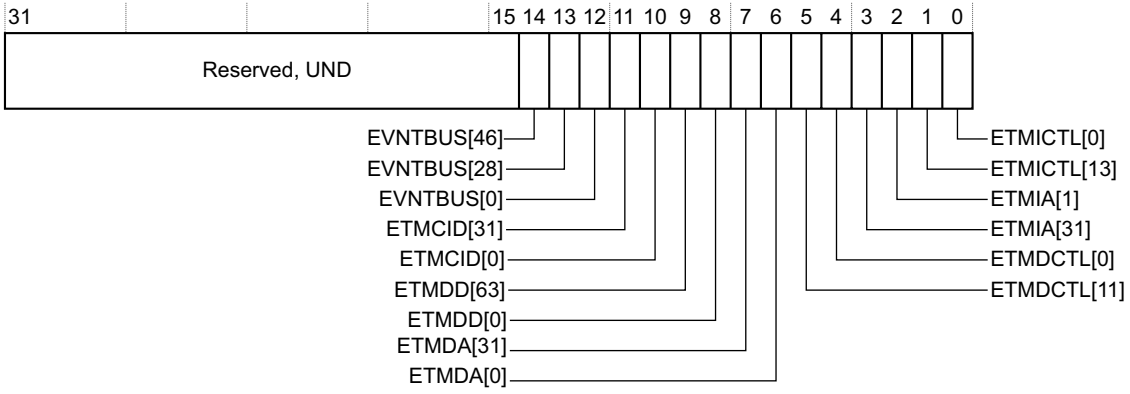


Figure 3-12 ITETMIF Register bit assignments

Table 3-21 on page 3-38 lists how the bit values correspond with the register functions. The value of these fields depend on the signals on the input pins when the register is read.

Table 3-19 ITETMIF Register bit assignments

Bits	Name	Function
[31:15]	-	Reserved. Read undefined.
[14]	EVNTBUS[46]	Returns the value of the **EVNTBUS[46]** input pin[a].
[13]	EVNTBUS[28]	Returns the value of the **EVNTBUS[28]** input pin[a].
[12]	EVNTBUS[0]	Returns the value of the **EVNTBUS[0]** input pin[a].
[11]	ETMCID[31]	Returns the value of the **ETMCID[31]** input pin[a].
[10]	ETMCID[0]	Returns the value of the **ETMCID[0]** input pin[a].
[9]	ETMDD[63]	Returns the value of the **ETMDD[63]** input pin[a].
[8]	ETMDD[0]	Returns the value of the **ETMDD[0]** input pin[a].
[7]	ETMDA[31]	Returns the value of the **ETMDA[31]** input pin[a].
[6]	ETMDA[0]	Returns the value of the **ETMDA[0]** input pin[a].
[5]	ETMDCTL[11]	Returns the value of the **ETMDCTL[11]** input pin[a].
[4]	ETMDCTL[0]	Returns the value of the **ETMDCTL[0]** input pin[a].
[3]	ETMIA[31]	Returns the value of the **ETMIA[31]** input pin[a].
[2]	ETMIA[1]	Returns the value of the **ETMIA[1]** input pin[a].
[1]	ETMICTL[13]	Returns the value of the **ETMICTL[13]** input pin[a].
[0]	ETMICTL[0]	Returns the value of the **ETMICTL[0]** input pin[a].

a. When a bit is set to 0, the corresponding input pin is LOW.
When a bit is set to 1, the corresponding input pin is HIGH.
The Integration Test Register bits values always correspond to the physical state of the input pins.

ITMISCOUT Register, miscellaneous outputs

The ITMISCOUT Register, register `0x3B7` at offset `0xEDC`, is write-only. Figure 3-13 shows the arrangement of bits in the register.

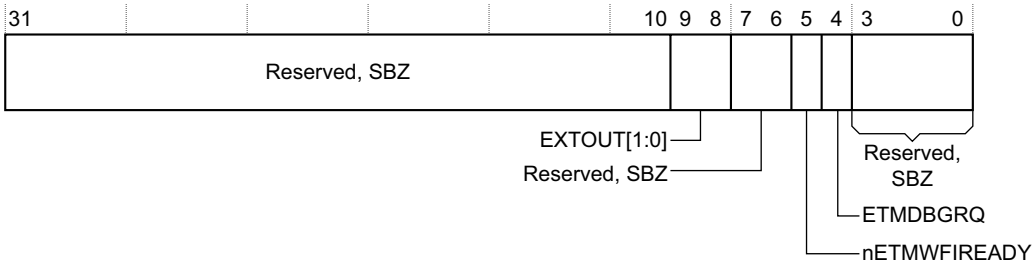


Figure 3-13 ITMISCOUT Register bit assignments

Table 3-20 lists how the bit values correspond with the register functions.

Table 3-20 ITMISCOUT Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Write as zero.
[9:8]	EXTOUT	Drives the **EXTOUT[1:0]** output pins[a].
[7:6]	-	Reserved. Write as zero.
[5]	ETMWFIREADY	Drives the **nETMWFIREADY** output pin[a].
[4]	ETMDBGRQ	Drives the **ETMDBGRQ** output pin[a].
[3:0]	-	Reserved. Write as zero.

a. When a bit is set to 0, the corresponding output pin is LOW.
When a bit is set to 1, the corresponding output pin is HIGH.
The Integration Test Register bits values correspond to the physical state of the output pins.

ITMISCIN Register, miscellaneous inputs

The ITMISCIN Register, register `0x3B8` at offset `0xEE0`, is read-only. Figure 3-14 shows the arrangement of bits in the register.

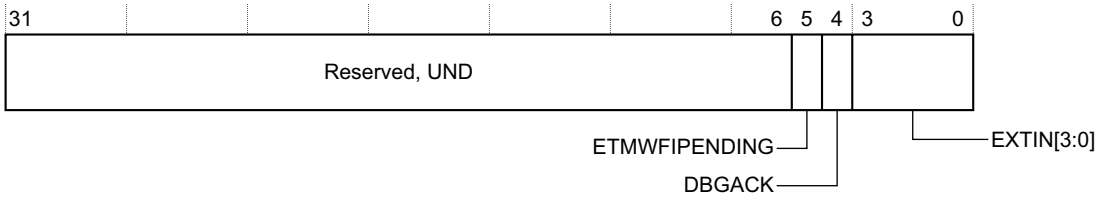


Figure 3-14 ITMISCIN Register bit assignments

Table 3-21 lists how the bit values correspond with the register functions. The values of these fields depend on the signals on the input pins when the register is read.

Table 3-21 ITMISCIN Register bit assignments

Bits	Name	Function
[31:6]	-	Reserved. Read undefined.
[5]	ETMWFIPENDING	Returns the value of the **ETMWFIPENDING** input pin[a].
[4]	DBGACK	Returns the value of the **DBGACK** input pin[a].
[3:0]	EXTIN	Returns the value of the **EXTIN[3:0]** input pins[a].

a. When a bit is set to 0, the corresponding input pin is LOW.
When a bit is set to 1, the corresponding input pin is HIGH.
The Integration Test Register bits values always correspond to the physical state of the input pins.

ITTRIGGERACK Register, trigger acknowledge

The ITTRIGGERACK Register, register `0x3B9` at offset `0xEE4`, is read-only. Figure 3-15 shows the arrangement of bits in the register.

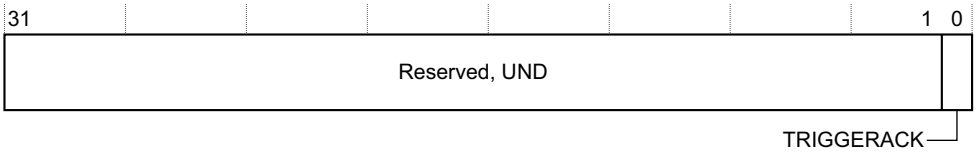


Figure 3-15 ITTRIGGERACK Register bit assignments

Table 3-22 lists how the bit values correspond with the register functions. The value of this field depends on the signal on the input pins when the register is read.

Table 3-22 ITTRIGGERACK Register bit assignments

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

a. When a bit is set to 0, the corresponding input pin is LOW.
When a bit is set to 1, the corresponding input pin is HIGH.
The Integration Test Register bits values always correspond to the physical state of the input pins.

ITTRIGGERREQ Register, trigger request

The ITTRIGGERREQ Register, register `0x3BA` at offset `0xEE8`, is write-only. Figure 3-16 shows the arrangement of bits in the register.

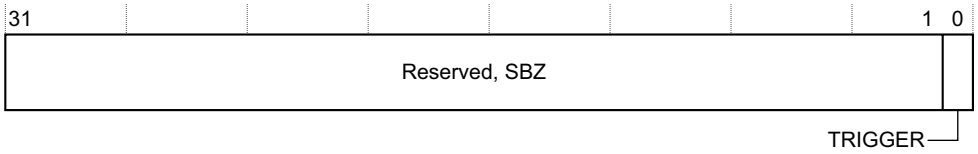


Figure 3-16 ITTRIGGERREQ Register bit assignments

Table 3-23 lists how the bit values correspond with the register functions.

Table 3-23 ITTRIGGERREQ Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved. Write as zero.
[0]	TRIGGER	Drives the **TRIGGER** output pin[a].

a. When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The Integration Test Register bits values always correspond to the physical state of the output pins.

ITATBDATA0 Register, ATB data 0

The ITATBDATA0 Register, register `0x3BB` at offset `0xEEC`, is write-only. Figure 3-17 shows the arrangement of bits in the register.

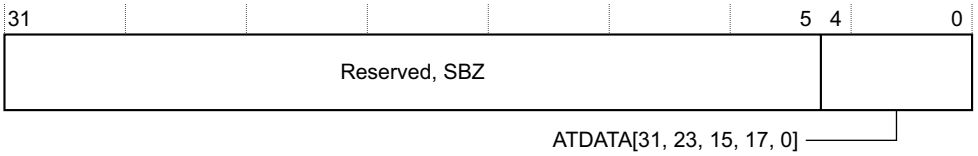


Figure 3-17 ITATBDATA0 Register bit assignments

Table 3-24 lists how the bit values correspond with the register functions.

Table 3-24 ITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	-	Reserved. Write as zero.
[4:0]	ATDATA	Drives the **ATDATA[31, 23, 15, 7, 0]** output pins[a].

a. When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The Integration Test Register bits values always correspond to the physical state of the output pins.

ITATBCTR2 Register, ATB control 2

The ITATBCTR2 Register, register `0x3BC` at offset `0xEF0`, is read-only. Figure 3-18 shows the arrangement of bits in the register.

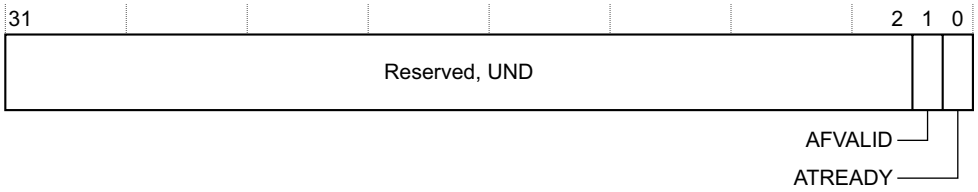


Figure 3-18 ITATBCTR2 Register bit assignments

Table 3-25 lists how the bit values correspond with the register functions. The values of these fields depend on the signals on the input pins when the register is read.

Table 3-25 ITATBCTR2 Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved. Read undefined.
[1]	AFVALID	Returns the value of the **AFVALID** input pin[a].
[0]	ATREADY	Returns the value of the **ATREADY** input pin[a].

a. When a bit is set to 0, the corresponding input pin is LOW.
When a bit is set to 1, the corresponding input pin is HIGH.
The Integration Test Register bits values always correspond to the physical state of the input pins.

ITATBCTR1 Register, ATB control 1

The ITATBCTR1 Register, register `0x3BD` at offset `0xEF4`, is write-only. Figure 3-19 shows the arrangement of bits in the register.

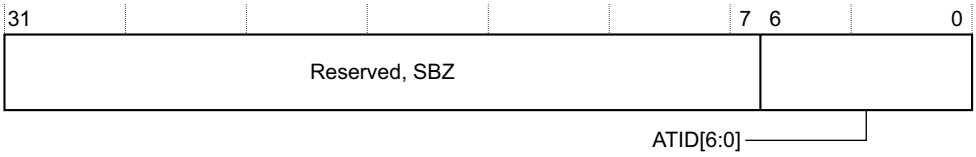


Figure 3-19 ITATBCTR1 Register bit assignments

Table 3-26 lists how the bit values correspond with the register functions.

Table 3-26 ITATBCTR1 Register bit assignments

Bits	Name	Function
[31:7]	-	Reserved. Write as zero.
[6:0]	ATID	Drives the **ATID[6:0]** output pins[a].

a. When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The Integration Test Register bits values always correspond to the physical state of the output pins.

ITATBCTR0 Register, ATB control 0

The ITATBCTR0 Register, register `0x3BE` at offset `0xEF8`, is write-only. Figure 3-20 shows the arrangement of bits in the register.

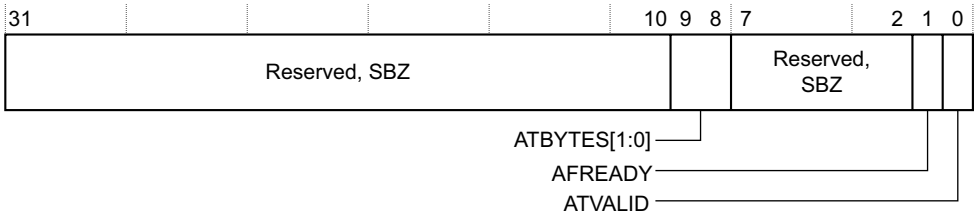


Figure 3-20 ITATBCTR0 Register bit assignments

Table 3-27 lists how the bit values correspond with the register functions.

Table 3-27 ITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	-	Reserved. Write as zero.
[9:8]	ATBYTES	Drives the **ATBYTES[1:0]** output pins[a].
[7:2]	-	Reserved. Write as zero.
[1]	AFREADY	Drives the **AFREADY** output pin[a].
[0]	ATVALID	Drives the **ATVALID** output pin[a].

a. When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The Integration Test Register bits values always correspond to the physical state of the output pins.

Appendix A
Signals Lists

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

- *ETMR4 Signals* on page A-2.

A.1 ETMR4 Signals

Table A-1 shows the ETMR4 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 ETM-R4 Integration Manual* for information about signals and connectivity.

Table A-1 ETMR4 signals

Signal	Input/output	Description	Clock domain
AFREADY	Output	ATB interface FIFO flush finished.	**ATCLK**
AFVALID	Input	ATB interface FIFO flush request.	**ATCLK**
ASICCTL[7:0]	Output	Contents of the ASIC control register, ASICCTL.	**CLK**
ATBYTES[1:0]	Output	Size of **ATDATA**.	**ATCLK**
ATCLK	Input	ATB interface clock.	-
ATCLKEN	Input	Enable signal for **ATCLK**.	**ATCLK**
ATDATA[31:0]	Output	ATB interface data.	**ATCLK**
ATID[6:0]	Output	ATB interface trace source ID.	**ATCLK**
ATREADY	Input	**ATDATA** can be accepted.	**ATCLK**
ATVALID	Output	ATB interface data valid.	**ATCLK**
CLK	Input	This is the main clock for the ETMR4.	-
CORESELECT[2:0]	Output	Where an ETM is shared between multiple cores, this signal specifies which core to trace. The value appears as bits [14:12] of the System Configuration Register.	**CLK**
DBGACK	Input	Indicates that the core is in debug state. This signal 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 interface between the core and the ETM.	**CLK**

Table A-1 ETMR4 signals (continued)

Signal	Input/output	Description	Clock domain
DBGEN	Input	Invasive debug enable. When HIGH (1), indicates that invasive debug is enabled. If this signal and **NIDEN** are both LOW, **CLK** and **ATCLK** are both stopped and no trace is generated.	-
ETMCID[31:0]	Input	Current value of the processor Context ID Register.	**CLK**
ETMDA[31:0]	Input	Address for data transfer.	**CLK**
ETMDBGRQ	Output	Request from the macrocell 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**
ETMDD[63:0]	Input	Contains the data value for a Load, Store, `MRC`, or `MCR` instruction.	**CLK**
ETMDCTL[11:0]	Input	Data control signals.	**CLK**
ETMEN	Output	Enable signal for trace output from the ETM, driven by bit [11] of the ETM Control Register.	**CLK**
ETMIA[31:1]	Input	Address for executed instruction.	**CLK**
ETMICTL[13:0]	Input	Instruction control signals.	**CLK**
ETMPWRUP	Output	When HIGH, indicates that the macrocell is in use. When LOW: • external logic supporting the macrocell can be clock-gated to conserve power • the Cortex-R4 processor disables the interface • logic within the macrocell is clock-gated to conserve power.	**CLK**
ETMWFIPENDING	Input	Indicates that the Cortex-R4 processor is about to go into Standby mode, and that the ETM must drain its FIFO.	**CLK**

Table A-1 ETMR4 signals (continued)

Signal	Input/output	Description	Clock domain
EVNTBUS[46:0]	Input	Gives the status of the performance monitoring events. Used as extended external inputs.	**CLK**
EXTIN[3:0]	Input	External input resources.	**CLK**
EXTOUT[1:0]	Output	External outputs.	**CLK**
FIFOPEEK[6:0]	Output	For validation purposes only. Indicates when various events occur before being written to the FIFO.	**CLK**
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. These signals determine the value of bits [14:12] of the System Configuration register, see the footnote to Table 3-1 on page 3-7.	**CLK**
MAXEXTIN[2:0]	Input	Number of external inputs supported by the ASIC (maximum 4). These signals determine the value bits [19:17] in the Configuration Code Register, see *Configuration Code Register* on page 3-22.	**CLK**
MAXEXTOUT[1:0]	Input	Number of external outputs supported by the ASIC (maximum 2). These signals determine the value bits [22:20] in the Configuration Code Register, see *Configuration Code Register* on page 3-22.	**CLK**
nETMWFIREADY	Output	Indicates that the macrocell FIFO is empty and that the Cortex-R4 processor can be put into Standby mode.	**CLK**
NIDEN	Input	Non-invasive debug enable. When HIGH (1), indicates that non-invasive debug is enabled. If this signal and **DBGEN** are both LOW, **CLK** and **ATCLK** are both stopped and no trace is generated.	-
PADDRDBG[11:2]	Input	Debug APB Address Bus.	**PCLKDBG**

Table A-1 ETMR4 signals (continued)

Signal	Input/output	Description	Clock domain
PADDRDBG31	Input	Originates as an output signal from the *Debug Access Port* (DAP): • **PADDRDBG31** at logic 1 indicates an access from hardware (JTAG) • **PADDRDBG31** at logic 0 indicates an access from software.	**PCLKDBG**
PCLKDBG	Input	Debug APB clock.	-
PCLKENDBG	Input	Debug APB clock enable.	**PCLKDBG**
PENABLEDBG	Input	The Debug APB interface is enabled for a transfer.	**PCLKDBG**
PRDATADBG[31:0]	Output	Debug APB read data.	**PCLKDBG**
PREADYDBG	Output	Used to extend Debug APB transfers.	**PCLKDBG**
PRESETDBGn	Input	Debug APB interface reset. Resets all registers.	Internally synchronized
PSELDBG	Input	Debug APB slave select signal.	**PCLKDBG**
PWDATADBG[31:0]	Input	Debug APB write data.	**PCLKDBG**
PWRITEDBG	Input	Debug APB transfer direction, !Read/Write.	**PCLKDBG**
RSTBYPASS	Input	Reset synchronization bypass DFT signal.	-
SE	Input	Scan enable DFT signal	-
nSYSPORESET	Input	Power-on (main) reset. Resets all registers.	Internally synchronized
TRIGGER	Output	Trigger request status signal.	**ATCLK**
TRIGGERACK	Input	ATB trigger acknowledge.	Internally synchronized
TRIGSBYPASS	Input	Trigger synchronization bypass.	**ATCLK**

A.1.1 The trigger signals

The **TRIGSBYPASS**, **TRIGGER**, and **TRIGGERACK** signals control trigger behavior and indicate when a trigger occurs.

TRIGSBYPASS controls whether asynchronous registering and handshaking is performed:

- When **TRIGSBYPASS** is HIGH, **TRIGGER** is asserted for one **ATCLK** cycle, and **TRIGGERACK** is ignored.
- When **TRIGSBYPASS** is LOW, **TRIGGER** is asserted, and is held until **TRIGGERACK** is asserted. When **TRIGGERACK** is asserted, **TRIGGER** is de-asserted.

TRIGGERACK is double-registered to **ATCLK** inside the ETM macrocell.

Appendix B
I/O Signal Timings

This appendix describes the macrocell I/O timing. It contains the following section:

- *ETMR4 I/O timing parameters* on page B-2.

B.1 ETMR4 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 based on these delays:

Early The delay is less than 20% of the period.

Middle The delay is between 20% and 80% of the period.

Late The delay is greater than 80% of the period.

Table B-1 describes the ETMR4 signal timing parameters.

Table B-1 ETMR4 signal timing parameters

Signal name	Timing classification	Input/Output
AFREADY	Middle	Output
AFVALID	Middle	Input
ASICCTL[7:0]	Middle	Output
ATBYTES[1:0]	Middle	Output
ATCLK	-	Input
ATCLKEN	Middle	Input
ATDATA[31:0]	Middle	Output
ATID[6:0]	Middle	Output
ATREADY	Middle	Input
ATVALID	Middle	Output
CLK	-	Input
CORESELECT[2:0]	Middle	Output
DBGACK	Middle	Input
DBGEN	Middle	Input
ETMCID[31:0]	Middle	Input
ETMDA[31:0]	Middle	Input

Table B-1 ETMR4 signal timing parameters (continued)

Signal name	Timing classification	Input/Output
ETMDBGRQ	Middle	Output
ETMDD[63:0]	Middle	Input
ETMDCTL[11:0]	Middle	Input
ETMEN	Middle	Output
ETMIA[31:1]	Middle	Input
ETMICTL[13:0]	Middle	Input
ETMPWRUP	Middle	Output
ETMWFIPENDING	Middle	Input
EVNTBUS[46:0]	Middle	Input
EXTIN[3:0]	Middle	Input
EXTOUT[1:0]	Middle	Output
FIFOPEEK[6:0]	Middle	Output
MAXCORES[2:0]	Middle	Input
MAXEXTIN[2:0]	Middle	Input
MAXEXTOUT[1:0]	Middle	Input
nETMWFIREADY	Middle	Output
NIDEN	Middle	Input
PADDRDBG[11:2]	Middle	Input
PADDRDBG31	Middle	Input
PCLKDBG	Middle	Input
PCLKENDBG	Middle	Input
PENABLEDBG	Middle	Input
PRDATADBG[31:0]	Middle	Output
PREADYDBG	Late	Output
PRESETDBGn	Late	Input

Table B-1 ETMR4 signal timing parameters (continued)

Signal name	Timing classification	Input/Output
PSELDBG	Middle	Input
PWDATADBG[31:0]	Middle	Input
PWRITEDBG	Middle	Input
RSTBYPASS	Middle	Input
SE	Middle	Input
nSYSPORESET	Middle	Input
TRIGGER	Middle	Output
TRIGGERACK	Middle	Input
TRIGSBYPASS	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.

Appendix C
Typical APB Transfers

The macrocell implements APBv3.0. This appendix describes typical APB read and write transfers, and contains the following sections:

C.1 Typical APB write transfers

Two types of write transfer are described in this section:

- *APB write transfer with no wait states*
- *APB write transfer with wait states* on page C-3.

C.1.1 APB write transfer with no wait states

Figure C-1 shows a basic write transfer with no wait states. The transfer starts with the address, write data, write signal, and select signal changing after the rising edge of the clock at T1. This is called the set-up cycle.

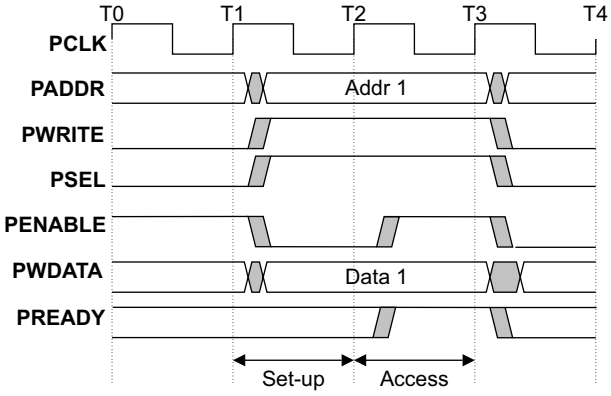


Figure C-1 APB write transfer with no wait states

After the following clock edge at T2, the enable signal, **PENABLE**, is asserted. This indicates that the access cycle is taking place. The address, data, and control signals remain valid throughout the access cycle and the transfer completes at the end of this cycle.

The enable signal, **PENABLE**, is deasserted at the end of the transfer. The select signal, **PSELx**, also goes LOW unless the transfer is to be followed immediately by another transfer to the same peripheral.

Note

PSELx indicates that the slave device is selected and that a data transfer is required.There is a **PSELx** signal for each slave.

C.1.2 APB write transfer with wait states

Figure C-2 shows how the **PREADY** signal from the slave can extend the transfer. During an access cycle, when **PENABLE** is HIGH, the transfer can be extended by driving **PREADY** LOW. The following signals remain unchanged for the additional cycles:

- address, **PADDR**
- write signal, **PWRITE**
- select signal, **PSEL**
- enable signal, **PENABLE**
- write data, **PWDATA**.

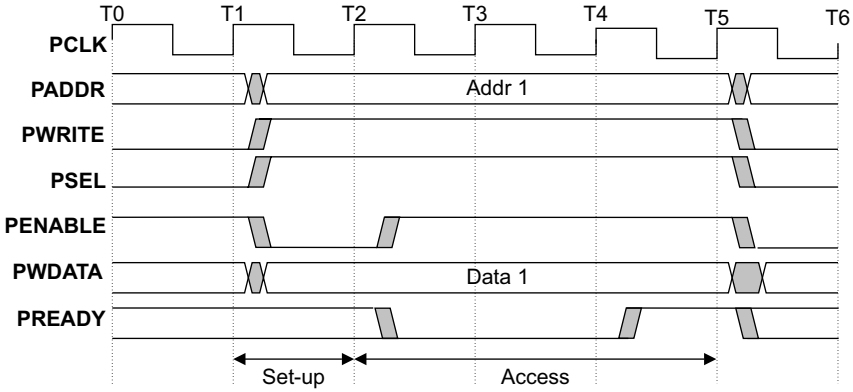


Figure C-2 APB write transfer with wait states

PREADY can take any value when **PENABLE** is LOW. This ensures that peripherals that have a fixed two cycle access can tie **PREADY** HIGH.

Note

It is recommended that the address and write signals are not changed immediately after a transfer but remain stable until another access occurs. This reduces power consumption.

C.2 Typical APB read transfers

Two types of read transfer are described in this section:

- *APB read transfers with no wait states*
- *APB read transfer with wait states*.

C.2.1 APB read transfers with no wait states

Figure C-3 shows a read transfer. The timing of the address, write, select and enable signals are all the same as those for write transfers as shown in Figure C-1 on page C-2. The slave must provide the data before the end of the read transfer.

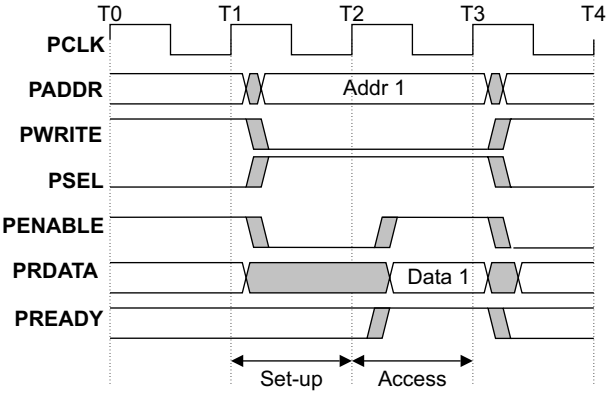


Figure C-3 APB read transfer with no wait states

C.2.2 APB read transfer with wait states

Figure C-4 on page C-5 shows how the **PREADY** signal can extend the transfer. The transfer is extended if **PREADY** is driven LOW during an Access cycle. The protocol ensures that the following remain unchanged for the additional cycles:

- address, **PADDR**
- write signal, **PWRITE**
- select signal, **PSEL**
- enable signal, **PENABLE**.

Figure C-4 on page C-5 shows that two cycles are added using the **PREADY** signal. However, any number of additional cycles, from zero upwards can be added.

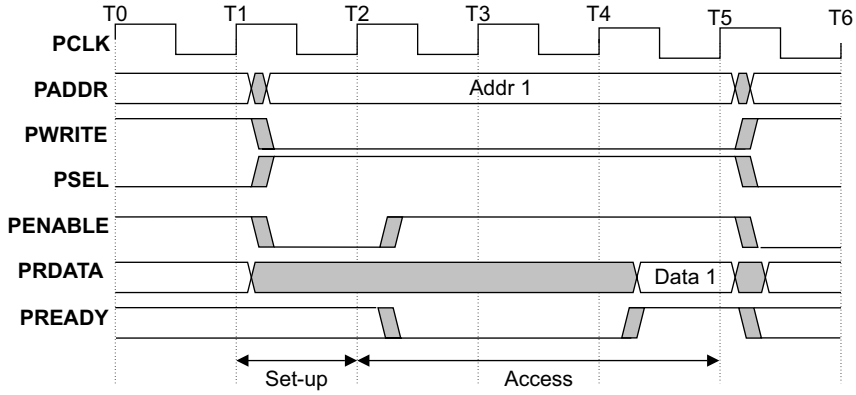


Figure C-4 APB read transfer with wait states

C.3 APB operating states

Figure C-5 shows the operational activity of the APB.

The state machine operates through the following states:

IDLE This is the default state of the APB.

SETUP When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, **PSELx**, is asserted. The bus only remains in the SETUP state for one clock cycle and always moves to the ACCESS state on the next rising edge of the clock.

ACCESS The enable signal, **PENABLE**, is asserted in the ACCESS state. The address, write, and select signals all remain stable during the transition from the SETUP to ACCESS state. The **PREADY** signal from the slave controls the exit from the ACCESS state:

- If **PREADY** is held LOW by the slave then the peripheral bus remains in the ACCESS state.
- If **PREADY** is driven HIGH by the slave then the ACCESS state is exited and the bus returns to the IDLE state if no more transfers are required. Alternatively, the bus moves directly to the SETUP state if another transfer follows.

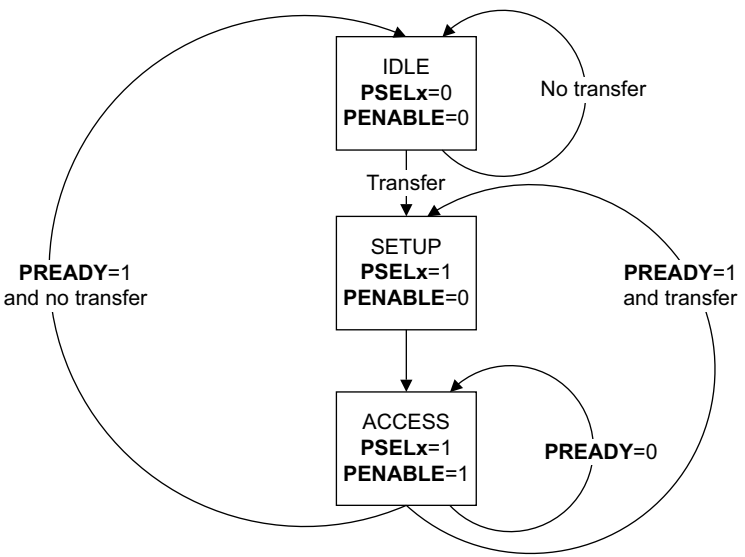


Figure C-5 Debug APB state diagram

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.

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 the 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.

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.