ETM11RV™

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0233B

ETM11RV
Technical Reference Manual

Release Information

Change history

Date	Issue	Change
December 2002	A	Released
August 2003	B	VHDL updates, Table 1-1 corrected. 3.9 Data instructions in Java state added, 3.10 added.Preface and Index updated.

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
ETM11RV Technical Reference Manual

Preface

About this book x
Feedback xiv

Chapter 1 Introduction

1.1 About the ETM11RV 1-2
1.2 ETM11RV configuration 1-3

Chapter 2 Programmer's Model

2.1 About the programmer's model 2-2
2.2 Programming and reading ETM registers 2-4

Chapter 3 Implementation-defined Behavior

3.1 ETM architecture version 3-2
3.2 Values of ETM11RV read-only registers 3-3
3.3 Precise TraceEnable events 3-6
3.4 Parallel instruction execution 3-7
3.5 Independent load/store unit 3-8
3.6 Context ID tracing 3-10
3.7 Interaction with the performance monitoring unit 3-11
3.8 Reset behavior 3-12
3.9 Data instructions in Java state 3-13

3.10 Restrictions and limitations 3-14

Appendix A **Signal Descriptions**

A.1 Signal descriptions A-2

Appendix B **AC Parameters**

B.1 Timing diagrams and timing parameters B-2
B.2 Reset timing parameters B-3
B.3 Trace interface signals B-4
B.4 Trace port timing parameters B-6
B.5 Debug timing parameters B-8
B.6 TAP interface timing parameters B-9
B.7 Control signal timing parameters B-10
B.8 ETM interface timing parameters B-12
B.9 Design for test (DFT) timing parameters B-14
B.10 External I/O timing parameters B-15

Glossary

List of Tables
ETM11RV Technical Reference Manual

	Change history	ii
Table 1-1	ETM11RV resources	1-3
Table 3-1	ETM ID Register fields	3-3
Table 3-2	ETM11RV Configuration Code Register	3-4
Table 3-3	ETM11RV Configuration Code Extension Register	3-5
Table 3-4	Conditions for observing address comparator sticky bits	3-8
Table 3-5	Bytecodes and amounts of trace	3-13
Table A-1	ETM11RV signals	A-2
Table B-1	Trace interface timing parameters	B-4
Table B-2	Trace port timing parameters	B-6
Table B-3	Debug timing parameters	B-8
Table B-4	TAP interface timing parameters	B-9
Table B-5	Control signal timing parameters	B-10
Table B-6	ETM interface timing parameters	B-12
Table B-7	DFT timing parameters	B-14
Table B-8	External input and output timing parameters	B-15

List of Figures
ETM11RV Technical Reference Manual

	Key to timing diagram conventions	xii
Figure 1-1	ETM11RV functional blocks	1-2
Figure 2-1	Programming ETM registers	2-3
Figure 2-2	ETM JTAG structure	2-5
Figure B-1	Trace interface timing parameters	B-5
Figure B-2	Trace port timing parameters	B-7
Figure B-3	Debug timing parameters	B-8
Figure B-4	TAP interface timing parameters example	B-9
Figure B-5	Control signal timing parameters	B-11
Figure B-6	ETM interface timing parameters	B-13
Figure B-7	Scan chain timing parameters	B-14
Figure B-8	External I/O timing parameters	B-16

Preface

This preface introduces the ARM11™ *Embedded Trace Macrocell RealView®* ETM11RV Revision: r0p1 Technical Reference Manual. It contains the following sections:

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

About this book

This book is the technical reference manual for the ETM11RV r0p1.

Product revision status

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

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

Intended audience

This document has been written for the following target audiences:

- Designers of development tools providing support for ETM functionality. Implementation-specific behaviour is described in this document. These users should also consult the *ETM Architecture Specification* (ARM IHI 0014).
- Hardware and software engineers integrating the ETM11RV into an ASIC that includes an ARM1136J-S or ARM1136JF-S processor. These users should also consult the *ETM11RV User Guide* (ARM DUI 0223).

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the ETM11RV.

Chapter 2 *Programmer's Model*

Read this chapter for information on how to program the registers that control the ETM11RV.

Chapter 3 *Implementation-defined Behavior*

Read this chapter for additional implementation-specific information relating to ETM11RV.

Appendix A *Signal Descriptions*

This appendix gives a list of ETM11RV signals.

Appendix B *AC Parameters*

This appendix gives a list of AC parameters for all ETM11RV signals.

Conventions

This section describes the conventions that this manual uses:

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

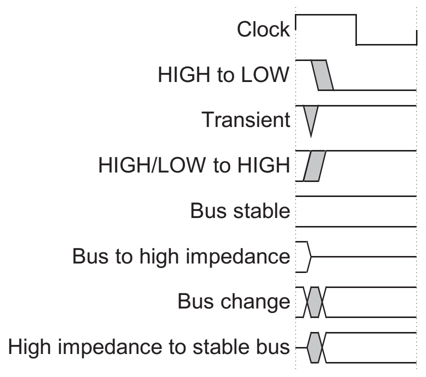
Typographical

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 ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` (`mono` underlined)	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	denotes language keywords when used outside example code.
< and >	Angle brackets enclose replaceable terms for assembler syntax where they appear in code or code fragments. They appear in normal font in running text. For example: • `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>` • The Opcode_2 value selects which register is accessed.

Timing diagrams

This manual contains one or more timing diagrams. The figure named *Key to timing diagram conventions* on page xii explains the components used in these diagrams. When variations occur they have clear labels. You must not assume any timing information that is not explicit in the diagrams.



Key to timing diagram conventions

Signal naming

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means HIGH for active-HIGH signals and LOW for active-LOW signals:

Prefix A	Denotes *Advanced eXtensible Interface* (AXI) global and 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 n	Denotes Active-LOW signals except in the case of AHB or *Advanced Peripheral Bus* APB reset signals. These are named **HRESETn** and **PRESETn** respectively.
Prefix P	Denotes an APB signal.
Prefix R	Denotes AXI read channel signals.
Prefix W	Denotes AXI write channel signals.

Numbering

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

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

- 'h7B4 is an unsized hexadecimal value.

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

Further reading

This section lists publications from both ARM Limited and third parties that provide additional information on developing code for the ARM family of processors.

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

ARM publications

This document contains information that is specific to the ETM11RV. Read the following manuals for additional information:

- *Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014)
- *ETM11RV Implementation Guide* (ARM DII 0061)
- *ETM11RV User Guide* (ARM DUI 0223)
- *ETB11 Implementation Guide* (ARM DII 067)
- *ETB11 Technical Reference Manual* (ARM DDI 0275)
- *ARM1136JF-S and ARM1136J-S Technical Reference Manual* (ARM DDI 0211)
- *ARM1136JF-S and ARM1136J-S Implementation Guide* (ARM DII 0022)
- *ARM1136JF-S and ARM1136J-S Test Chip Implementation Guide* (ARM DXI 0144)
- *Multi-ICE User Guide* (ARM DUI 0048).

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

- *ARM Reference Peripherals Specification* (ARM DDI 0062)
- the ARM datasheet or technical reference manual for your hardware device.

Feedback

ARM Limited welcomes feedback on both the ETM11RV, and its documentation.

Feedback on the ETM11RV

If you have any problems with the ETM11RV, contact your supplier. To help us provide a rapid and useful response, please give:

- details of the release you are using
- details of the platform you are running on, such as the hardware platform, operating system type and version
- a small standalone sample of code that reproduces the problem
- a clear explanation of what you expected to happen, and what actually happened
- the commands you used, including any command-line options
- sample output illustrating the problem
- the version string of the tool, including the version number and date.

Feedback on this book

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

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

General suggestions for additions and improvements are also welcome.

Chapter 1
Introduction

This chapter introduces the ETM11RV r0p1. It contains the following sections:

- *About the ETM11RV* on page 1-2
- *ETM11RV configuration* on page 1-3.

1.1 About the ETM11RV

The ETM11RV provides instruction trace and data trace for the ARM1136J-S and ARM1136JF-S microprocessors. Figure 1-1 shows the main functional blocks of ETM11RV.

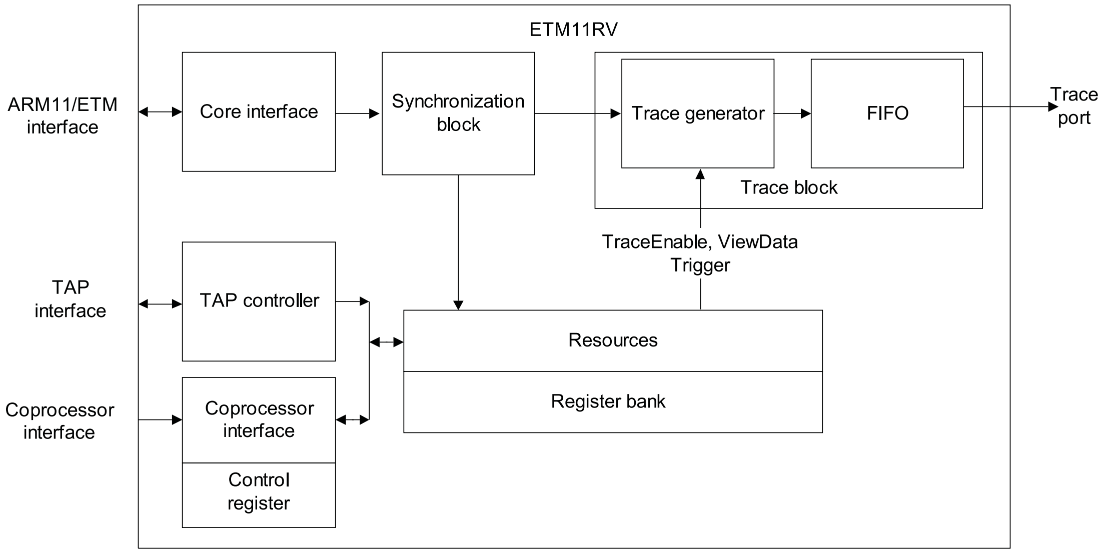


Figure 1-1 ETM11RV functional blocks

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

For information about ETM11RV signals, see Appendix A *Signal Descriptions*.

1.2 ETM11RV configuration

The ETM11RV resources are shown in Table 1-1.

Table 1-1 ETM11RV resources

Resource description	Number/size
Context ID comparators	1
External inputs	4
External outputs	2
Sequencers	1
Counters	2
Memory map decoders	0
Data comparators	2
Pairs of address comparators	4
Extended external inputs	20
Extended external input comparators	2
FIFO size	69 bytes
Trace packet width[a]	4/8/16/24/32 bits

a. Software-selectable using the ETM Control Register.

Chapter 2
Programmer's Model

This chapter describes the mechanisms for programming the registers used to set up the trace and triggering facilities of the ETM11RV. ETM11RV supports software access in addition to JTAG TAP access. It contains the following sections:

- *About the programmer's model* on page 2-2
- *Programming and reading ETM registers* on page 2-4
- *Coprocessor access* on page 2-6.

2.1 About the programmer's model

When programming the ETM registers you must enable all the changes at the same time. For example, if the counter is reprogrammed, it might start to count based on incorrect events, before the trigger condition has been correctly set up. In addition, the JTAG clock, **TCLK**, can often be asynchronous to the core clock.

You can use the ETM programming bit in the ETM Control Register to disable all operations during programming. To do this follow the procedure shown in Figure 2-1 on page 2-3.

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

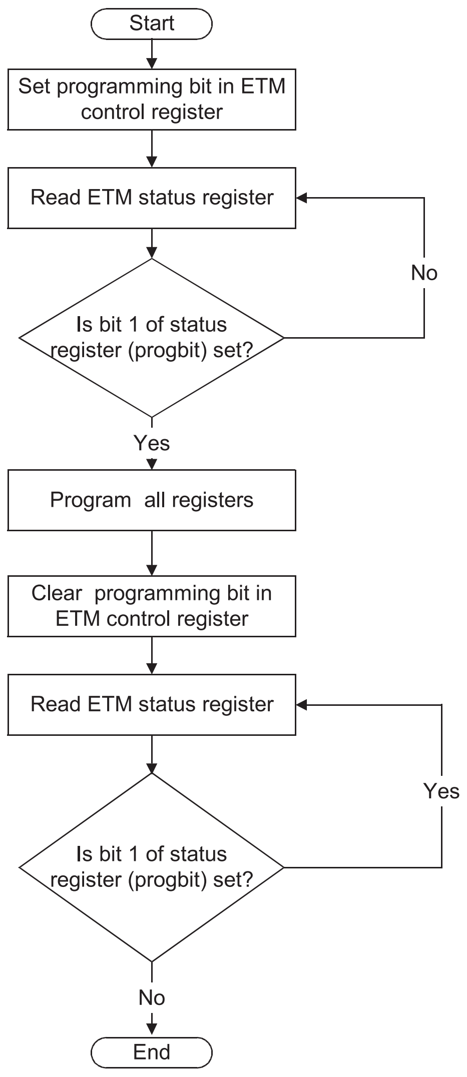

Start
Set programming bit in ETM control register
Read ETM status register
Is bit 1 of status register (progbit) set?
No
Yes
Program all registers
Clear programming bit in ETM control register
Read ETM status register
Is bit 1 of status register (progbit) set?
Yes
No
End

Figure 2-1 Programming ETM registers

It is not necessary for the core to be in debug state while the registers are being programmed.

2.2 Programming and reading ETM registers

There are two methods of access to the ETM11RV registers:

- *JTAG access*
- *Coprocessor access* on page 2-6.

2.2.1 JTAG access

All registers in the ETM are programmed using a JTAG interface. The interface is an extension of the ARM TAP controller, and is assigned scan chain number 6. The scan chain consists of a 40-bit shift register comprising:

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

All registers in the ETM can be configured using the JTAG interface. The general arrangement of the ETM JTAG registers is shown in Figure 2-2 on page 2-5.

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

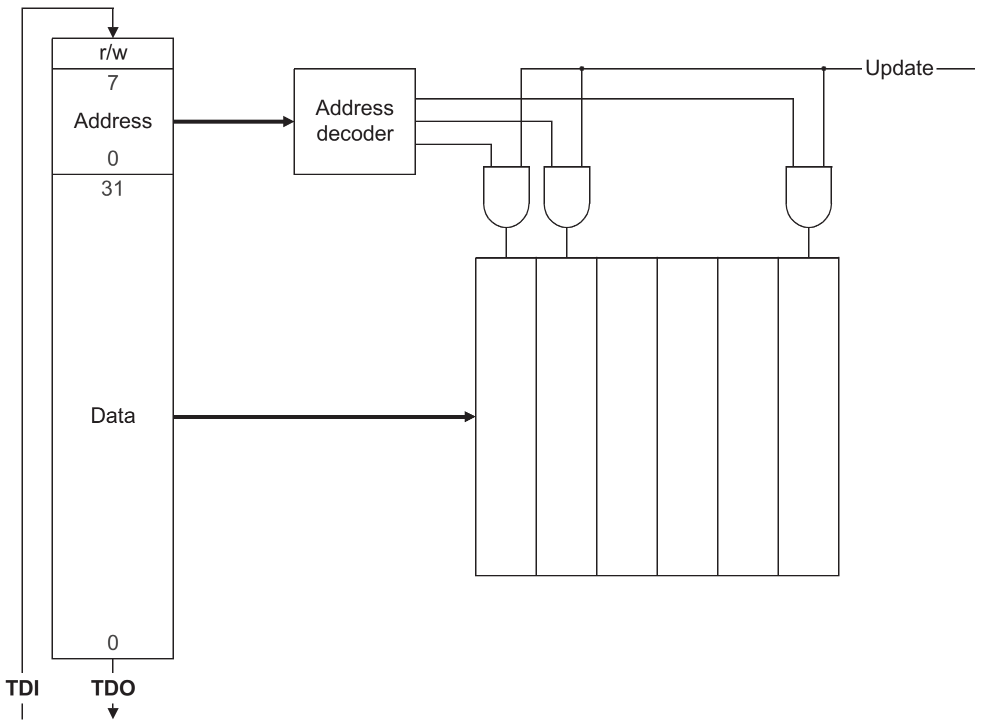


Figure 2-2 ETM JTAG structure

Note

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

Restricting JTAG access

JTAG access to the ETM registers can be made read-only by setting bit 22 of the ETM control register, `0x00`. This bit can only be set from the coprocessor interface.

2.2.2 Coprocessor access

ETM11RV uses ETMv3.1, which provides improved coprocessor access. This enables you to use the ETM as an extended breakpoint unit to test for unit failure while testing multiple devices. The improved coprocessor access also means that you do not have to program each device individually by connecting a probe. You can now do the following without external hardware:

- program the ETM
- collect trace
- examine the contents of the ETB11.

This section describes the changes to the programmer's model:

- *Coprocessor model*
- *Concurrent access* on page 2-7.

Coprocessor model

The software access method is processor-architecture defined. If you are using a non-ARM architecture then you must specify your own method for software access.

All registers of the ETM are mapped to coprocessor 14. All instructions in this coprocessor with Opcode1 equal to 1 are reserved for ETM use.

The instructions to read and write ETM registers are as follows:

```
MRC p14, 1, Rd, 0, reg[3:0], reg[6:4]
MCR p14, 1, Rd, 0, reg[3:0], reg[6:4]
```

These instructions have CRn equal to 0 and the register number encoded in Opcode2 and CRm.

Unlike coprocessor accesses to debug registers, a read from a non-existent register returns an Unpredictable value, and a write to a non-existent register silently fails. No Undefined Instruction exceptions are ever generated. It is up to the tools to determine what registers are valid from the programmers' model.

The read-only, write-only, and read/write status of the registers is the same as for JTAG access.

Restricting coprocessor access

Coprocessor access to the ETM registers can be made read-only by setting bit 23 of the ETM control register, `0x00`. This bit can only be set from the JTAG interface.

Concurrent access

Concurrent access from the JTAG and coprocessor interfaces is permitted. Conflicting JTAG accesses are delayed by a cycle.

Chapter 3
Implementation-defined Behavior

This chapter contains implementation-specific information relating to the ETM11RV. It contains the following sections:

- *ETM architecture version* on page 3-2
- *Values of ETM11RV read-only registers* on page 3-3
- *Precise TraceEnable events* on page 3-6
- *Parallel instruction execution* on page 3-7
- *Independent load/store unit* on page 3-8
- *Context ID tracing* on page 3-10
- *Interaction with the performance monitoring unit* on page 3-11
- *Reset behavior* on page 3-12
- *Data instructions in Java state* on page 3-13
- *Restrictions and limitations* on page 3-14.

3.1 ETM architecture version

ETM11RV implements version 3.1 of the ETM architecture, ETMv3.1.

3.2 Values of ETM11RV read-only registers

This section describes the following read-only registers:

- *ETM ID Register*
- *ETM11RV Configuration Code Register* on page 3-4
- *ETM11RV Configuration Code Extension Register* on page 3-5.

3.2.1 ETM ID Register

Table 3-1 shows the value of the fields when reading the ETM ID Register in the ETM11RV. The ETM ID Register has the value of `0x41013211`.

Table 3-1 ETM ID Register fields

Bit numbers	Value	Meaning
[31:24]	`0x41`	Implementor = A (ARM Limited).
[23:18]	b000000	Reserved.
[17]	b0	CPRTs count as data instructions.
[16]	b1	Load pc first. Special handling is not required to reconstruct the data addresses of an LDM with the pc in the register list because ETMv3.1 requires noncontiguous data addresses to be traced. However, special handling is required to determine which transfers correspond to which register.
[15:12]	b0011	ARM core family = ARM11.
[11:8]	b0010	Major ETM architecture version number = 3.
[7:4]	b0001	Minor ETM architecture version number = 1.
[3:0]	b0001	Implementation revision.

3.2.2 ETM11RV Configuration Code Register

Table 3-2 shows the value of the fields when reading the ETM11RV Configuration Code Register. The configuration code register has the value `0x85294024`.

Table 3-2 ETM11RV Configuration Code Register

Bit numbers	Value	Meaning
[31]	1	ETM11RV ID Register present
[30:28]	b000	Not used
[27]	1	Software access is supported
[26]	1	Trace start/stop block is present
[25:24]	1	Number of Context ID comparators
[23]	1	**FIFOFULL** logic present
[22:20]	2	Number of external outputs
[19:17]	4	Number of external inputs
[16]	1	The sequencer is present
[15:13]	2	Number of counters
[12:8]	0	Number of memory map decoders
[7:4]	2	Number of data comparators
[3:0]	4	Number of pairs of address comparators

3.2.3 ETM11RV Configuration Code Extension Register

Table 3-3 shows the value of the fields when reading the ETM11RV Configuration Code Extension Register.

Table 3-3 ETM11RV Configuration Code Extension Register

Bit numbers	Value	Meaning
[31:12]	0	Reserved
[11]	1	All registers are readable.
[10:3]	20	Size of extended external input bus.
[2:0]	2	Number of extended external input selectors.

3.3 Precise TraceEnable events

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

- single address comparators
- address range comparators.

3.4 Parallel instruction execution

ARM11 cores support *branch folding*, where correctly predicted branches are executed in parallel with the following instruction. The ETM11RV is therefore capable of tracing two instructions per cycle, although only the second can have data associated with it.

While the trace start/stop block is calculated for each instruction as required, the ETM11RV is not capable of tracing one instruction without the other. In particular, if a folded branch is traced, the instruction it is paired with is also traced, along with any data associated with it if **ViewData** is active.

3.5 Independent load/store unit

ARM1136JF-S and ARM1136J-S cores have independent load/store units. This means that they are capable of continuing to transfer data values for an earlier instruction after later instructions have been executed. In these circumstances, the later instructions are said to have executed underneath the data instruction. When a data instruction has been traced, all instructions executed underneath it are also traced.

Note

This does not create much additional trace, because the instructions executed underneath cannot be data instructions or indirect branches.

All address comparators have a sticky bit that is observed when the comparator is used as shown in Table 3-4.

Table 3-4 Conditions for observing address comparator sticky bits

Comparator usage	Single address comparator	Address range comparator
Selected as an event	Sticky bit ignored	Sticky bit observed
Selected by **TraceEnable**	Sticky bit ignored	Sticky bit ignored
Selected by start/stop block	Sticky bit ignored	Sticky bit ignored
Selected by **ViewData**	Sticky bit observed	Sticky bit observed

Where it is observed, the sticky bit behaves as follows:

Sticky bit set	When the comparator matches the instruction address of a data instruction.
Sticky bit cleared	When the data instruction completes.

The sticky bit is never set for address comparators configured for data addresses.

Where observed, the comparator continues to match while the sticky bit is set.

Single address comparators cannot observe the sticky bit when selected as an event, because they are defined to be active for only one cycle for the benefit of the counters and sequencer. If an event of the kind `instruction address = X AND data address = Y` is required, the instruction address comparator must be an address range comparator to guarantee that it still matches when the data address comparator matches.

ViewData always observes the sticky bits so that if an instruction address comparator is selected, all the data corresponding to that instruction match. In normal circumstances, you must not use single instruction address comparators as the enabling event, because they do not observe the sticky bit.

There is no requirement for **TraceEnable** or the start/stop block to observe the sticky bit, because when a data instruction has been traced, tracing cannot be disabled until all data transfers corresponding to that instruction have occurred.

Note

Using data values to create an event, such as a sequencer transition, might result in out-of-order events occurring because the load data might be returned out of order. If you are concerned that the ARM1136JF-S or ARM1136J-S nonblocking cache might affect programmed events, you can disable it in the core by writing to bit 21 of the cp15 Control Register (r1). See the *ARM1136JF-S and ARM1136J-S Technical Reference Manual* for more information.

3.6 Context ID tracing

The ETM11RV detects the MCR instruction that changes the context ID and traces the appropriate number of bytes as a context ID packet instead of a normal data packet. As a result, if context ID tracing is enabled, an MCR instruction that changes the context ID does not have its data traced separately.

3.7 Interaction with the performance monitoring unit

ARM1136JF-S and ARM1136J-S processors include a performance monitoring unit that enables events, such as cache misses and instructions executed, to be counted over a period of time. These events are all available for use by ETM11RV using the extended external input facility. Each bit in the **EVNTBUS[19:0]** input is mapped to the corresponding extended external input. See the *ARM1136JF-S and ARM1136J-S 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 ARM1136JF-S or ARM1136J-S performance monitoring unit can count the two ETM11RV external outputs as additional events. These events are not provided back to ETM11RV as extended external inputs.These facilities allow the system events to be further qualified with ETM resources, such as instruction address ranges or the start/stop resource, before being passed back to the performance monitoring unit for counting. This can be done as follows:

- Configure the ETM extended external input selectors to the system events you wish to count.
- Configure the desired ETM filtering resource as appropriate.
- Configure the ETM external outputs to extended external input selector AND the desired ETM filtering resource.
- Select the ETM external outputs as the events to be counted in the ARM1136JF-S or ARM1136J-S performance monitoring unit.

3.8 Reset behavior

The ETM Control Register, `0x00`, is reset only on a power-on reset, **nPORESET** LOW.

The TAP ID register and scan chain select register, updated by the SCAN_N TAP instruction, are reset by driving **DBGnTRST** LOW or by taking the TAP state machine through the Test-Logic-Reset state.

3.9 Data instructions in Java state

Table 3-5 gives the amount of trace expected for each data bytecode, which is Implementation Defined but common to ETM10RV and ETM11RV.

Table 3-5 Bytecodes and amounts of trace

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

Table 3-5 assumes the bytecode is implemented in hardware. The Jazelle specification allows any bytecode to be implemented in software, and normal ARM state data trace is output instead in this case. This applies to aastore in current implementations.

3.10 Restrictions and limitations

Refer to your ARM CPU TRM for any restrictions or limitations on what can be traced.

Appendix A
Signal Descriptions

This appendix describes the signals used in the ETM11RV. It contains the following section:

- *ETM11RV signals* on page A-2.

A.1 Signal descriptions

The ETM11RV signals are described in Table A-1.

Table A-1 ETM11RV signals

Signal name	Input/ output	Description
ASICCTL[7:0]	Output	Contents of the ASIC control register.
CLK	Input	Main clock, identical to the processor clock.
DBGACK	Input	Indicates that the core is in debug state. This is connected to the core general purpose **DBGACK** output, so that it can be used to know when **ETMDBGRQ** can be deasserted. It is also used for other purposes in the ETM, and care must be taken to ensure the timing of this signal is appropriate because it does not come through the main core/ETM interface.
DBGnTRST	Input	Reset for JTAG TAP controller.
DBGTCKEN	Input	Synchronous enable for the JTAG interface.
DBGTDI	Input	Test data in. Must be connected to DBGTDI input to the core.
DBGTDO	Output	Test data out. Must be multiplexed externally with DBGTDO from the core, controlled by DBGTDOSEL.
DBGTDOSEL	Output	Selects between core and ETM **DBGTDO**.
DBGTMS	Input	Test mode select. Must be connected to the **DBGTMS** input to the core.
ETMCPADDRESS[14:0]	Input	Coprocessor interface address. The relevant fields of the `MRC` or `MCR` instruction are encoded here as follows: Bits 14:12 `<Opcode_1>` Bits 11:8 `<CRn>` Bits 7:4 `<CRm>` Bit 3 = 0 cp14 Bit 3 = 1 cp15 Bits 2:0 `<Opcode_2>` For an ETM register access: Opcode_1 = 0 CRn = 0 CRm = Bits 3:0 of the register number CP14 or CP15 Opcode_2 = Bits 6:4 of the register number.
ETMCPCOMMIT	Input	Coprocessor interface commit. If this is LOW two cycles after **ETMCPEnable** is asserted, the transfer is canceled and must not take any effect.

Table A-1 ETM11RV signals (continued)

Signal name	Input/ output	Description
ETMCPENABLE	Input	Coprocessor interface enable. **ETMCPWrite** and **ETMCPAddress** are valid this cycle, and the remaining signals are valid two cycles later.
ETMCPRDATA[31:0]	Output	Coprocessor interface read data.
ETMCPWDATA[31:0]	Input	Coprocessor interface write value.
ETMCPWRITE	Input	Coprocessor interface read or write (asserted for write).
ETMDA[31:3]	Input	Address for data transfer
ETMDACTL[17:0]	Input	Data address interface control signals.
ETMDBGRQ	Output	Request from the ETM11RV 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.
ETMDD[63:0]	Input	Contains the data value for a Load, Store, `MRC`, or `MCR` instruction.
ETMDDCTL[3:0]	Input	Instruction interface control signals.
ETMEN	Output	Trace port enabled. Must not be used to power down functionality required by the ETM unless it only relates to the trace port.
ETMIA[31:0]	Input	Address for executed instruction.
ETMIACTL[17:0]	Input	Instruction interface control signals.
ETMIARET[31:0]	Input	Address to return to if branch is incorrectly predicted.
ETMPADV[2:0]	Input	Pipeline advance signals.
ETMPWRUP	Output	Indicates that the ETM is in use. When LOW, logic supporting the ETM must be clock-gated to conserve power.
ETMWFIPENDING	Input	Indicates that the ARM1136JF-S or ARM1136J-S core is about to go into Standby mode, and that the ETM must drain its FIFO.

Table A-1 ETM11RV signals (continued)

Signal name	Input/ output	Description
EVNTBUS[19:0]	Input	Gives the status of the performance monitoring events. Used as extended external inputs.
EXTIN[3:0]	Input	External input resources.
EXTOUT[1:0]	Output	External outputs.
FIFOPEEK[8:0]	Output	For validation purposes only. Indicates when various events occur before being written to the FIFO.
MAXEXTIN[2:0]	Input	External inputs supported by the ASIC (maximum 4). This appears in the configuration code register.
MAXEXTOUT[1:0]	Input	External outputs supported by the ASIC (maximum 2). This appears in the configuration code register.
MAXPORTSIZE[3:0]	Input	Maximum port size supported. This appears in the system configuration register. You must set this bus to one of the following values: b0000 4-bit port b0001 8-bit port b0010 16-bit port b0011 24-bit port b0100 32-bit port.
MUXINSEL	Input	Enable test wrapper for shared inputs.
MUXOUTSEL	Input	Enable test wrapper for shared outputs.
nETMWFIREADY	Output	Indicates that ETM11RV FIFO is empty and that ETM11RV can be put into Standby mode.
nPORESET	Input	Reset for most registers. This is a power-on reset, and must not be asserted during normal core reset to enable tracing through reset.
PORTMODE[2:0]	Output	Currently requested port mode.
PORTSIZE[3:0]	Output	Currently requested port size
SCANMODE	Input	Selects scan test mode.
SE	Input	Scan enable

Table A-1 ETM11RV signals (continued)

Signal name	Input/ output	Description
TRACECLK	Output	Clock for **TRACEDATA[31:0]** and TRACECTL.Data is valid on both edges of this clock for maximum integrity.
TRACECTL	Output	Used by trace capture devices. This signal is valid for the same time as **TRACEDATA**.Trigger = **TRACECTL** & !**TRACEDATA[0]** TraceDisabled = **TRACECTL** & **TRACEDATA[0]**.
TRACEDATA[31:0]	Output	32-bit trace port. Only data on this bus has to be captured.
TRACEREADY	Input	If this signal is LOW, FIFO draining is not permitted this cycle and **TRACEVALID**, TRIGGER, **TRACECTL** and **TRACEDATA** must hold the same values next cycle. The behavior of **TRACECLK** is undefined if this input is used.
TRACEVALID	Output	Trace is valid on this **CLK** cycle.Equivalent to !**TRACECTL** | !**TRACEDATA[0**], but only asserted for one **CLK** cycle. This can therefore be used as a clock enable for **TRACECTL** and **TRACEDATA**.
TRIGGER	Output	Trigger output this cycle.Equivalent to **TRACECTL** & !**TRACEDATA[0]**.

Appendix B
AC Parameters

This appendix describes the AC timing parameters for the ETM11RV. It contains the following sections:

- *Timing diagrams and timing parameters* on page B-2.
- *Reset timing parameters* on page B-3
- *Trace interface signals* on page B-4
- *Trace port timing parameters* on page B-6
- *Debug timing parameters* on page B-8
- *TAP interface timing parameters* on page B-9
- *Control signal timing parameters* on page B-10
- *ETM interface timing parameters* on page B-11
- *Design for test (DFT) timing parameters* on page B-13
- *External I/O timing parameters* on page B-14.

B.1 Timing diagrams and timing parameters

All figures are expressed as percentages of the **CLK** period at maximum operating frequency.

The figures quoted are relative to the rising clock edge after the clock skew for internal buffering has been added. Inputs given a 0% hold figure therefore require a positive hold relative to the top-level clock input. The amount of hold required is equivalent to the internal clock skew.

B.2 Reset timing parameters

The reset inputs, **nPORESET** and **DBGnTRST**, are asynchronous inputs and are synchronized internally. They are therefore unconstrained.

B.3 Trace interface signals

Table B-1 shows the trace interface timing parameters.

Table B-1 Trace interface timing parameters

Symbol	Parameter	Min	Max
$T_{isetmiactl}$	**ETMIACTL** input set up to rising **CLK**	40%	-
$T_{ihetmiactl}$	ETMIACTL input hold from rising **CLK**	0%	-
$T_{isetmia}$	ETMIA input set up to rising **CLK**	40%	-
$T_{ihetmia}$	ETMIA input hold from rising **CLK**	0%	-
$T_{isetmdact}l$	ETMDACTL input set up to rising **CLK**	40%	-
$T_{ihetmdactl}$	ETMDACTL input hold from rising **CLK**	0%	-
$T_{isetmda}$	ETMDA input set up to rising **CLK**	40%	-
$T_{ihetmda}$	ETMDA input hold from rising **CLK**	0%	-
$T_{isetmdd}$	ETMDD input set up to rising **CLK**	40%	-
$T_{ihetmdd}$	ETMDD input hold from rising **CLK**	0%	-
$T_{isetmddctl}$	ETMDDCTL input set up to rising **CLK**	40%	-
$T_{ihetmddctl}$	ETMDDCTL input hold from rising **CLK**	0%	-
$T_{isetmpadv}$	ETMPADV input set up to rising **CLK**	40%	-
$T_{ihetmpadv}$	ETMPADV input hold from rising **CLK**	0%	-

Figure B-1 on page B-5 shows the trace interface timing parameters.

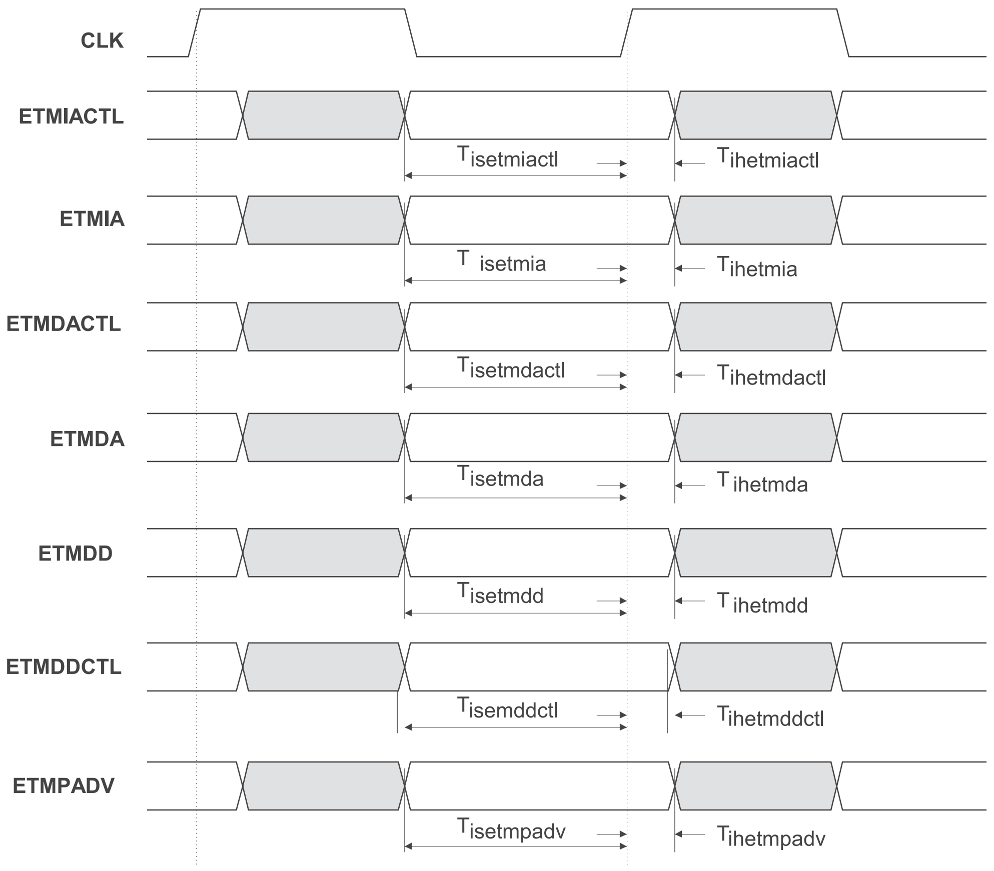


Figure B-1 Trace interface timing parameters

B.4 Trace port timing parameters

Table B-2 shows the trace port timing parameters.

Table B-2 Trace port timing parameters

Symbol	Parameter	Min	Max
$T_{ovtracedata}$	**TRACEDATA** output valid from rising **CLK**	-	60%
$T_{ohtracedata}$	**TRACEDATA** output hold from rising **CLK**	>0%	-
$T_{ovtraceclk}$	**TRACECLK** output valid from rising **CLK**	-	60%
$T_{ohtraceclk}$	TRACECLK output hold from rising **CLK**	>0%	-
$T_{ovtracectl}$	**TRACECTL** output valid from rising **CLK**	-	60%
$T_{ohtracectl}$	TRACECTL output hold from rising **CLK**	>0%	-
$T_{ovtracevalid}$	**TRACEVALID** output valid from rising **CLK**	-	60%
$T_{ohtracevalid}$	TRACEVALID output hold from rising **CLK**	>0%	-
$T_{ovtrigger}$	**TRIGGER** output valid from rising **CLK**	-	60%
$T_{ohtrigger}$	TRIGGER output hold from rising **CLK**	>0%	-
$T_{ovetmen}$	**ETMEN** output valid from rising **CLK**	-	60%
$T_{ohetmen}$	ETMEN output hold from rising **CLK**	>0%	-
$T_{istraceready}$	**TRACEREADY** input set up to rising **CLK**	40%	-
$T_{ihtraceready}$	TRACEREADY input hold from rising **CLK**	0%	-

Figure B-2 on page B-7 shows the trace port timing parameters.

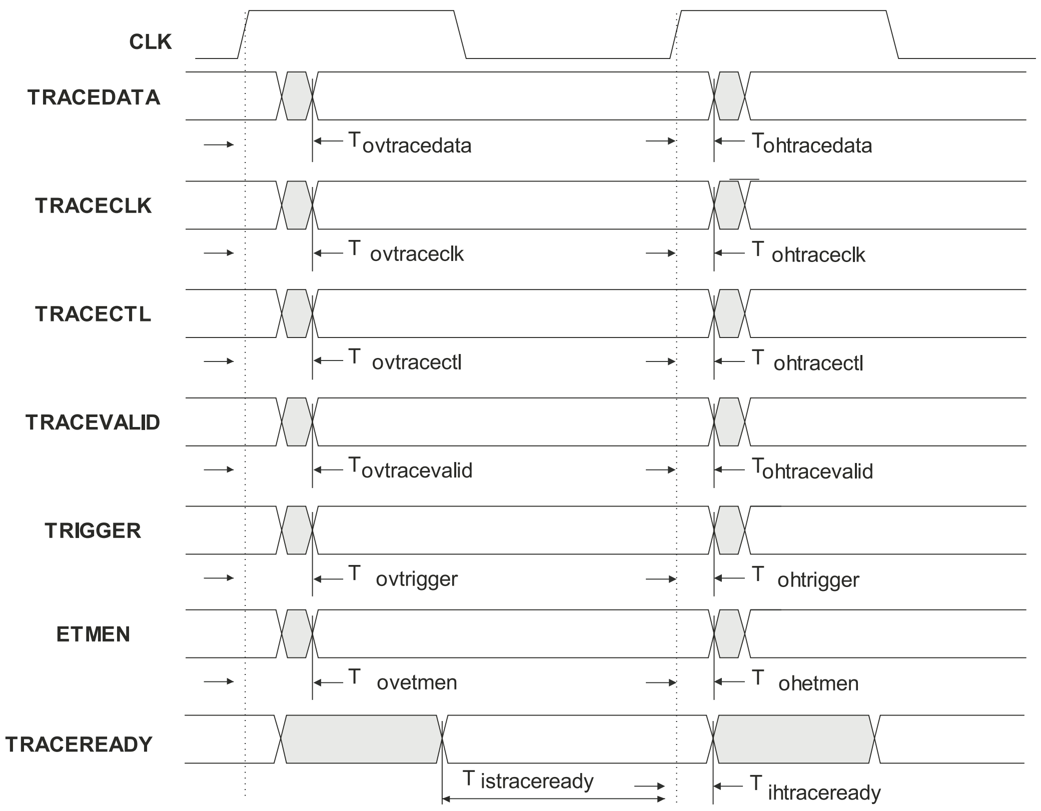


Figure B-2 Trace port timing parameters

B.5 Debug timing parameters

Table B-3 shows the debug timing parameters.

Table B-3 Debug timing parameters

Symbol	Parameter	Min	Max
$T_{ovetmdbgrq}$	**ETMBDBGRQ** output valid from rising **CLK**	-	60%
$T_{ohetmdbgrq}$	**ETMBDBGRQ** output hold from **rising CLK**	>0%	-
$T_{isdbgack}$	DBGACK input set up to rising **CLK**	40%	-
$T_{ihdbgack}$	DBGACK input hold from **rising CLK**	0%	-

Figure B-3 shows the debug timing parameters.

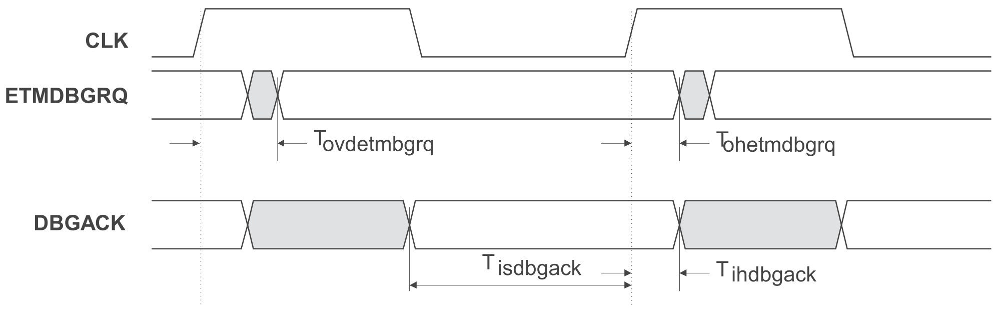


Figure B-3 Debug timing parameters

B.6 TAP interface timing parameters

Table B-4 shows the TAP interface timing parameters.

Table B-4 TAP interface timing parameters

Symbol	Parameter	Min	Max
$T_{isdbgtdi}$	**DBGTDI** input set up to rising **CLK**	40%	-
$T_{ihdbgtdi}$	**DBGTDI** input hold from **rising CLK**	0%	-
$T_{ovdbgtdo}$	**DBGTDO** output valid from rising **CLK**	-	60%
$T_{ohdbgtdo}$	**DBGTDO** output hold from **rising CLK**	>0%	-
$T_{ovdbgtdosel}$	**DBGTDSELOSEL** output valid from rising **CLK**	-	60%
$T_{ohdbgtdosel}$	**DBGTDOSEL** output hold from **rising CLK**	>0%	-

Figure B-4 shows the TAP interface timing parameters.

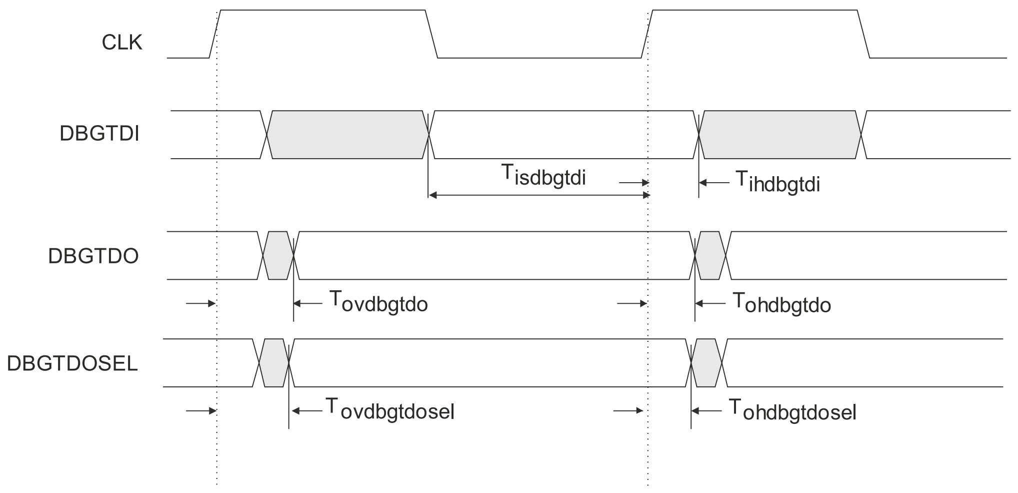


Figure B-4 TAP interface timing parameters example

B.7 Control signal timing parameters

Table B-5 shows the control signal parameters.

Table B-5 Control signal timing parameters

Symbol	Parameter	Min	Max
$T_{ovetmpwrup}$	**ETMPWRUP** output valid from rising **CLK**	-	60%
$T_{ohetmpwrup}$	ETMPWRUP output hold from **rising CLK**	>0%	-
$T_{ovasicctl}$	**ASICCTL** output valid from rising **CLK**	-	60%
$T_{ohasicctl}$	ASICCTL output hold from **rising CLK**	>0%	-
$T_{isetmwfipending}$	**ETMWFIPENDING** input set up to rising **CLK**	40%	-
$T_{ihetmwfipending}$	ETMWFIPENDING input hold from **rising CLK**	0%	-
$T_{ovnetmwfiready}$	**ETMWFIREADY** output valid from rising **CLK**	-	60%
$T_{ohnetmwfiready}$	ETMWFIREADY output hold from **rising CLK**	>0%	-

Figure B-5 shows the control signal timing parameters.

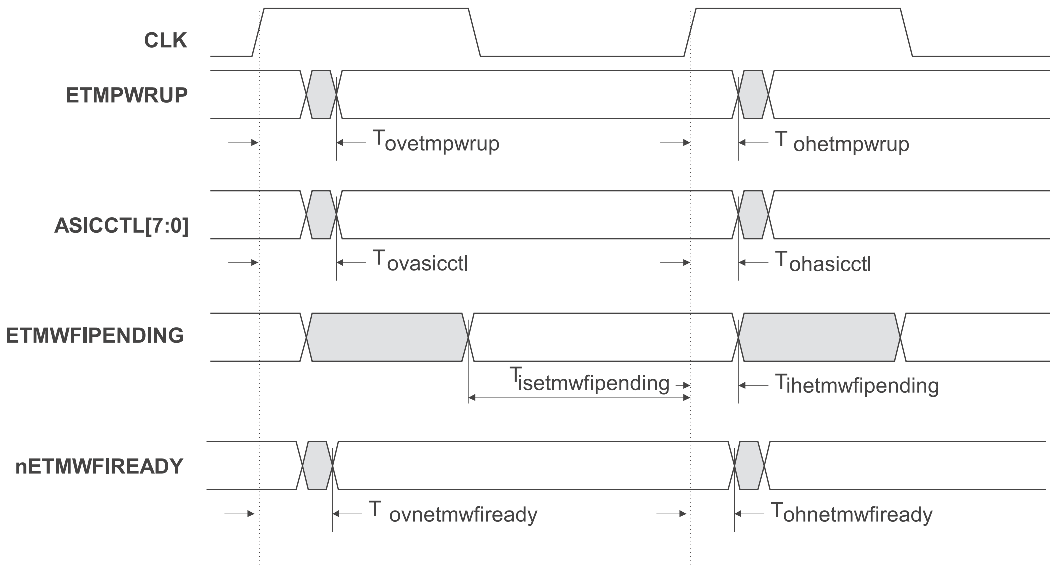


Figure B-5 Control signal timing parameters

B.8 ETM interface timing parameters

Table B-6 shows the ETM interface timing parameters

Table B-6 ETM interface timing parameters

Symbol	Parameter	Min	Max
$T_{ovetmcprdata}$	**ETMCPRDATA** output valid from rising **CLK**	-	60%
$T_{ohetmcprdata}$	ETMCPRDATA output hold from **rising CLK**	>0%	-
$T_{isetmcpwrite}$	**ETMCPWRITE** input set up to rising **CLK**	40%	-
$T_{ihetmcpwrite}$	ETMCPWRITE input hold from **rising CLK**	0%	-
$T_{isetmcpenable}$	**ETMCPENABLE** input set up to rising **CLK**	40%	-
$T_{ihetmcpenable}$	ETMCPENABLE input hold from **rising CLK**	0%	-
$T_{isetmcpcommit}$	**ETMCPCOMMIT** input set up to rising **CLK**	40%	-
$T_{ihetmcpcommit}$	ETMCPCOMMIT input hold from **rising CLK**	0%	-
$T_{isetmcpaddress}$	**ETMCPADDRESS** input set up to rising **CLK**	40%	-
$T_{ihetmcpaddress}$	ETMCPADDRESS input hold from **rising CLK**	0%	-
$T_{isetmcpwdata}$	**ETMCPWDATA** input set up to rising **CLK**	40%	-
$T_{ihetmcpwdata}$	ETMCPWDATA input hold from **rising CLK**	0%	-

Figure B-6 on page B-12 shows the ETM interface timing parameters.

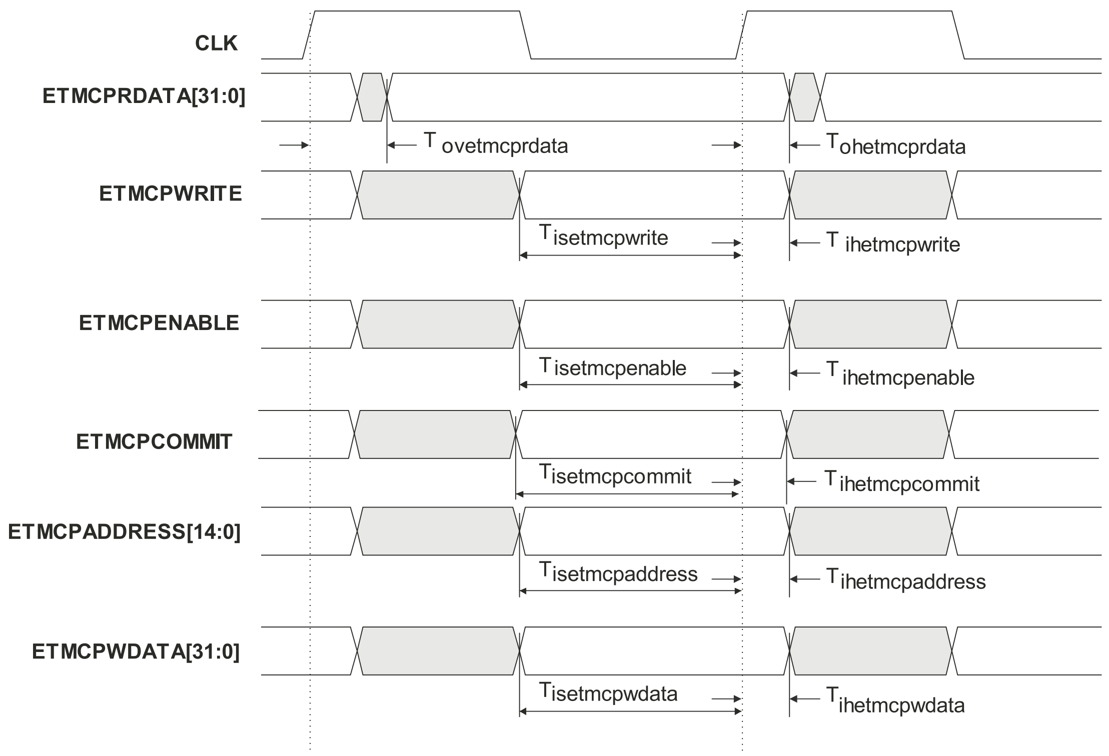


Figure B-6 ETM interface timing parameters

B.9 Design for test (DFT) timing parameters

Table B-7 shows the DFT timing parameters.

Table B-7 DFT timing parameters

Symbol	Parameter	Min	Max
T_{isse}	**SE** input set up to rising **CLK**	95%	-
T_{ihse}	SE input hold from **rising CLK**	0%	-
$T_{isscanmode}$	**SCANMODE** input set up to rising **CLK**	95%	-
$T_{ihscanmode}$	SCANMODE input hold from **rising CLK**	0%	-
$T_{ismuxinsel}$	**MUXINSEL** input set up to rising **CLK**	95%	-
T_{ih}muxinsel	MUXINSEL input hold from **rising CLK**	0%	-
T_{is}muxoutsel	MUXOUTSEL input set up to rising **CLK**	95%	-
T_{ih}muxoutsel	MUXOUTSEL input hold from **rising CLK**	0%	-

Figure B-7 shows the DFT timing parameters.

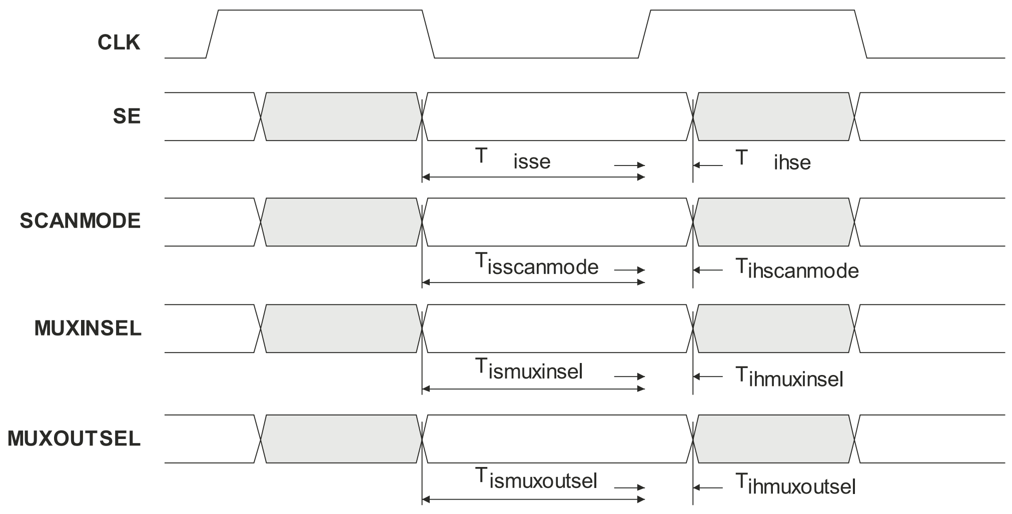


Figure B-7 Scan chain timing parameters

B.10 External I/O timing parameters

Table B-8 shows the external input and output timing parameters.

Table B-8 External input and output timing parameters

Symbol	Parameter	Min	Max
$T_{ovextout}$	**EXTOUT** output valid from rising **CLK**	-	60%
$T_{ohextout}$	EXTOUT output hold from **rising CLK**	>0%	-
$T_{ovfifopeek}$	**FIFOPEEK** output valid from rising **CLK**	-	60%
$T_{ohfifopeek}$	FIFOPEEK output hold from **rising CLK**	>0%	-
$T_{ovportmode}$	**PORTMODE** output valid from rising **CLK**	-	60%
$T_{ohportmode}$	PORTMODE output hold from **rising CLK**	>0%	-
$T_{ivevntbus}$	EVNTBUS input set up to rising **CLK**	40%	-
$T_{ihevntbus}$	EVNTBUS input hold from **rising CLK**	0%	-
$T_{ivextin}$	EXTIN input set up to rising **CLK**	40%	-
$T_{ihextin}$	EXTIN input hold from **rising CLK**	0%	-
$T_{ivmaxportsize}$	MAXPORTSIZE input set up to rising **CLK**	40%	-
$T_{ihmaxportsize}$	MAXPORTSIZE input hold from **rising CLK**	0%	-
$T_{ivportsize}$	PORTSIZE input set up to **rising CLK**	40%	-
$T_{ihportsize}$	PORTSIZE input hold from **rising CLK**	0%	-
$T_{ivmaxetin}$	MAXEXTIN input set up to **rising CLK**	40%	-
$T_{ihmaxetin}$	MAXEXTIN input hold from **rising CLK**	0%	-
$T_{ivmaxextout}$	MAXEXTOUT input set up to rising **CLK**	40%	
$T_{ihmaxextout}$	MAXEXTOUT input hold from **rising CLK**	0%	-

Figure B-8 on page B-15 shows the external input and output timing parameters.

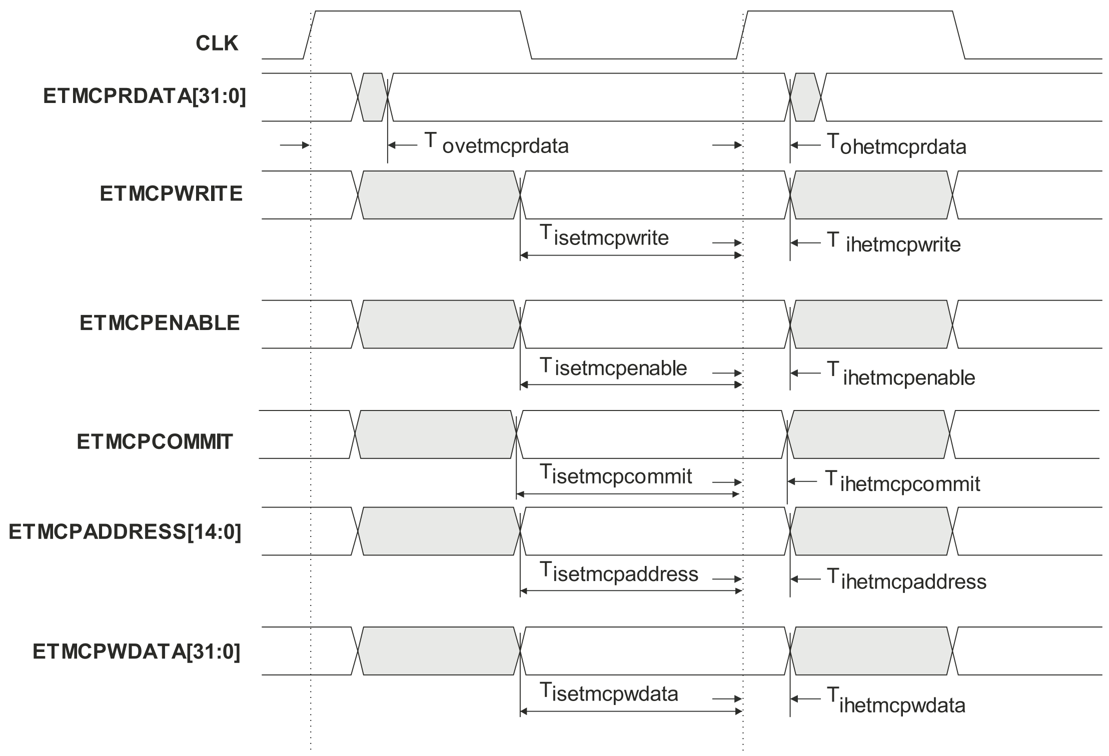


Figure B-8 External I/O timing parameters

Glossary

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

Application Specific Integrated Circuit(ASIC)	An integrated circuit that has been designed to perform a specific application function. It can be custom-built or mass-produced.
Application Specific Standard Part/Product (ASSP)	Another name for an Application Specific Integrated Circuit. The name implies that the device performs complete functions, and can be used as a building block in a range of products.
ASIC	*See* Application Specific Integrated Circuit.
ASSP	*See* Application Specific Standard Part/Product.
Coprocessor Register Transfer (CPRT)	`MCR`, `MRC`,`MCRR`, and `MRRC` instructions are Coprocessor Register Transfers.
CPRT	*See* Coprocessor Register Transfer.
Clock gating	Gating a clock signal for a macrocell with a control signal (such as **PWRDOWN**) 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.

Debugger
A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.

An application that monitors and controls the operation of a second application. Usually used to find errors in the application program flow.

Embedded Trace Macrocell (ETM)
A hardware macrocell which, when connected to a processor core, outputs instruction and data trace information on a trace port.

ETM
See *Embedded Trace Macrocell*.

Implementation-defined
Means that the behavior is not architecturally defined, but should be defined and documented by individual implementations.

Implementation Specific
Means that the exact behavior is not architecturally defined, and need not be documented by individual implementations. This term is used when there are a number of implementation options available and the option chosen does not affect software compatibility.

Imprecise Tracing
A filtering configuration where instruction or data tracing can start or finish earlier or later than expected. Most cases cause tracing to start or finish later than expected.

For example, if **TraceEnable** is configured to use a counter so that tracing begins after the fourth write to a location in memory, the instruction that caused the fourth write is not traced, although subsequent instructions are. This is because the use of a counter in the **TraceEnable** configuration always results in Imprecise Tracing.

Jazelle architecture
The ARM Jazelle architecture extends the Thumb and ARM operating states by adding a Java state to the processor. Instruction set support for entering and exiting Java applications, real-time interrupt handling, and debug support for mixed Java/ARM applications is present. When in Java state, the processor fetches and decodes Java bytecodes and maintains the Java operand stack.

Joint Test Action Group (JTAG)
The name of the organization that developed standard IEEE 1149.1. This standard defines a boundary-scan architecture used for in-circuit testing of integrated circuit devices. It is commonly known by the initials JTAG.

JTAG	*See* Joint Test Action Group.
Macrocell	A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as an ARM9E-S, an ETM9, and a memory block) plus application-specific logic.
Power-on reset	*See* Cold reset.
SCREG	The currently selected scan chain number in an ARM TAP controller.
SPICE	An accurate transistor-level simulation tool.
TAP	*See* Test access port.
TCD	*See* Trace Capture Device.
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**.
Trace Capture Device	A generic term to describe Trace Port Analyzers, logic analyzers, and on-chip trace buffers.
Trace driver	A Remote Debug Interface target that controls a piece of trace hardware. That is, the trigger macrocell, trace macrocell, and trace capture tool.
Trace hardware	A term for a device that contains an Embedded Trace Macrocell.
Trace port	A port on a device, such as a processor or ASIC, used to output trace information.
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.
Unpredictable	Means that the behavior of the ETM cannot be relied upon. 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 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 may 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.

Index

The items in this index are listed in alphabetical order. The references given are to page numbers.

A

Accesses
  concurrent 2-7
  conflicting 2-7
Address comparators
  pairs 1-3
  sticky bit 3-8
ASICCTL A-2

C

CLK A-2
  clock 2-2
Clocks
  CLK 2-2
  core 2-2
  JTAG 2-2
  skew B-2
  TCLK 2-2
  TRACECLK 2-2
  TRACEVAVLID 2-2
Comparators
  Context ID 1-3
  resources 1-3
Configuration code extension register 3-5
Configuration code register 3-3
Context ID
  and MCR instructions 3-10
  comparator 1-3
Conventions
  numerical xiii
  signal naming xii
  timing diagram xi
  typographical xi
Core
  clock 2-2

D

DBGACK A-2
DBGnTRST A-2
DBGTDIl A-2
DBGTDO A-2
DBGTDOSEL A-2
DBGTMS A-2

E

ETM
  JTAG structure 2-4
  programming bit 2-2
  registers 2-2
  registers, programming and reading 2-4
ETMCPADDRESS A-3
ETMCPCOMMIT A-3
ETMCPENABLE A-3
ETMCPRDATA A-3
ETMCPWRITE A-3
ETMDA A-3
ETMDACTL A-3
ETMDBGRQ A-3
ETMDD A-3
ETMDDCTL A-3

ETMEN A-4
ETMIA A-4
ETMIACTL A-4
ETMIARET A-4
ETMPADV A-4
ETMPWRUP A-4
ETMWFIPENDING A-4
ETM11RV 3-5
EVNTBUS A-4, B-15
EXTIN A-4
EXTOUT A-4

F

FIFO size
  resources 1-3
FIFOPEEK A-4

I

ID register
  ETM11RV 3-3

J

JTAG
  clock 2-2
  structure 2-4

M

MAXEXTIN A-4
MAXEXTOUT A-4
MAXPORTSIZE A-5
MCR instructions
  and Context ID 3-10
MUXINSEL A-5
MUXOUTSEL A-5

N

nETMWFIREADY A-5
nPORESET A-5
  and resets 3-12
Numerical conventions xiii

P

PORTMODE A-5
PORTSIZE A-5
Product revision status x
Programming bit
  ETM 2-2
Protocols
  trace 1-2

R

Registers
  configuration code 3-3
  configuration code extension 3-5
  ETM 11RV ID register 3-3
  programming and reading 2-4
Resets
  and DBGnTRST 3-12
  and nPORESET 3-12
  power-on 3-12
  TAP 3-12
Resources
  comparators 1-3
  counters 1-3
  FIFO size 1-3
  sequencers 1-3
Revision
  status x

S

SCANMODE A-5
SE A-5
Sequencers
  resources 1-3
Setting trigger conditions correctly 2-2
Signal naming conventions xii
Sticky bit
  address comparators 3-8
  TraceEnable 3-9
  ViewData 3-9

T

TAP
  reset 3-12
TCLK
  clock 2-2
Timing diagram conventions xi
Trace
  protocol 1-2
TRACECLK A-5
TRACECTL A-5
TRACEDATA A-5
TraceEnable
  sticky bit 3-9
TRACEREADY A-6
TRACEVALID A-6
TRIGGER A-6
Typographical conventions xi

V

ViewData
  sticky bit 3-9