ETB Rev0 Errata List

CPU Cores Division

Document number:	**ETB-PRDC-001103 17.0**
Date of Issue:	11 February 2011
Author:	ARM

Abstract

This document describes the known errata in the Embedded Trace Buffer design.

Keywords

ETM, ETB, errata

This is a working document throughout the product lifecycle and, as such, the content may be modified as new information is uncovered.

Contents

1 ABOUT THIS DOCUMENT

1.1 Current History

Issue	Date	Change
1.0	4 December 2001	Initial Release – No Errata
5.0	17 March 2003	Errata 4.1 and 5.1 added
13.0	17 July 2003	Errata 4.2 and 4.3 added
15.0	7 November 2003	Affected revision has been corrected for erratum 5.1.
16.0	9 May 2008	Errata 4.4 and 4.5 added
17.0	11 February 2011	Errata 6.1 and 7.1 added

Intermediate issue numbers were internal drafts and are not included.

1.2 References

This document refers to the following documents.

Ref.	Document No	Author(s)	Title
1	ARM DDI 0242A	ARM	ETM Trace Buffer Technical Reference Manual

1.3 Scope

This document describes the errata discovered in the implementation of the ETM Trace Buffer, categorised by level of severity. Each description includes:

- where the implementation deviates from the specification
- the conditions under which erroneous behaviour occurs
- the implications of the erratum with respect to typical applications
- the application and limitations of a ‘work-around’ where possible
- the status of corrective action.

1.4 Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
ETM	Embedded Trace Macrocell
ETB	Embedded Trace Buffer

2 CATEGORISATION OF ERRATA

Errata recorded in this document are split into three groups:

Category	Definition
1	Features which are impossible to work around and severely restrict the use of the device in all or the majority of applications rendering the device unusable.
2	Features which contravene the specified behaviour and may limit or severely impair the intended use of specified features but does not render the device unusable in all or the majority of applications.
3	Features that were not the originally intended behaviour but should not cause any problems in applications.
System	Errata or possible issues that have system implications and therefore should be considered by system designers
Documentation	Errata or possible issues that have documentation implications.

2.1 Errata Summary

The errata associated with this product are categorised in the following way.

Category	Errata Description	Affected Rev.
1	None	-
2	Potential Loss of Last Data Packet when Trace Capture is Disabled	rev.0p0
	Synchronous Mode – Incorrect Software Read Access to ETB RAM following an IDLE transfer with HWRITE HIGH	rev.0p0
	Synchronous Mode – Incorrect Software Read Access to ETB RAM following or preceding a Software Read Access to ETB Registers	rev.0p0 and rev.0p1
	ETB violates AHB interface protocol during error response	rev.0p0 and rev.0p1 and rev.0p2
	ETB register accesses using JTAG interface might not operate correctly	rev.0p0 and rev.0p1 and rev.0p2
3	Synchronous Mode - Register Read directly followed by Memory Read	rev.0p0
System	AHB interface might respond incorrectly to IDLE or BUSY transactions	rev.0p0 and rev.0p1 and rev.0p2
Documentation	ETB AHB interface is little endian	rev.0p0 and rev.0p1 and rev.0p2

3 CATEGORY 1 ERRATA

There are no errata in this category.

4 CATEGORY 2 ERRATA

4.1 Potential Loss of Last Data Packet when Trace Capture is Disabled

This erratum affects ETB rev.0p0.

4.1.1 Description

A Data Formatter constructs ETB RAM data from trace that has been read in from the **TRACEOUTPUT** port. This is to allow multiple cycles of trace to be stored in a single word of ETB Trace RAM. If trace capture is disabled and the Data Formatter contains a partially built value, the Data Formatter inserts Trace Disabled (TD) packets until the data value is complete. Under certain configurations the last data word inserted with TD packets is not guaranteed to have been written to the RAM and is lost.

4.1.2 Conditions

This erratum does not apply if:

- **PROTOCOL[1:0]** = b10 for the generic architecture used by ETMv3 devices, or
- **TRACEPKT** width = 16 bits, or
- **SWEN** and SoftwareCntl are HIGH, or
- AcqComp and Triggered are HIGH.

The erratum occurs when the following four conditions are met:

- The ETM architecture version supplied to the ETB is:
    - **PROTOCOL[1:0]**=b00 for ETMv1 architecture, or
    - **PROTOCOL[1:0]**=b01 for ETMv2 architecture.
- The packet width (**TRACEPKT**) is set to:
    - **PORTSIZE[2:0]**=b000 for **TRACEPKT** width = 4 bits, or
    - **PORTSIZE[2:0]**=b001 for **TRACEPKT** width = 8 bits
- Software access to the ETB is disabled
    - The software access pin **SWEN** for AHB access is set LOW, or
    - Software access to the ETB registers is disabled by setting the SoftwareCntl bit of the Control Register LOW.
- The ETB has not stopped capturing trace due to the trigger counter having reached 0. In other words, when reading the Status register (register 3), either:
    - Triggered (bit 1) is LOW, or
    - AcqComp (bit 2) is LOW.

  If the ETB has stopped capturing trace due to the Trigger Counter Register decrementing to 0, then this erratum does not apply and Triggered and AcqComp will be HIGH. If the ETB has not stopped capturing trace, or trace capture has been stopped for any other reason then this erratum may occur.

In summary, any data packets (a maximum of 2) remaining in the Data Formatter may be lost when:

(ETMv1 ***OR*** ETMv2 architecture) ***AND***

(4-bit ***OR*** 8-bit trace packet width) ***AND***

(Software Enable input = 0 ***OR*** SoftwareCntl = 0) ***AND***

(AcqComp = 0 ***OR*** Triggered = 0).

4.1.3 Implications

Any data packets in the Data Formatter when trace capture is disabled (by setting the TraceCaptEn bit of the Control Register LOW) will not be written to the ETB RAM and the data will be lost:

- If **TRACEPKT** width = 4 bits a maximum of 2 trace packets may be lost.
- If **TRACEPKT** width = 8 bits a maximum of 1 trace packet may be lost.

4.1.4 Workaround

To guarantee that no data is lost under the conditions described in section 4.1.2 one of three methods can be used:

- Set **TRACEPKT** width to 16 bits.
- If **SWEN** is HIGH then set SoftwareCntl HIGH at the same time as TraceCaptEn is set LOW. SoftwareCntl will be automatically cleared when the INTEST instruction is selected. To set SoftwareCntl high an additional instruction, such as BYPASS must be selected, then the INTEST instruction can be reselected.
- If **SWEN** is LOW, and it is not possible to change it to HIGH, then reprogram the ETM to produce additional trace data as described in section 4.1.4.1.

4.1.4.1 Reprogramming the ETM to produce more trace data

The ETM should be configured to trace the minimum amount of additional data:

1. Configure and program the ETM to trace the desired data. Do not set the TraceCaptEn bit of the Control Register LOW at this point – Ensure that ETB trace capture is still enabled.
2. Reprogram the ETM as described in Table 1. This configures the ETM for:
   - Cycle accurate instruction only tracing,
   - of executed ARM state instructions,
   - in the full address range,
   - with ETM counter 1 set to start at a value of 2 and decrementing each cycle,
   - with tracing disabled when the counter reaches 0, and never reloading.

   This results in three additional instructions being traced.

Table 12: ETM programming to ensure ETB is fully drained of valid trace data

Register	Value	Register Description	Configuration
0x00	Set bit [12] = 1	ETM control register	Enable cycle-accurate tracing, in addition to other control bits.
0x06	0x00000000	Trace start/stop resource control	No start/stop addresses
0x07	0x0000	TraceEnable control 2 register	No single address comparator include/exclude control
0x08	0x04040	TraceEnable event register	Counter 1 is non-zero
0x09	0x1000000	TraceEnable control 1 register	Exclude nothing

0x0C	0x0406F	ViewData event register	Always 0x0, no data trace
0x10	0x00000	Address comparator 1 value register	Address 0x00000000
0x11	0xFFFFFFFF	Address comparator 2 value register	Address 0xFFFFFFFF
0x20	0x019	Address comparator 1 access type register	ARM instruction, instruction execute
0x21	0x019	Address comparator 2 access type register	ARM instruction, instruction execute
0x50	0x0002	Counter 1 reload value register	Initial count = 0x2
0x54	0x00010	Counter 1 enable register	Address range comparator 1
0x58	0x0406F	Counter 1 reload event	Always 0x0, never reload

3. Set the TraceCaptEn bit of the Control Register LOW.

4.1.5 Correction

This erratum is corrected in ETB rev.0p1.

4.2 Synchronous Mode – Incorrect Software Read Access to ETB RAM following an IDLE transfer with HWRITE HIGH

This erratum affects ETB rev.0p0.

4.2.1 Description

During an AHB transfer, if an IDLE transfer with HWRITE HIGH is made to the ETB RAM, then an ETB RAM read directly following this transfer will be invalid. An undefined value is written to the intended RAM read address and the returned read value is undefined.

JTAG access to the ETB is unaffected.

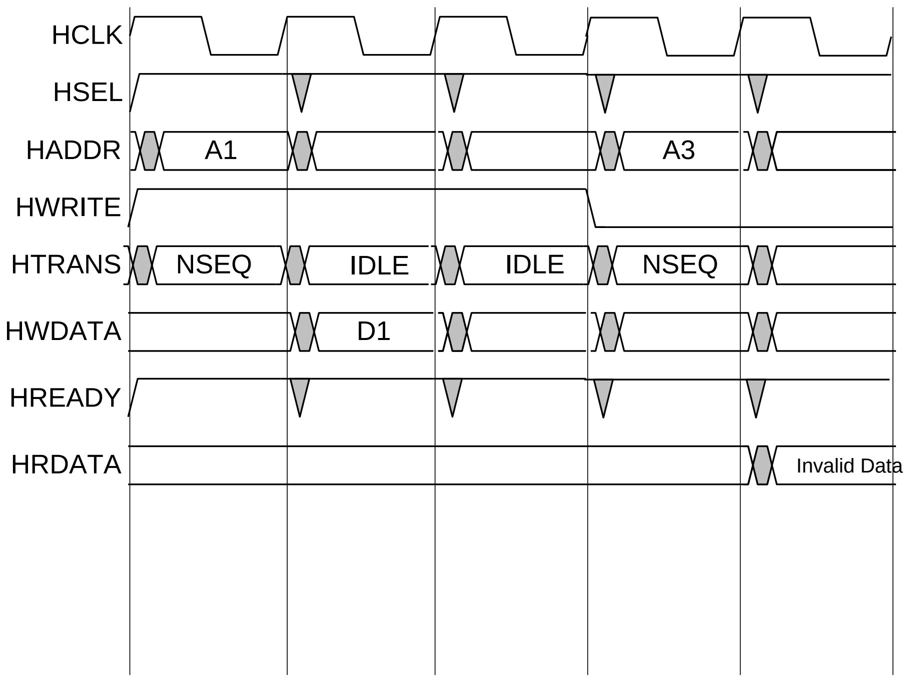


Figure 1: Waveform Diagram

4.2.2 Conditions

HWRITE must be high during the AHB IDLE cycle with the ETB selected. This is usually because HWRITE is held at its last value between AHB transfers. If HWRITE is driven low during AHB IDLE cycles, this erratum does not occur.

The ARM AHB Bus Matrix does not hold the previous HWRITE values during IDLE AHB cycles, so an ETB connected to it is not affected.

The only ARM CPU that is known to be affected is ARM1026EJ-S r0p1.

AHB Wrappers and matrix/bridge AHB components not supplied by ARM might cause this erratum.

ETB is not affected by this erratum if there is a single bus master that is any of the following ARM CPUs:

- ARM7TDMI rev3A and rev4 (with ARM supplied AHB wrapper)
- ARM720T rev3 (with ARM supplied AHB wrapper)
- ARM720T rev4
- ARM7TDMI-S rev4 (with ARM supplied AHB wrapper)
- ARM920T rev1 (with ARM supplied AHB wrapper)
- ARM922T rev0 (with ARM supplied AHB wrapper)

- ARM926EJ-S rev0p0, rev0p3 and rev0p4
- ARM946ES rev1
- ARM966ES rev1 and rev2
- ARM1020E rev1
- ARM1022E rev0
- ARM1026EJ-S r0p0

ARM has verified this using the ETM Integration Kit.

Multi-master systems that are connected with an AHB Bus Matrix not supplied by ARM might cause this erratum.

If the ARM supplied AHB Bus Matrix is used in a design, then idle AHB transfers have HWRITE low, so this erratum does not occur.

4.2.3 Implications

An undefined data value is written to the intended RAM read location and the returned read value is undefined.

4.2.4 Workaround

There are no known workarounds for this erratum in existing designs. For new designs, this erratum can be avoided by using the ETB in asynchronous mode (by tying **SBYPASS** to 0).

4.2.5 Correction

This erratum is corrected in ETB rev.0p1.

4.3 Synchronous Mode – Incorrect Software Read Access to ETB RAM following or preceding a Software Read Access to ETB Registers

This erratum affects ETB rev.0p0 and ETB rev.0p1.

4.3.1 Description

If a an AHB read access of the ETB RAM is made following or preceding an AHB read access of an ETB register, then the value read back for the ETB RAM access will be corrupted.

This erratum only occurs if the transactions are in sequence, i.e. Memory -> Register, or Register -> Memory with accesses to no other device in between.

JTAG access to the ETB is unaffected.

4.3.2 Conditions

The ETB must be in synchronous mode, with HCLK half the speed of CLK or slower for this erratum to exist.

Write accesses are not affected.

4.3.3 Implications

An undefined data value is read from the ETB RAM.

4.3.4 Workaround

- If the ETB RAM read precedes the register read:
    - Read the same RAM location twice and ignore the last memory read.
- If the ETB RAM read follows the register read:
    - Read the same RAM location twice and ignore the first memory read.

4.3.5 Correction

This erratum is corrected in ETB rev.0p2.

4.4 ETB violates AHB interface protocol during error response

ARM erratum reference: 518918.

This erratum affects ETB rev.0p0, rev.0p1 and rev.0p2.

4.4.1 Description

The ETB supports RAM accesses and programming register accesses via an AHB interface or via the JTAG interface. Accesses over the AHB interface can be disabled by clearing the SoftwareCntl bit in the ETB Control register 8. This bit can only be cleared by a JTAG access, or whenever the INTEST instruction is selected using the JTAG interface. The reset status of the SoftwareCntl bit is HIGH. AHB accesses can also be disabled by driving the SWEN input to the ETB LOW.

When AHB accesses are disabled the ETB should respond with an ERROR response over the AHB interface. The AHB specification requires an ERROR response to take at least 2 HCLK cycles. This erratum might cause the ETB to drive the ERROR response for only 1 HCLK cycle.

4.4.2 Conditions

The following conditions must occur:

- At least one of the following must occur:
  - The SWEN input to the ETB is LOW
  - The SoftwareCntl bit in the ETB Control register is LOW
- The SBYPASS input to the ETB is HIGH
- An AHB access is performed to the ETB

4.4.3 Implications

The AHB master might behave Unpredictably, including locking up the AHB interface. However, many AHB masters are tolerant to this single-cycle ERROR response. If your AHB master is tolerant to the single-cycle ERROR response then this erratum has no effect.

It is expected that this situation will not occur during normal usage of the ETB. There are two main uses of the ETB, for trace capture and as system memory. When used as system memory, it is not expected that a debugger will interact with the ETB and as such the SoftwareCntl bit will always be HIGH and this erratum does not occur. When used for trace capture, the ETB is usually only accessed by either the JTAG interface or the AHB interface and as such this erratum is not expected to occur.

4.4.4 Workaround

There is no workaround for this erratum.

Users should ensure that simultaneous accesses to the ETB from on-chip software and an external debugger should be avoided.

4.4.5 Correction

This erratum is not currently corrected.

4.5 ETB register accesses using JTAG interface might not operate correctly

ARM erratum reference: 532266.

This erratum affects ETB rev.0p0, rev.0p1 and rev.0p2.

4.5.1 Description

The ETB supports RAM accesses and programming register accesses via an AHB interface or via the JTAG interface. Accesses over the AHB interface can be disabled by clearing the SoftwareCntl bit in the ETB Control register 8. This bit can only be modified by a JTAG access.

If the SoftwareCntl bit is set, writes to all ETB registers via the JTAG interface might be ignored except for writes to the SoftwareCntl bit. Any reads of ETB registers return an Unknown value.

4.5.2 Conditions

The following conditions must occur:

- The SoftwareCntl bit in the ETB Control register is set
- A read or write access is attempted to any ETB register using the JTAG interface

It should be noted that whenever INTEST is selected in the ETB JTAG TAP controller, the SoftwareCntl bit is automatically cleared. Therefore this erratum only occurs if debug tools deliberately set the SoftwareCntl bit while performing JTAG accesses.

4.5.3 Implications

Writes to all bits of all ETB registers might be ignored, except writes to the SoftwareCntl bit of the ETB Control register 8. This means the ETB might not be enabled correctly.

Reads of ETB registers return an Unknown value.

4.5.4 Workaround

This is a workaround for tools vendors.

Tools should ensure that when accessing the ETB via JTAG the SoftwareCntl bit is clear. If the SoftwareCntl bit is set then it must be cleared before accessing any ETB registers.

4.5.5 Correction

This erratum is not currently corrected.

5 CATEGORY 3 ERRATA

5.1 Synchronous Mode - Register Read directly followed by Memory Read

This erratum affects ETB rev.0p0 and rev.0p1.

5.1.1 Description

When a memory read directly follows a register read in synchronous mode then the memory value read is incorrect.

5.1.2 Conditions

This has been observed with SBYPASS set to 1, using synchronous clocks. The data value read from the ETB memory is incorrect when the preceding AHB access is to an ETB register, and the next cycle is an access to the ETB memory. If an IDLE or non-ETB transfer occurs between the register and memory read, then the operation is not affected.

5.1.3 Implications

The user will see incorrect data for the memory read.

5.1.4 Workaround

If the user needs to perform a register read directly followed by a memory read then the first memory address must be read twice to obtain the correct data.

5.1.5 Correction

This erratum is corrected in ETB rev.0p2.

6 SYSTEM ERRATA

6.1 AHB interface might respond incorrectly to IDLE or BUSY transactions

ARM erratum reference: 643717.

This erratum affects ETB rev.0p0, rev.0p1 and rev.0p2.

6.1.1 Description

The AMBA AHB specification requires that a slave must respond to an IDLE or BUSY transaction with a zero wait-state OK response. The ETB might not always respond with a zero wait-state OK response to IDLE or BUSY transactions.

6.1.2 Conditions

The following conditions must occur:

- One of HSELMEM or HSELREG is HIGH
- An IDLE or BUSY transaction is indicated on HTRANS

The HREADY output from the ETB might be driven low during the data phase of the IDLE or BUSY transaction.

6.1.3 Implications

The AHB master driving the ETB might assume that IDLE or BUSY transactions will always result in a zero wait-state response. This erratum might cause the AHB master to drive a new NSEQ or SEQ transaction on the AHB interface while HREADY is low and therefore causing the ETB to miss the new transaction.

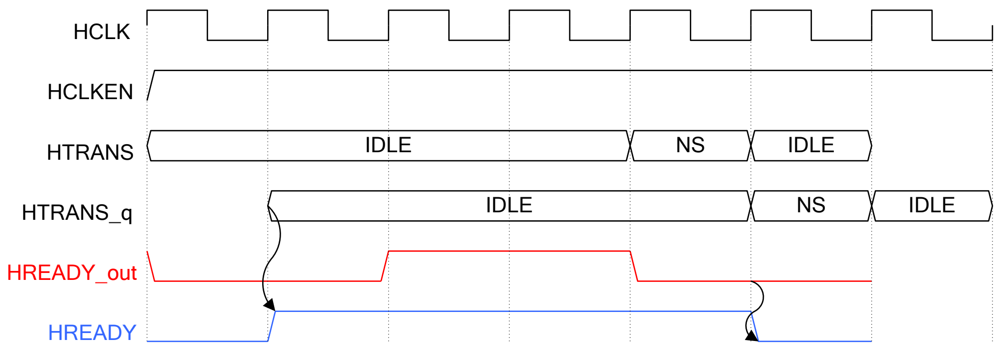
The new transaction might have incorrect data returned in the case of a read transaction, or a write transaction might be performed incorrectly in the ETB.

This erratum does not affect usage when only accessing via JTAG.

6.1.4 Workaround

This is a workaround for system implementors.

The HREADY output from the ETB must be qualified based on whether an IDLE or BUSY transaction is in progress. It is important that this is done during the data phase of the transfer (second cycle) and therefore a registered version of HTRANS[1] should be used. While this registered version of HTRANS[1] is b0 HREADY must be high. This is shown in the following timing diagram example illustrating an IDLE transaction:



This can be achieved with the following logic applied to HREADY output from the ETB creating a new HREADY output.

```
module ETBSetHready (/*AUTOARG*/
  // Inputs
  HCLK, HCLKEN, HRESETn, HTRANS, HREADY_out, HREADYBUS,   // Outputs
  HREADY
  );
  input       HCLK;          // AHB Clock
  input       HCLKEN;        // AHB Clock Enable
  input       HRESETn;       // AHB Reset
  input [1:0] HTRANS;        // AHB Trans
  input       HREADY_out;    // HREADY from ETB (slave)
  input       HREADYBUS;     // HREADY from Master
  output      HREADY;        // New ETB HREADY
  // wire declaration
  wire        htrans_en;     // Enable when to sample HTRANS
  reg         htrans_q;      // registered version of HTRANS
  wire        d_phase_idlebusy;   // during a data phase of a transfer HTRANS is IDLE/BUSY
  // ---------------------------------------------------------------------------
  // Main code
  // ---------------------------------------------------------------------------
  // Clock gating term to select HTRANS
  assign      htrans_en = HCLKEN & HREADYBUS;

  always @(posedge HCLK or negedge HRESETn)
    if (!HRESETn)
      htrans_q <= 1'b0;
    else if (htrans_en)
      htrans_q <= HTRANS[1];
  // We use the registered version of HTRANS because we are interested in the data
  // phase of the transfer.
  assign      d_phase_idlebusy = (htrans_q == 1'b0);
  // If trans is IDLE/BUSY the HREADY signal will always be high
  assign      HREADY = d_phase_idlebusy | HREADY_out;
endmodule
```

6.1.5 Correction

This erratum is not currently corrected.

7 DOCUMENTATION ERRATA

7.1 ETB AHB interface is little endian

ARM erratum reference: 752619.

This erratum affects ETB rev.0p0, rev.0p1 and rev.0p2.

7.1.1 Description

The AHB interface on the ETB is a little-endian interface. This means that if the ETB is used in a big-endian memory system and byte or half-word accesses are performed to the ETB RAM using the AHB interface then the bytes in the transfer must be rotated. This rotation should either be performed with on-chip logic, or by the software which is accessing the ETB RAM.

7.1.2 Implications

If the byte rotation is not performed then the data in the RAM might be incorrectly interpreted.

7.1.3 Workaround

System implementors must be aware that the ETB is a little endian device and must ensure that the correct byte rotations are performed for byte and half-word accesses.