AMBA™ 3 ATB Protocol

v1.0

Specification

ARM®

ARM IHI 0032A

AMBA 3 ATB Protocol
Specification

Release Information

Change history

Date	Issue	Change
19 June 2006	A	First release

Proprietary Notice

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

Preface
About this specification xii
Feedback xvi

Chapter 1 Introduction
1.1 About the protocol 1-2
1.2 About the interface 1-3

Chapter 2 Signal Descriptions
2.1 Global signals 2-2
2.2 Data signals 2-3
2.3 Flush control signals 2-4

Chapter 3 Flow Control
3.1 About flow control 3-2
3.2 Flow control signals 3-3
3.3 ATIDs 3-7

Chapter 4 Flushing
4.1 Buffer flush 4-2

Chapter 5 **Signal Rules**
5.1 Interface rules 5-2

Glossary

List of Tables
AMBA 3 ATB Protocol Specification

	Change history	ii
Table 2-1	Global signals	2-2
Table 2-2	Data signals	2-3
Table 2-3	Flush control signals	2-4
Table 3-1	ATB states	3-4
Table 3-2	Width relationship of ATDATA and ATBYTES	3-5
Table 4-1	Flush events	4-4
Table 4-2	Basic AFREADY control algorithm	4-4
Table 4-3	Events associated with flushing from a master	4-5

List of Figures
AMBA 3 ATB Protocol Specification

	Key to timing diagram conventions	xiv
Figure 1-1	ATB relationships	1-3
Figure 3-1	Flow of trace data between master and slave	3-3
Figure 3-2	Normal ATVALID and ATREADY flow control	3-4
Figure 4-1	Trace generation and output	4-2
Figure 4-2	Flush procedure	4-4
Figure 4-3	Flushing from a master that lacks internal storage	4-5

Preface

This preface introduces the *Advanced Microcontroller Bus Architecture* (AMBA) 3 *Advanced Trace Bus* (ATB) specification. It contains the following sections:

- *About this specification* on page xii
- *Feedback* on page xvi.

About this specification

This specification describes the AMBA 3 ATB protocol. All references to ATB in this specification refer to AMBA 3 ATB. The information in this document supersedes ATB information located in the *CoreSight Architecture Specification*.

Intended audience

This specification is written for the following target audiences:

- engineers who are familiar with ARM architecture
- hardware engineers integrating ATB protocol-compliant components into a design
- hardware engineers designing ATB protocol-compliant components
- advanced users of development tools that provide support for ATB protocol functionality.

This specification does not document the behavior of individual components unless they form a fundamental part of the architecture.

Using this specification

This specification contains the following chapters:

Chapter 1 *Introduction*

Read this chapter for an overview of the protocol and its interface.

Chapter 2 *Signal Descriptions*

Read this chapter for a description of the protocol signal descriptions. It contains information about naming conventions, signal descriptions and values.

Chapter 3 *Flow Control*

Read this chapter for a description of ATB protocol flow control. This chapter includes information about the flow control signals and the use of *Trace Data IDs* (ATIDs) to designate streams.

Chapter 4 *Flushing*

Read this chapter for a description of the flush process for the protocol. This chapter includes flush scenarios and examples of flush timing.

Chapter 5 *Signal Rules*

Read this chapter for a description of the protocol signal rules. It contains a listing of rules that govern the protocol.

Conventions

This section describes the conventions this specification uses:

- *Typographical*
- *Timing diagrams* on page xiv
- *Signal naming* on page xiv
- *Numbering* on page xv.

Typographical

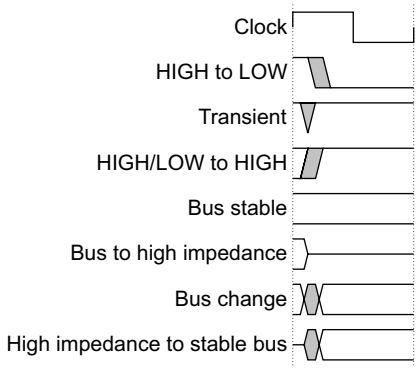
This specification uses the following typographical conventions:

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 >	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 specification contains one or more timing diagrams. The figure named *Key to timing diagram conventions* 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.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.



Key to timing diagram conventions

Signal naming

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 AMBA 3 *Advanced eXtensible Interface* (AXI) interface global and address channel signals.

Prefix AF Denotes signals relating to ATB flush control.

Prefix AT Denotes signals relating to ATB data flow.

Prefix B Denotes AXI interface write response channel signals.

Prefix C Denotes AXI interface low-power interface signals.

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

Prefix n Denotes active-LOW signals except in the case of AHB, *Advanced Peripheral Bus* (APB), or ATB interface reset signals. These are named **HRESETn**, **PRESETn**, and **ATRESETn** respectively.

Prefix P Denotes an APB interface signal.

Prefix R Denotes AXI interface read channel signals.

Prefix W Denotes AXI interface 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 by ARM Limited, and by third parties.

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

ARM publications

This specification contains information that is specific to the ATB protocol. See the following documents for other relevant information:

- *AMBA AXI Protocol Specification* (ARM IHI 0022).
- *AMBA 3 AHB-Lite Protocol Specification* (ARM IHI 0033)
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024)

Feedback

ARM Limited welcomes feedback on this protocol and its documentation.

Feedback on the protocol

Contact ARM Limited if you have comments or suggestions about the ATB protocol.

Feedback on this specification

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

- the title
- the number
- the relevant 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 ATB protocol. It contains the following sections:

- *About the protocol* on page 1-2
- *About the interface* on page 1-3.

1.1 About the protocol

The ATB protocol is part of the AMBA 3 protocol family.

The ATB protocol defines how trace information transfers between components in a trace system. The ATB is a common bus used by the trace components to pass format-independent trace data through a CoreSight system.

A trace component or platform that has trace capabilities requires an ATB interface. The ATB interfaces are designated according to one of two functions:

Master An interface that generates trace data on the ATB interface.

Slave An interface that receives trace data on the ATB interface.

The ATB interface supports various features, including:

- stalling of data, using valid and ready responses
- control signals to indicate the number of bytes valid in a cycle
- originating component marker, each data packet has an associated ID
- any trace protocol information, data information or data format requirements
- identification of data from all originating components
- flushing.

1.2 About the interface

The ATB is a common bus used by the trace components to pass trace data in a trace system in a data agnostic format. The ATB protocol defines the bus behavior and the interface signals are named according to their function.

Figure 1-1 shows the general relationships between the ATB and associated components.

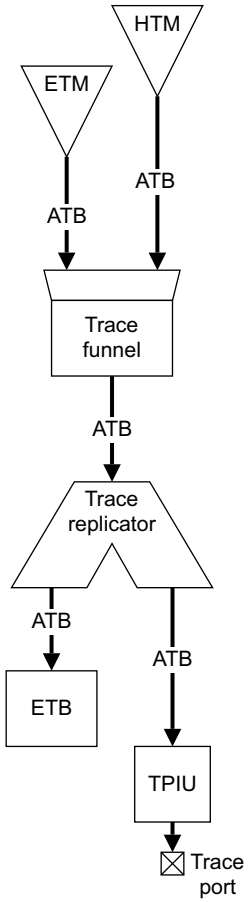


Figure 1-1 ATB relationships

The major components shown in Figure 1-1 are:

Embedded Trace Buffer (ETB)

This is as on-chip trace data storage device.

Embedded Trace Macrocell (ETM)

This outputs instruction and data trace information relating to a connected processor.

Advanced High-performance bus Trace Macrocell (HTM)

This outputs trace information relating to the AHB interconnect it is monitoring.

Trace Port Interface Unit (TPIU)

This is an intermediary device between ATB and a trace port for external storage of trace data.

Trace Replicator

This replicates a single ATB slave interface into two independent ATB masters

Trace funnel

This combines multiple trace sources onto a single bus.

Within a trace system, there are one or more sources of trace data, for example ETM or HTM. There are also one or more sinks, or receivers of the data. In Figure 1-1 on page 1-3 these sinks are ETB and TPIU. Between the trace sources and the sinks, there can be trace links that help manage the passage of data over ATB interfaces, such as the trace funnel and trace replicator.

The trace sources within the system generate ATB data traffic that can be managed by trace links on the ATB path that can re-transmit the information to the trace sinks. Trace sinks act as the ultimate receivers of the trace data generated by the trace sources and typically request a flush of the trace structure to ensure all trace data relating to a particular point in time has been received.

These flush requests propagate back up the ATB system from the trace sinks towards the trace sources which mark all historical data as occurring before the flush request and also indicate completion when all historical data is output. Any trace links present on the ATB system must pass the flush requests up to trace sources and ensure that they only acknowledge flush completion after all visible trace sources are acknowledged.

Chapter 2
Signal Descriptions

This chapter describes the ATB interface signals and rules. It contains the following sections:

- *Global signals* on page 2-2
- *Data signals* on page 2-3
- *Flush control signals* on page 2-4.

2.1 Global signals

Table 2-1 describes the global signals for a component implementing a trace data interface.

The signals are named according to their function. Clock and reset signals apply to both **AT** and **AF** types.

Table 2-1 Global signals

Name	Master	Slave	Description
ATCLK	Input	Input	Global ATB clock.
ATCLKEN	Input	Input	Enable signal for **ATCLK** domain.
ATRESETn	Input	Input	ATB interface reset when LOW. This signal is asserted LOW asynchronously, and deasserted HIGH synchronously.

2.2 Data signals

Table 2-2 describes the data signals for a component implementing a trace data interface, including the clamp value. The clamp values shown in the table reflect the values required when the component is powered down or disabled.

Table 2-2 Data signals

Name	Master	Slave	Clamp value	Description
ATBYTES[m:0][a]	Output	Input	LOW	The number of bytes on **ATDATA** to be captured, minus 1.
ATDATA[n:0]	Output	Input	LOW	Trace data.
ATID[6:0]	Output	Input	LOW	An ID that uniquely identifies the source of the trace. See *ATIDs* on page 3-7.
ATREADY	Input	Output	HIGH	Slave is ready to accept data. See Chapter 3 *Flow Control*.
ATVALID	Output	Input	LOW	A transfer is valid during this cycle. If LOW, all other **AT** signals must be ignored this cycle. See Chapter 3 *Flow Control*.

a. The letters *m* and *n* are explained in *Relationship between ATDATA and ATBYTES* on page 3-5.

2.3 Flush control signals

Table 2-3 lists the flush control signals for a component implementing a trace data interface, including the clamp value. The clamp values shown in the table reflect the values required when the component is powered down or disabled.

Table 2-3 Flush control signals

Name	Master	Slave	Clamp value	Description
AFVALID	Input	Output	LOW	This is the flush signal. All buffers must be flushed because trace capture is about to stop. See *Buffer flush* on page 4-2.
AFREADY	Output	Input	HIGH	This is a flush acknowledge. Asserted when buffers are flushed. See *Buffer flush* on page 4-2.

Chapter 3
Flow Control

Flow control uses the handshake between **ATVALID** and the **ATREADY** ATB protocol signals to transfer data and control information.

This chapter describes the ATB protocol flow control. It contains the following sections:

- *About flow control* on page 3-2
- *Flow control signals* on page 3-3
- *ATIDs* on page 3-7.

3.1 About flow control

This mechanism is a two-way flow control that enables the master and the slave to control the rate at which the data and control information moves. The source generates the **ATVALID** signal to indicate when the data or control information is available. The destination generates the **ATREADY** signal to indicate that it accepts the data or control information. Transfer of data only occurs when both the **ATVALID** and **ATREADY** signals are HIGH and can assign specific IDs to each data item in a cycle to identify streams.

3.2 Flow control signals

Figure 3-1 shows the flow of trace data information between the master and slave interfaces.

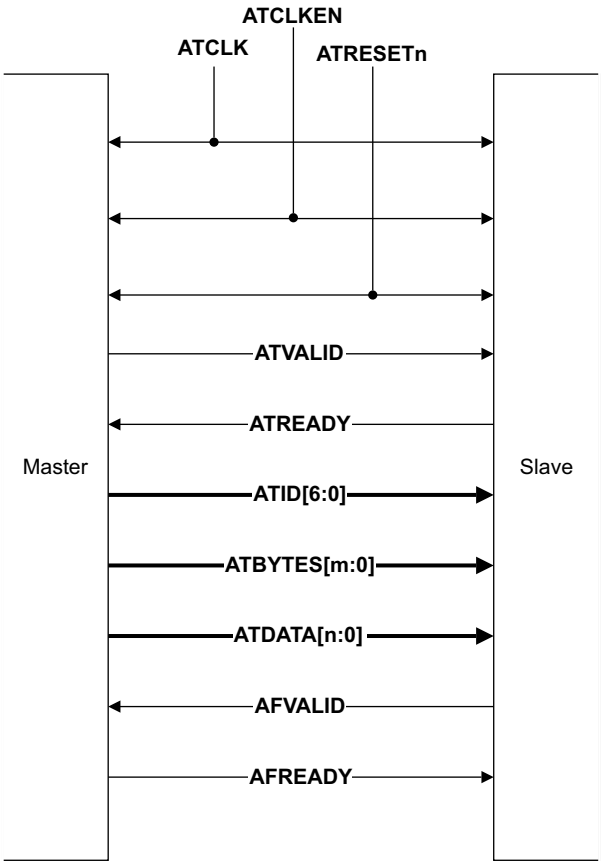


Figure 3-1 Flow of trace data between master and slave

ATVALID and **ATREADY** communicate between the master and the slave about trace data item availability.

The master indicates that it has trace available to output by asserting **ATVALID**.

If the slave can accept trace data, it responds by asserting **ATREADY**. If the slave cannot accept trace data, or does not send the **ATREADY** signal, the same trace must be output again on the next cycle. Figure 3-2 on page 3-4 shows this process.

A trace source can only start driving **ATVALID** HIGH after **ATRESETn** has been HIGH at a rising edge of **ATCLK**.

Although **ATRESETn** can be asserted LOW and asynchronously, deassertion must be synchronous after the rising edge of **ATCLK**.

If **ATREADY** is LOW and **ATVALID** is HIGH on a cycle, **ATVALID**, **ATDATA**, **ATID**, and **ATBYTES** must retain the same value on the next cycle.

If **ATVALID** is LOW, **ATREADY** must be ignored.

Figure 3-2 shows normal ATVALID and ATREADY flow control.

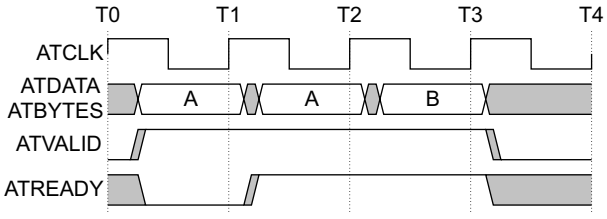


Figure 3-2 Normal ATVALID and ATREADY flow control

Table 3-1 shows the states of the ATB relationship to Figure 3-2.

Table 3-1 ATB states

Clock cycle	State
T1	Stalled, **ATREADY**
T2	A accepted
T3	B accepted
T4	Ignored, not valid

If a slave cannot respond with **ATREADY** during the same cycle that **ATVALID** is asserted, then it is recommended that you implement sufficient internal buffering in the slave. This internal buffering can store one or more cycles of trace. The slave can then assert **ATREADY** when space is available within the buffer, even when **ATVALID** is not asserted.

The following sections describe the interrelationship between **ATDATA**, **ATBYTES** and what happens if a master or slave cannot respond:

- *Relationship between ATDATA and ATBYTES*
- *Slave unable to respond* on page 3-6
- *Master unable to respond* on page 3-6.

3.2.1 Relationship between ATDATA and ATBYTES

The relationship between the widths of **ATBYTES[m:0]** and **ATDATA[n:0]** is described by the equation:

$m = \log_2(n+1) - 4$

Table 3-2 shows the relationship between **ATBYTES[m:0]** and **ATDATA[n:0].**

Table 3-2 Width relationship of ATDATA and ATBYTES

ATDATA[n:0]	ATBYTES[m:0]
n = 7	**ATBYTES** not required
n = 15	m = 0
n = 31	m = 1
n = 63	m = 2
n = 127	m = 3

For example, if **ATDATA** is 32 bits wide, then **ATBYTES** is 2 bits wide.

If **ATREADY** and **ATVALID** are HIGH, then the bottom bytes of **ATDATA** must be captured and the data must be aligned to the least significant bit.

Note

Zero bytes cannot be captured.

The width of **ATDATA** must be a power of two equal to or greater than eight bits.

It is recommended that components use at least a 32-bit implementation.

3.2.2 Slave unable to respond

If a slave is incapable of responding, then it must drive **ATREADY** HIGH and **AFVALID** LOW.

Note

This ensures a replicator does not block because one of its two outputs are disabled.

Examples of this include:

- when a slave is powered down
- when a slave is not present
- the result of programming.

3.2.3 Master unable to respond

If an ATB interface master is incapable of responding, then it must drive **AFREADY** HIGH and **ATVALID** LOW.

Examples of this include:

- when a master is powered down
- when a master is not present
- the result of programming.

3.3 ATIDs

Trace data items are generated with separate IDs. These separate IDs provide:

- differentiation between trace from different sources
- the ability to distinguish between high and low bandwidth components of a trace, so components downstream from the trace source can perform selective filtering
- alignment of synchronization information by changing the ID at an alignment synchronization point, such as the beginning of a trace packet.

Most sources use a single, static ID.

Note

The **ATID** values `0x00` and `0x70-0x7F` are reserved values for use within the CoreSight Architecture and must not be used.

Because the ID for each trace stream must be unique, there are two options:

Fixed IDs The IDs for all ATB interface sources are chosen when designing the system, and no ATB interfaces are exported from the system.

Programmed IDs

Because the ID for each ATB interface source is programmable by the debugger, components can be reused in larger systems.

This option is mandatory for reusable CoreSight components.

Ensure that the **ATDATA** value is captured at the same time as bytes on **ATID** are captured.

3.3.1 Trace byte order

It is imperative to preserve the order of trace bytes, even between trace with different **ATID**s.

Although a CoreSight trace funnel can order trace from different sources in any order, when funneled onto a single bus the order cannot be changed, see *Buffer flush* on page 4-2.

This differs from ID signals used in AXI interfaces where ordering is preserved only between transactions with the same ID.

Chapter 4
Flushing

Because of the characteristics inherent in the buffering of trace data items, it is often necessary to remove remaining data from the buffer to prepare for new data as cycles progress. This process of removing data from a buffer is called flushing.

This chapter describes the flush process. It contains the following section:

- *Buffer flush* on page 4-2.

4.1 Buffer flush

In a typical trace source there is a fixed amount of time and number of pipeline stages between the occurrence of an event to be traced, and the trace generation for that event. An example of this is an instruction executed by the processor, Figure 4-1 shows an example of a trace generation.

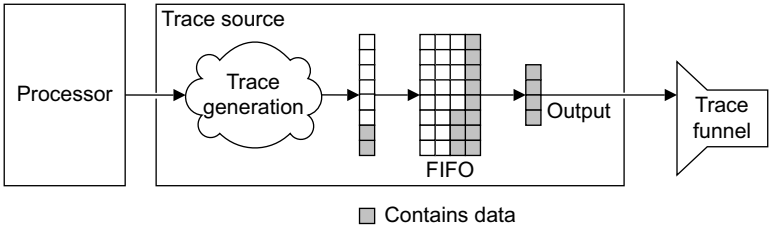


Figure 4-1 Trace generation and output

When a trace data item is generated, it is written to a FIFO. It is then only output a whole dataword at a time. The width of the **ATDATA** bus defines the size of the dataword.

The period of time that exists between the trace generation and its output is, in theory, unlimited. An example of this potentially extended period of time is when the trace source withholds output until a whole dataword of trace data is available for output.

There are various situations associated with the whole dataword of trace data feature that might require a buffer flush:

- The system, or a portion of the system, is about to be powered down or its clock is about to be stopped.

  Because any trace remaining within buffers or FIFOs in trace sources must be output before powering down or stopping, the **AFVALID** signal signals a flush. Normally, no additional trace is generated because processor and memory activity has stopped. If trace is still generated after the flush it can be ignored. An example of trace data generated after a flush is that from a processor idle loop.

- The trace capture device, an off-chip *Trace Port Analyzer* (TPA) or an on-chip *Embedded Trace Buffer* (ETB), is about to stop capturing, usually because of a trigger point.

  In this case the possibilities include:

  — The on-chip logic is signaled that capture is about to stop.

  A flush is issued at this point.

  — The on-chip logic is signaled about the trigger but does not have information about how much additional trace is to be captured.

A flush can be issued at the point of the trigger. This ensures all trace generated before the trigger is captured, although this makes no difference to trace generated after the trigger.

— A flush is issued periodically.

The period defined for the flush must be so that a flush always occurs between a trigger occurrence and the stoppage of the trace capture.

When a flush occurs, indicated by **AFVALID** HIGH, it is expected that trace funnels give the highest priority to trace sources that have not yet asserted **AFREADY**.

Note

- A flush sequence can not be cancelled by the trace slave (sink) after it begins.
- Ensure that **ATB** signals are sampled on the rising edge of **ATCLK** and that **AFVALID** remains asserted until **AFREADY** is deasserted.

AFREADY is asserted when all trace generated at the time **AFVALID** was asserted has been output, not when the FIFO is next empty.

Trace that is generated after **AFVALID** is asserted is stored in the FIFO and is output after **AFREADY** is asserted.

When **AFVALID** is asserted, then buffered trace must begin output immediately.

When **AFREADY** is asserted, then **AFVALID** must be deasserted on the following cycle, unless an additional flush is intended.

When **AFVALID** is asserted, then **AFREADY** must be asserted one cycle after all the trace has been output that was generated and stored within internal buffers on the cycle in which **AFVALID** was first asserted. For additional information, see *Buffer flush* on page 4-2.

Figure 4-2 on page 4-4 shows an example flush procedure.

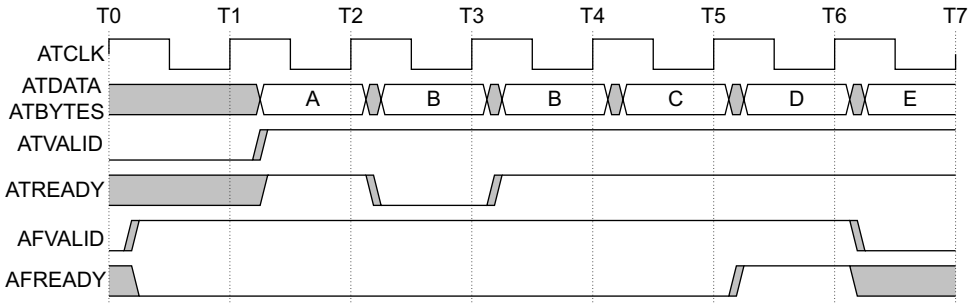


Figure 4-2 Flush procedure

Table 4-1 describes the events that occur during the flush procedure as Figure 4-2 shows.

Table 4-1 Flush events

Clock cycle	Event
T1-T2	Flush requested. A, B, C, in FIFO.
T2	Draining begins.
T3-T5	Stalls still possible.
T6	Flush complete. Output continues.
T6	**AFVALID** deasserted.

4.1.1 Behavior of trace sources that lack internal storage

If an ATB master interface is a trace source and does not store any trace internally, it uses the information in Table 4-2 to control **AFREADY**.

Table 4-2 Basic AFREADY control algorithm

ATVALID	ATREADY	AFREADY next cycle
0	-	1
1	0	0
1	1	1

Figure 4-3 shows the **AFREADY** control used when the master lacks internal storage.

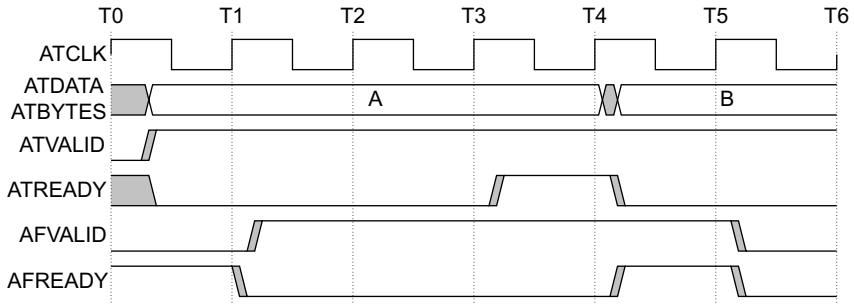


Figure 4-3 Flushing from a master that lacks internal storage

Table 4-3 describes the events that occur during the flush procedure from a master that lacks internal storage as Table 4-2 shows.

Table 4-3 Events associated with flushing from a master

Clock cycle	Event
T1	Data (A) generated in the master
T2	Flush request, data (A) is now old data
T3	Flush is not completed because old data (A) is still present in the master
T4	Data (A) is accepted by the slave
T5-T6	New data (B) is generated in the master, flush request acknowledged

Chapter 5
Signal Rules

This chapter describes the ATB interface signal rules. It contains the following section:

- *Interface rules* on page 5-2.

5.1 Interface rules

The ATB protocol conforms to the following rules:

- Ensure that **ATB** signals are sampled on the rising edge of **ATCLK**.
- If **ATREADY** is LOW and **ATVALID** is HIGH on a cycle, **ATVALID**, **ATDATA**, **ATID**, and **ATBYTES** must retain the same value on the next cycle.

  If **ATVALID** is LOW, **ATREADY** must be ignored.
- If **ATREADY** and **ATVALID** are HIGH, the bottom bytes of **ATDATA** must be captured.

  Align the data to the least significant byte.

  Note

  Zero bytes cannot be captured.
- The width of **ATDATA** must be a power of two equal to or greater than eight bits.

  It is recommended that components use at least a 32-bit implementation.
- $m = \log_2(n+1) - 4$

  This equation describes the relationship between the widths of **ATBYTES[m:0]** and **ATDATA[n:0]**.

  For example, if **ATDATA** is 32 bits wide, **ATBYTES** is 2 bits wide.
- Ensure that the **ATDATA** value is captured at the same time as bytes on **ATID** are captured.
- Preserve the order of trace bytes, even between trace with different **ATIDs**.

  A CoreSight trace funnel can order trace from different sources in any order, but when funneled onto a single bus, the order cannot be changed, see *Buffer flush* on page 4-2.

  Note

  This differs from ID signals used in AXI interfaces where ordering is preserved only between transactions with the same ID.
- Ensure that **AFVALID** remains asserted until **AFREADY** is asserted.

  AFREADY is ignored while **AFVALID** is deasserted.
- When **AFVALID** is asserted, buffered trace must begin output immediately.

- When **AFREADY** is asserted, **AFVALID** must be deasserted on the following cycle, unless an additional flush is intended.
- When **AFVALID** is asserted, **AFREADY** must be asserted one cycle after all the trace has been output that was generated and stored within internal buffers on the cycle in which **AFVALID** was first asserted. See *Buffer flush* on page 4-2.
- Although **ATRESETn** can be asserted, LOW and asynchronously, deassertion must be synchronous after the rising edge of **ATCLK**.
- A trace source can only start driving **ATVALID** HIGH after **ATRESETn** has been HIGH at a rising edge of **ATCLK**.
- If an ATB interface slave is incapable of responding, it must drive **ATREADY** HIGH and **AFVALID** LOW. Examples of this include powering down, not being present, or the result of programming.

  This ensures a replicator does not block because one of its two outputs are disabled.
- If an ATB interface master is incapable of responding, it must drive **AFREADY** HIGH and **ATVALID** LOW. Examples of this include powering down, not being present, or the result of programming.

  ATID, **ATBYTES**, and **ATDATA** can have any value. Implement this behavior to tie off unused CoreSight trace funnel slave signals.
- The **ATID** values `0x00` and `0x70-0x7F` are reserved values within the CoreSight Architecture and must not be used.

Glossary

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

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 Limited 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.

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 details a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

AHB *See* Advanced High-performance Bus.

AHB-AP *See* AHB Access Port.

AMBA *See* Advanced Microcontroller Bus Architecture.

Advanced Trace Bus (ATB)
A bus used by trace devices to pass trace data around a CoreSight system in a trace protocol agnostic manner.

ATB *See* Advanced Trace Bus.

AXI *See* Advanced eXtensible Interface.

Byte An 8-bit data item.

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

Data cache A block of on-chip fast access memory locations, situated between the processor and main memory, used for storing and retrieving copies of often used data. This is done to greatly increase the average speed of memory accesses and so improve processor performance.

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

Embedded Trace Macrocell (ETM)
A hardware macrocell that, when connected to a processor core, outputs instruction and data trace information on a trace port. The ETM provides processor driven trace through a trace port compliant to the ATB protocol.

Macrocell A complex logic block with a defined interface and behavior. A typical VLSI system comprises several macrocells (such as a processor, an ETM, and a memory block) plus application-specific logic.

Microprocessor *See* Processor.

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.

Trace funnel A device that combines multiple trace sources onto a single bus.

Trace port A port on a device, such as a processor or ASIC, used to output trace information.

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.

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

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