AMBA® ATB Protocol Specification

arm

ARM IHI 0032C (ID120321)

AMBA ATB Protocol Specification

Release Information

Change history

Date	Issue	Confidentiality	Change
19 June 2006	A	Non-Confidential	First release, AMBA3, ATB version 1.0
28 March 2012	B	Non-Confidential	Second release, AMBA 4, ATB version 1.1
01 December 2021	C	Non-Confidential	Third release

Proprietary Notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
LES-PRE-21451 version 2.2

AMBA SPECIFICATION LICENCE

THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED ("ARM") FOR THE USE OF ARM'S INTELLECTUAL PROPERTY (INCLUDING, WITHOUT LIMITATION, ANY COPYRIGHT) IN THE RELEVANT AMBA SPECIFICATION ACCOMPANYING THIS LICENCE. ARM LICENSES THE RELEVANT AMBA SPECIFICATION TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING "I AGREE" OR OTHERWISE USING OR COPYING THE RELEVANT AMBA SPECIFICATION YOU INDICATE THAT YOU AGREE TO BE BOUND BY ALL THE TERMS OF THIS LICENCE.

"LICENSEE" means You and your Subsidiaries.

"Subsidiary" means, if You are a single entity, any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by You. A company shall be a Subsidiary only for the period during which such control exists.

1. Subject to the provisions of Clauses 2, 3 and 4, Arm hereby grants to LICENSEE a perpetual, non-exclusive, non-transferable, royalty free, worldwide licence to:

   (i) use and copy the relevant AMBA Specification for the purpose of developing and having developed products that comply with the relevant AMBA Specification;

   (ii) manufacture and have manufactured products which either: (a) have been created by or for LICENSEE under the licence granted in Clause 1(i); or (b) incorporate a product(s) which has been created by a third party(s) under a licence granted by Arm in Clause 1(i) of such third party's AMBA Specification Licence; and

   (iii) offer to sell, sell, supply or otherwise distribute products which have either been (a) created by or for LICENSEE under the licence granted in Clause 1(i); or (b) manufactured by or for LICENSEE under the licence granted in Clause 1(ii).

2. LICENSEE hereby agrees that the licence granted in Clause 1 is subject to the following restrictions:

   (i) where a product created under Clause 1(i) is an integrated circuit which includes a CPU then either: (a) such CPU shall only be manufactured under licence from Arm; or (b) such CPU is neither substantially compliant with nor marketed as being compliant with the Arm instruction sets licensed by Arm from time to time;

   (ii) the licences granted in Clause 1(iii) shall not extend to any portion or function of a product that is not itself compliant with part of the relevant AMBA Specification; and

   (iii) no right is granted to LICENSEE to sublicense the rights granted to LICENSEE under this Agreement.

3. Except as specifically licensed in accordance with Clause 1, LICENSEE acquires no right, title or interest in any Arm technology or any intellectual property embodied therein. In no event shall the licences granted in accordance with Clause 1 be construed as granting LICENSEE, expressly or by implication, estoppel or otherwise, a licence to use any Arm technology except the relevant AMBA Specification.

4. THE RELEVANT AMBA SPECIFICATION IS PROVIDED "AS IS" WITH NO REPRESENTATION OR WARRANTIES EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF SATISFACTORY QUALITY, MERCHANTABILITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE, OR THAT ANY USE OR IMPLEMENTATION OF SUCH ARM TECHNOLOGY WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

5. NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS AGREEMENT, TO THE FULLEST EXTENT PETMITTED BY LAW, THE MAXIMUM LIABILITY OF ARM IN AGGREGATE FOR ALL CLAIMS MADE AGAINST ARM, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS AGREEMENT (INCLUDING WITHOUT LIMITATION (I) LICENSEE'S USE OF THE ARM TECHNOLOGY; AND (II) THE IMPLEMENTATION OF THE ARM TECHNOLOGY IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS AGREEMENT) SHALL NOT EXCEED THE FEES PAID (IF ANY) BY LICENSEE TO ARM UNDER THIS AGREEMENT. THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

6. No licence, express, implied or otherwise, is granted to LICENSEE, under the provisions of Clause 1, to use the Arm tradename, or AMBA trademark in connection with the relevant AMBA Specification or any products based thereon. Nothing in Clause 1 shall be construed as authority for LICENSEE to make any representations on behalf of Arm in respect of the relevant AMBA Specification.

7. This Licence shall remain in force until terminated by you or by Arm. Without prejudice to any of its other rights if LICENSEE is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to You. You may terminate this Licence at any time. Upon expiry or termination

of this Licence by You or by Arm LICENSEE shall stop using the relevant AMBA Specification and destroy all copies of the relevant AMBA Specification in your possession together with all documentation and related materials. Upon expiry or termination of this Licence, the provisions of clauses 6 and 7 shall survive.

8. The validity, construction and performance of this Agreement shall be governed by English Law.

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
AMBA ATB Protocol Specification

Preface

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

About this book

This specification describes the AMBA ATB protocol. All references to ATB in this specification refer to AMBA 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 audience:

- 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 book

This specification is organized into 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 connection of an ATB Transmitter to an ATB Receiver, and the use of flow control signals to control data transfer between the Transmitter and Receiver.

Chapter 4 *Other ATB Features*

Read this chapter for a description of the other features of an ATB implementation. It describes the use of *Trace Data IDs* (ATIDs), the flush process for the protocol, and trace trigger and synchronization signaling added in ATB-B (v1.1).

Chapter 5 *Interface Signaling Rules*

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

Chapter 6 *Wake-up signaling*

Read this for a description of wake-up signaling. This is used to indicate any activity on the ATB interface.

Appendix A *Signal matrix*

Read this for an overview of the ATB interface signals.

Appendix B *Revisions*

Read this for a description of the technical changes between released issues of this book.

Glossary

Read this for definitions of some terms used in this book. The glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

Conventions

The following sections describe conventions that this book can use:

- *Typographic conventions*
- *Signal naming*
- *Timing diagrams* on page x
- *Numbers* on page xi

Typographic conventions

The typographical conventions are:

italic

Introduces special terminology, denotes internal cross-references and citations, or highlights an important note.

bold

Denotes signal names, and is used for terms in descriptive lists, where appropriate.

`monospace`

Used for assembler syntax descriptions, pseudocode, and source code examples.

Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.

SMALL CAPITALS

Used for a few terms that have specific technical meanings.

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.

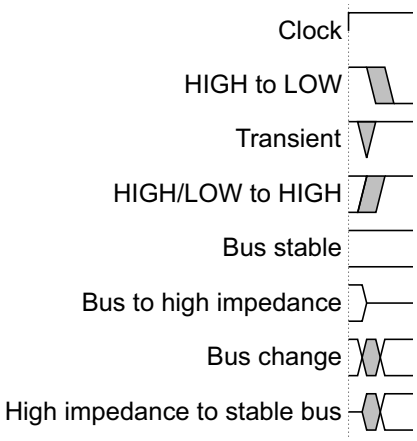
Prefix R Denotes AXI interface read channel signals.

Prefix W Denotes AXI interface write channel signals.

Timing diagrams

The components used in timing diagrams are explained in the figure *Key to timing diagram conventions*. Variations have clear labels, when they occur. Do 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

Timing diagrams sometimes show single-bit signals as HIGH and LOW at the same time and they look similar to the bus change shown in *Key to timing diagram conventions*. If a timing diagram shows a single-bit signal in this way then its value does not affect the accompanying description.

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by `0b`, and hexadecimal numbers by `0x`. In both cases, the prefix and the associated value are written in a `monospace` font, for example `0xFFFF0000`.

Additional reading

This section lists relevant publications from Arm.

Arm publications

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

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

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an e-mail to errata@arm.com. Give:

- The title, AMBA ATB Protocol Specification
- The number, *ARM IHI 0032C*
- The page numbers to which your comments apply
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Progressive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have terms that can be offensive.

Arm strives to lead the industry and create change.

Previous issues of this document included terms that can be offensive. We have replaced these terms. If you find offensive terms in this document, please contact terms@arm.com.

Chapter 1
Introduction

This chapter introduces the ATB protocol. It contains the following sections:

1.1 About the ATB protocol

The ATB protocol is part of the AMBA protocol family:

- ATBv1.0 is defined in Issue A of this specification, as part of AMBA 3.
- ATBv1.1 is defined in Issue B of this specification, as part of AMBA 4.
- ATB-C is defined in Issue C of this specification, as part of AMBA 5.

The ATB protocol defines how trace information transfers between components in a trace system. ATB is a common bus used by 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 by either:

Transmitter An interface that generates trace data onto the ATB bus.

Receiver An interface that receives trace data from the ATB bus.

The ATB interface supports various features, including:

- Stalling of data, using valid and ready responses.
- Control signals that indicate the number of bytes valid in a cycle.
- Identification of the originating component, by signaling an associated ID with each data packet.
- Support for any trace protocol information, data information or data format requirements.
- Flushing.

1.2 About the ATB interface

The ATB is a common bus used by the trace components to pass trace data in a 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.

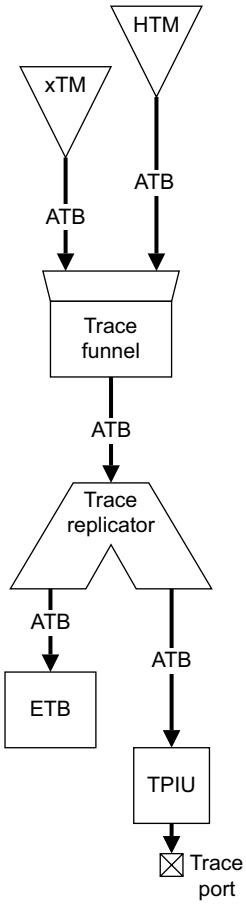


Figure 1-1 ATB relationships

The major components shown in Figure 1-1 are:

Advanced High-performance bus Trace Macrocell (HTM)

This outputs trace information describing the AHB interconnect it is monitoring.

Trace Macrocell (xTM)

This generates and outputs trace information from a connected processor. Arm trace macrocells include:

- The *Embedded Trace Macrocell* (ETM). An ETM generates instruction trace and could generate data trace.
- The *Program-flow Trace Macrocell* (PTM). A PTM generates instruction trace.

Trace funnel

This combines multiple trace sources onto a single bus.

Trace replicator

This replicates a single ATB Receiver interface into two independent ATB Transmitters.

Embedded Trace Buffer (ETB)

This is as on-chip trace data storage device.

Trace Port Interface Unit (TPIU)

This is a device that connects ATB to a trace port to provide external storage of trace data.

A trace system includes:

- At least one source of trace data, for example ETM or HTM.
- At least one trace sink or receiver of the data. In Figure 1-1 on page 1-17, the 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 generate ATB data traffic that can be managed by trace links on the ATB path. Trace links can re-transmit information to the trace sinks. Trace sinks act as the receivers of the trace data generated by the trace sources.

Typically, a trace sink can request a flush of the trace structure to ensure all trace data, up to a particular time, has been received. These flush requests propagate back up the ATB system from the trace sinks towards the trace sources. On receiving a flush request, a trace source marks all data that occurred before the flush request, and indicates completion when it has output all of this data. Any trace link in the ATB system must:

- Pass flush requests up to its connected trace sources.
- Ensure that it acknowledges flush completion only after all connected trace sources have acknowledged completion.

Chapter 2
Signal Descriptions

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

2.1 ATB signals

This following sections describe the ATB interface signals, by functional group, and any signaling rules:

- *Global signals*
- *Data signals*
- *Flush control signals* on page 2-21
- *Synchronization request signal* on page 2-21

The tables in the following sections include a clamp value of the signal when the component is powered down or disabled.

2.1.1 Global signals

Table 2-1 shows the global signals for a component implementing an ATB trace data interface.

Table 2-1 Global signals

Signal	Transmitter	Receiver	Description
ATCLK	Input	Input	Global ATB clock.
ATCLKEN	Input	Input	Enable signal for **ATCLK** domain. From ATB Issue B, this signal is optional.
ATRESETn	Input	Input	ATB interface reset, active LOW. This signal is asserted LOW asynchronously, and deasserted HIGH synchronously.

Clock and reset signals apply to both **AT** and **AF** signal types.

2.1.2 Data signals

Table 2-2 shows the data signals for a component implementing an ATB trace data interface.

Table 2-2 Data signals

Signal	Transmitter	Receiver	Clamp value	Description
ATBYTES[m:0][a]	Output	Input	LOW	The number of bytes on **ATDATA** to be captured, minus 1.
ATDATA[n:0][a]	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 4-30.
ATREADY	Input	Output	HIGH	The Receiver 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. See Chapter 3 *Flow Control*.

a. *Relationship between ATDATA and ATBYTES* on page 3-26 explains the relationship between the values of m and n.

2.1.3 Flush control signals

Table 2-3 shows the flush control signals for a component implementing an ATB trace data interface.

Table 2-3 Flush control signals

Signal	Transmitter	Receiver	Clamp value	Description
AFVALID	Input	Output	LOW	The flush signal to indicate that all buffers must be flushed because trace capture is about to stop. See *Buffer flush* on page 4-31.
AFREADY	Output	Input	HIGH	The flush acknowledge to indicate that buffers have been flushed. See *Buffer flush* on page 4-31.

2.1.4 Synchronization request signal

Table 2-4 shows the recommended synchronization request signal on an ATB implementation.

Table 2-4 Synchronization request signals

Signal	Transmitter	Receiver	Clamp value	Description
SYNCREQ	Input	Output	LOW	The synchronization request signal to request the insertion of synchronization information in the trace stream. See *Buffer flush* on page 4-31.

2.1.5 Wake-up signal

Table 2-5 shows the wake-up signal for an ATB interface.

Table 2-5 Wake-up signal

Signal	Transmitter	Receiver	Description
ATWAKEUP	Input	Output	The wake-up signal indicating any activity associated with an ATB interface. See Chapter 6 *Wake-up signaling*.

Chapter 3
Flow Control

This chapter describes the ATB protocol flow control which uses the **ATVALID** and **ATREADY** signals and shows the interconnection of a Transmitter and a Receiver. It contains the following sections:

3.1 About flow control

The ATB flow control mechanism is a two-way flow control that the Transmitter and the Receiver can use 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 occurs only when both the **ATVALID** and **ATREADY** signals are HIGH.

3.2 ATB connections and flow control signaling

Figure 3-1 shows the interconnection of a Transmitter and a Receiver.

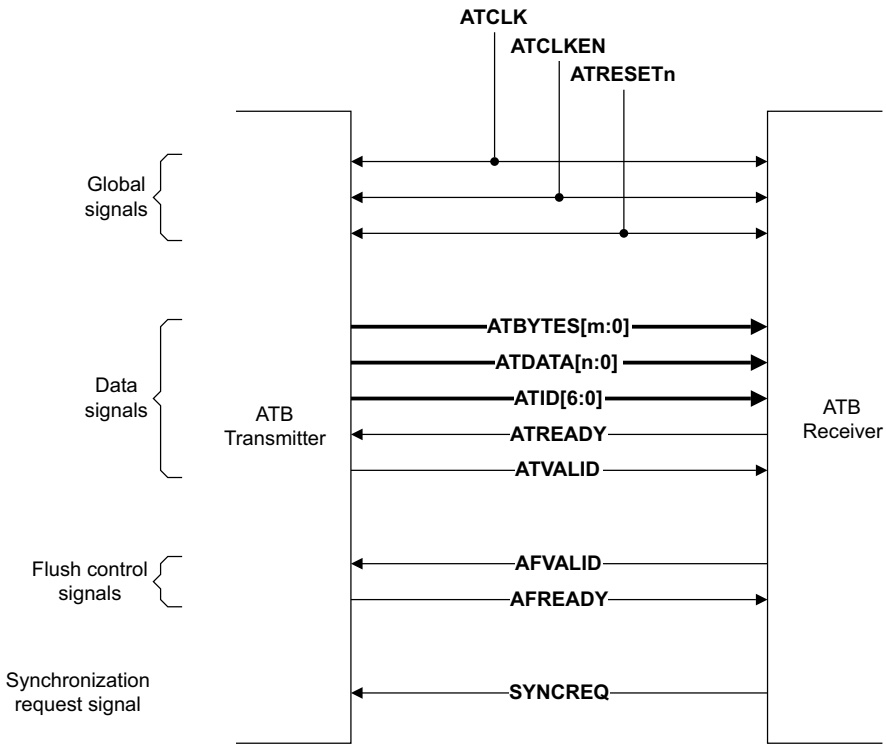


Figure 3-1 ATB signaling between Transmitter and Receiver

ATVALID and **ATREADY** provide a handshake between the Transmitter and the Receiver, to indicate trace data item availability:

- The Transmitter indicates that it has trace available to output by asserting **ATVALID**.
- If the Receiver can accept trace data, it responds by asserting **ATREADY**.

  If the Receiver cannot accept trace data, or does not assert the **ATREADY** signal, the same trace must be output again on the next cycle. Figure 3-2 shows this process.

Figure 3-2 shows the **ATVALID** and **ATREADY** flow control.

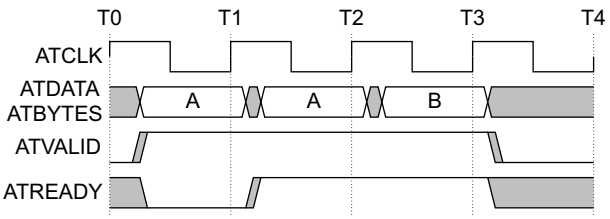


Figure 3-2 ATVALID and ATREADY flow control signaling

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 asynchronously, deassertion must be synchronous with a 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.

Table 3-1 shows the status indicated by the ATB signals at each of the **ATCLK** rising edges in Figure 3-2 on page 3-25.

Table 3-1 ATB status

Clock cycle	State
T1	Trace stalled, **ATREADY** not asserted
T2	Value A accepted
T3	Value B accepted
T4	Trace ignored, not valid

If an implementation of a Receiver cannot guarantee that it can respond with **ATREADY** during the same cycle that **ATVALID** is asserted, then it is recommended that the Receiver implements sufficient internal buffering. This internal buffering must be sufficient to store one or more cycles of trace. The Receiver can then assert **ATREADY** when space is available in the buffer, even when **ATVALID** is not asserted.

The following sections describe the relationship between **ATDATA** and **ATBYTES,** and what happens if a Transmitter or Receiver cannot respond:

- *Relationship between ATDATA and ATBYTES*
- *Receiver unable to respond* on page 3-27
- *Transmitter unable to respond* on page 3-27

3.2.1 Relationship between ATDATA and ATBYTES

The following equation defines the relationship between the widths of **ATBYTES** and **ATDATA**, defined by the values of m and n in **ATBYTES[m:0]** and **ATDATA[n:0]**:

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

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

Table 3-2 Width relationship between 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 (n=31), then **ATBYTES** is 2 bits wide (m=1).

If **ATREADY** and **ATVALID** are both 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 8 bits.

3.2.2 Receiver unable to respond

If a Receiver interface cannot respond, it must drive **ATREADY** HIGH and **AFVALID** LOW.

Note

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

Examples of when a Receiver cannot respond include:

- The Receiver is powered down.
- A Receiver is not present.
- The system has been wrongly programmed.

3.2.3 Transmitter unable to respond

If a Transmitter interface cannot respond, it must drive **AFREADY** HIGH and **ATVALID** LOW.

Examples of when a Transmitter cannot respond include:

- The Transmitter is powered down.
- A Transmitter is not present.
- The system has been wrongly programmed.

Chapter 4
Other ATB Features

This chapter describes the features of the ATB specification that are outside the scope of Chapter 2 *Signal Descriptions* and Chapter 3 *Flow Control*. It contains the following sections:

4.1 ATIDs

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

- Differentiation between trace from different sources.
- Identification of high bandwidth 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.

ATB trace data items can use:

- Any ATID values in the range `0x01-0x6F`
- From ATB-B (v1.1) and onwards, ATID value `0x7D`, but only to generate a trace trigger, see *Signaling a trace trigger* on page 4-34.

Note

The **ATID** values `0x00`, `0x70-0x7C`, `0x7E`, and `0x7F` are reserved for use by the CoreSight Architecture and must not be used as ATB IDs.

There are two options to ensure different trace streams use unique ID values:

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

The ID for each ATB interface source is programmable by the debugger, allowing components to be reused in larger systems.

Reusable CoreSight components must implement Programmed IDs.

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

4.1.1 Trace byte order

An ATB implementation must 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 4-31.

Note

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

4.2 Buffer flush

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

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.

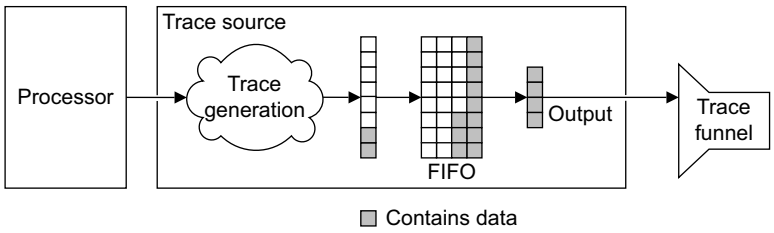


Figure 4-1 Trace generation and output

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

The period of time between the trace generation and output is, in theory, unlimited because the FIFO might never receive enough trace data to fill a whole output dataword. An example of a possibly extended delay between generation and output is when the trace source withholds output until it has a whole dataword of trace data available for output.

In a system where data is output only when a whole dataword is available, there are various situations that might require a buffer flush. For example:

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

  Any trace remaining within buffers or FIFOs in trace sources must be output before powering down or stopping the clock. Therefore, the **AFVALID** signal signals a flush. Normally, after signaling the flush, no more trace is generated because processor and memory activity has stopped. If trace is 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 is about to stop capturing, usually because of a trigger point.

  The trace capture device might be an off-chip *Trace Port Analyzer* (TPA) or an on-chip *Embedded Trace Buffer* (ETB).

  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 any 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, but makes no difference to trace generated after the trigger.

  — A flush is issued periodically.

    The period defined for the flush must be chosen so that a flush always occurs between a trigger occurring and the trace capture stopping.

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

Note

- A flush sequence can not be canceled by a Receiver interface after it initiates. This means it cannot be canceled by the trace sink.
- An implementation must ensure that ATB signals are sampled on the rising edge of **ATCLK**. The **AFVALID** signal remains asserted until **AFREADY** is deasserted.

Figure 4-2 shows an example of an ATB flush.

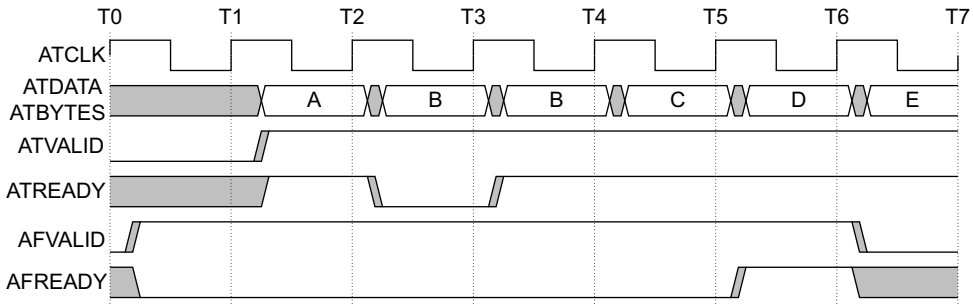


Figure 4-2 Flush signaling

AFREADY is asserted when all trace generated at the time **AFVALID** was asserted has been output. This assertion of **AFREADY** does not indicate that the FIFO is empty.

Trace, which is generated after **AFVALID** is asserted, is stored in the FIFO. This trace is then output after **AFREADY** is asserted.

When **AFVALID** is asserted, the Transmitter must start outputting buffered trace immediately.

When the Transmitter asserts **AFREADY**, if an additional flush is not required, the Receiver must deassert **AFVALID** on the following cycle.

When **AFVALID** is asserted, the Transmitter must assert **AFREADY** one cycle after the output data that was buffered on the cycle in which **AFVALID** was first asserted. That is, it must assert **AFREADY** one cycle after it has output the last of the data that the assertion of **AFVALID** required it to flush. For more information, see *Buffer flush* on page 4-31.

Table 4-1 describes the events that occur during the ATB flush shown in Figure 4-2.

Table 4-1 Flush events

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

4.2.1 Behavior of trace sources that do not store trace locally

If a Transmitter interface is a trace source that does not store any trace locally, Table 4-2 shows how it controls **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 this **AFREADY** control for a Transmitter that does not store any trace locally.

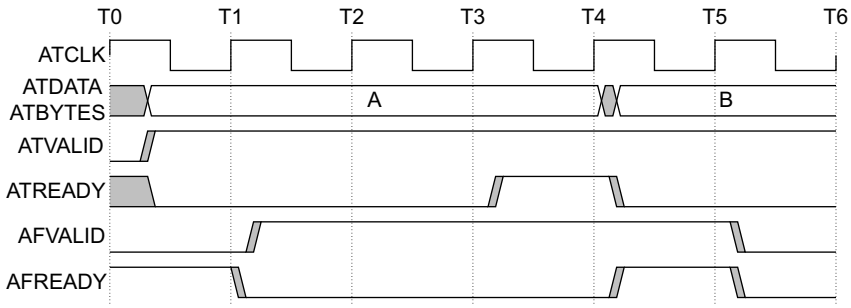


Figure 4-3 Flushing from a Transmitter that lacks internal storage

Table 4-3 describes the events that occur during the flush from a Transmitter that does not store any trace locally shown in Figure 4-3.

Table 4-3 Events associated with flushing from a Transmitter that does not store trace locally

Clock edge	Event
T1	Value A generated in the Transmitter
T2	Flush request, Value A is now old data
T3	Flush is not completed because old Value A is still present in the Transmitter
T4	Value A is accepted by the Receiver
T5-T6	New Value B is generated in the Transmitter, flush request acknowledged

4.3 Signaling a trace trigger

A trace trigger signal is an IMPLEMENTATION DEFINED message from a trace source to the trace sink. Typically, it might be used to indicate that a significant condition has occurred and that trace capture should prepare to stop.

In ATB Issue B, a Transmitter can signal a trace trigger as follows:

- The **ATID** value must be `0x7D`.
- **ATBYTES** indicates the number of trace triggers output in this cycle. It is recommended that, wherever possible, only one trace trigger is generated at a time. In this case:
  - There is only one byte of data.
  - **ATBYTES** is zero.
- **ATDATA** indicates the source of each signaled trace trigger, as follows:
  - Each byte of data indicates a separate trace trigger.
  - A byte of data that is zero indicates a trigger from an unknown source. It is recommended that trace sources do not use this value.
  - A byte of data that is non-zero indicates the ID of the trace source that generated the trace trigger.

For example:

- A trace source that uses ATID `0x10` for its ATB signaling can indicate a trigger on its ATB interface by generating a byte of trace data with:
  - **ATID** signaling `0x7D`, indicating a trace trigger.
  - **ATBYTES** signaling `0x0`, indicating a single data byte.
  - **ATDATA** signaling `0x10`, indicating the trace source ID.
- A trace stream that uses ATIDs `0x10` and `0x12` can indicate simultaneous trace triggers on both trace sources by generating two bytes of trace data with:
  - **ATID** signaling `0x7D`, indicating a trace trigger.
  - **ATBYTES** signaling `0x1`, indicating two data bytes.
  - **ATDATA** signaling `0x1210`, indicating the trace source IDs of `0x10` and `0x12`.

4.4 Synchronization request

Optionally, an ATB implementation can include the **SYNCREQ** synchronization request signal, see *Synchronization request signal* on page 2-21.

A Receiver signals a synchronization request to a Transmitter by asserting **SYNCREQ** HIGH for a single **ATCLK** cycle.

Optionally, a Transmitter can implement a **SYNCREQ** input. This allows the Transmitter to recognize a synchronization request from a Receiver when this input is asserted HIGH for a single **ATCLK** cycle.

The **SYNCREQ** signal is not related in any way to any other signal on the ATB interface. However, from ATB-B (v1.1) onwards, the specification expects:

- **SYNCREQ** to be routed through a system with the other ATB signals.
- Any trace link, such as a trace funnel, to pass an input synchronization request from an ATB Transmitter interface to an upstream trace sources.

An enabled trace source must attempt to provide synchronization information in the trace stream once for each synchronization request received. The trace source can:

- Delay outputting the synchronization information to avoid overflowing any internal buffers.
- Ignore a synchronization request that is received before the trace source has output the synchronization information corresponding to a previous synchronization request.

An ATB Transmitter must be aware that a pulse on **SYNCREQ** might be received at any time, and must ensure it is responsive to **SYNCREQ** when required.

Chapter 5
Interface Signaling Rules

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

5.1 Interface signaling rules

ATB protocol signaling must conform to the following rules:

- ATB signals must be 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.

  Data must be aligned to the least significant byte.

  Note

  Zero bytes cannot be captured.
- The width of **ATDATA**, in bits, must be a power-of-two equal to or greater than 8.
- The relationship between the widths of **ATBYTES[m:0]** and **ATDATA[n:0]** is defined by the equation:

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

  For examples of possible implementations, see *Relationship between ATDATA and ATBYTES* on page 3-26.
- The **ATDATA** value must be captured at the same time as bytes on **ATID** are captured.
- The order of trace bytes must be preserved, even between trace with different **ATIDs**.

  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 4-31.

  Note

  This differs from the ordering required by AXI interfaces, where ordering is preserved only between transactions with the same ID.
- When **AFVALID** is asserted, it must remain asserted until **AFREADY** is asserted.

  AFREADY is ignored while **AFVALID** is deasserted.
- When **AFVALID** is asserted, a Transmitter must output any buffered trace immediately.
- When **AFREADY** is asserted, **AFVALID** must be deasserted on the following cycle, unless an additional flush is required.
- When **AFVALID** is asserted, the Transmitter must assert **AFREADY** one cycle after it has output the last of the trace that it generated and stored, up to and including any trace generated on the cycle in which **AFVALID** was asserted. For more information, see *Buffer flush* on page 4-31.
- **ATRESETn** can be asserted asynchronously. However, **ATRESETn** must be deasserted synchronously on a rising edge of **ATCLK**.

  Note

  ATRESETn is an active LOW signal, meaning it must be asserted LOW.
- A trace source can only start driving **ATVALID** HIGH after **ATRESETn** has been HIGH at a rising edge of **ATCLK**.
- Whenever a Receiver interface cannot respond, **ATREADY** must be driven HIGH and **AFVALID** must be driven LOW. Examples of when this must happen include powering down, the Receiver not being present, and the result of incorrect programming of the system.

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

- Whenever a Transmitter interface cannot respond, **AFREADY** must be driven HIGH and **ATVALID** must be driven LOW. Examples of when this must happen include powering down, the Transmitter not being present, and the result of incorrect programming of the system.
- **ATID**, **ATBYTES**, and **ATDATA** can have any value. This permitted behavior can be used to tie off unused CoreSight trace funnel input signals.
- The **ATID** values `0x00`, `0x70-0x7C`, `0x7E`, and `0x7F` are reserved for use by the CoreSight Architecture and must not be used as ATB IDs.

Chapter 6
Wake-up signaling

This chapter introduces wake-up signaling used in the ATB protocol. It contains the following section:

6.1 Introduction

The wake-up signal is used to indicate that there is activity associated with an ATB interface. This can be used to provide a glitch-free signal that can be routed to a clock controller, or similar component, to enable power and clocks to the connected components.

The Wakeup_Signal property is used to indicate whether a component supports wake-up signaling:

True Wake-up signal is present.

False Wake-up signal is not present.

If Wakeup_Signal is not declared, it is considered False.

Wake-up signaling can only be added to ATB Issue C interfaces.

6.2 ATWAKEUP signaling

Table 6-1 shows **ATWAKEUP** signal description.

Table 6-1 ATWAKEUP signal

Signal	Width	Source	Description
ATWAKEUP	1	Transmitter	Wake-up signal. Indicates that there is activity associated with an ATB interface.

The rules for **ATWAKEUP** are:

- **ATWAKEUP** is synchronous to **ATCLK** but is more appropriate for crossing clock domains to a controller.
- **ATWAKEUP** must be glitch-free and generated directly from a register.
- It is permitted for **ATWAKEUP** to be asserted at any point before or after the assertion of **ATVALID**.
- A Receiver is permitted to wait for **ATWAKEUP** to be asserted, before asserting **ATREADY**. This means that the ATB interface could deadlock if **ATWAKEUP** is present, but never asserted.
- If **ATWAKEUP** and **ATVALID** are HIGH in the same cycle, **ATWAKEUP** must remain HIGH until **ATREADY** is asserted.

It is recommended:

- **ATWAKEUP** is asserted at least one cycle before the assertion of **ATVALID** to prevent the acceptance of a new transaction being delayed.
- **ATWAKEUP** is deasserted when no further transfers are required.

  It is permitted, but not recommended, to assert **ATWAKEUP** then deassert it without a transaction taking place.

Appendix A
Signal matrix

This appendix describes a summary of the interface signals used in ATB. It contains the following section:

- *ATB signals* on page A-46

A.1 ATB signals

Table A-1 shows the interface signals. Optional signals have a default value that should be used for any undriven inputs.

The following code is used:

Y Mandatory
N Must not be present
O Optional for inputs and outputs
OO Optional for output ports, mandatory for inputs
C Conditional, must be present if property is True
OC Optional conditional, optional but can be present if property is True

Table A-1 Interface signals

Signal	Width	Default	Property	ATB-C	ATB-B (v1.1)	ATB-A (v1.0)
ATCLK	1	-	-	Y	Y	Y
ATCLKEN	1	1	-	O	O	Y
ATRESETn	1	-	-	Y	Y	Y
ATBYTES	$\log_2$(DATA_WIDTH) - 3	-	-	Y	Y	Y
ATDATA	DATA_WIDTH	-	-	Y	Y	Y
ATID	7	-	-	Y	Y	Y
ATREADY	1	-	-	Y	Y	Y
ATVALID	1	-	-	Y	Y	Y
ATWAKEUP	1	1	Wakeup_Signal	C	N	N
AFVALID	1	-	-	Y	Y	Y
AFREADY	1	-	-	Y	Y	Y
SYNCREQ	1	0	-	O	O	N

Appendix B
Revisions

This appendix describes the technical changes between released issues of this book.

Table B-1 Differences between issue A and issue B

Change	Location
Addition of **SYNCREQ** signal	• *Synchronization request signal* on page 2-21 • Figure 3-1 on page 3-25
Change to reserved ATID values	• *ATIDs* on page 4-30 • *Interface signaling rules* on page 5-38
Addition of trace synchronization signaling	*Signaling a trace trigger* on page 4-34
Addition of synchronization request signaling	*Synchronization request* on page 4-35
ATCLKEN becomes an optional signal	• *Global signals* on page 2-20 • Figure 3-1 on page 3-25
Removal of recommendation that **ATDATA** is at least 32 bits wide	• *Relationship between ATDATA and ATBYTES* on page 3-26 • *Interface signaling rules* on page 5-38

Table B-2 Differences between Issue B and Issue C

Change	Location
Terminology update	Throughout the specification
Addition of wake-up signaling	Chapter 6 *Wake-up signaling*

Glossary

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

Advanced eXtensible Interface (AXI)
An AMBA bus protocol that supports:

- Separate phases for address or control and data.
- Unaligned bus transfers using byte strobes.
- Burst-based transactions with only start address issued.
- Separate read and write channels.
- Issuing multiple outstanding addresses.
- Out-of-order transaction completion.
- Optional addition of register stages to meet timing or repropagation requirements.

The AXI protocol includes optional signaling extensions for low-power operation.

Advanced High-performance Bus (AHB)
An AMBA bus protocol supporting pipelined operation, with the address and data phases occurring during different clock periods. This means the address phase of a transfer can occur during the data phase of the previous transfer. The AHB provides a subset of the functionality of the AMBA AXI protocol

See also Advanced Microcontroller Bus Architecture.

Advanced Microcontroller Bus Architecture (AMBA)
The AMBA family of protocol specifications is the Arm open standard for on-chip buses. AMBA provides solutions for the interconnection and management of the functional blocks that make up a System-on-Chip (SoC). Applications include the development of embedded systems with one or more processors or signal processors and multiple peripherals.

See Advanced High-performance Bus.

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

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 (ETB)
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.