ARM® CoreSight™ STM-500 System Trace Macrocell

Revision: r0p0

Technical Reference Manual

ARM®

ARM DDI 0528A (ID100313)

ARM CoreSight STM-500 System Trace Macrocell
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
01 October 2013	A	Non-Confidential	First release for r0p0

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
ARM CoreSight STM-500 System Trace Macrocell Technical Reference Manual

Preface

About this book vi
Feedback x

Chapter 1 Introduction

1.1 About the STM-500 System Trace Macrocell 1-2
1.2 Compliance 1-4
1.3 Features 1-5
1.4 Interfaces 1-7
1.5 Configurable options 1-8
1.6 Test features 1-9
1.7 Product documentation, design flow, and architecture 1-10
1.8 Product revisions 1-11

Chapter 2 Functional Description

2.1 About the functions 2-2
2.2 Interfaces 2-3
2.3 Clocking and resets 2-5
2.4 Trace protocol 2-6
2.5 Timestamping 2-10
2.6 Triggering 2-11
2.7 Extended stimulus port interface 2-12
2.8 Hardware event tracing 2-18
2.9 DMA control 2-19
2.10 Data compression 2-21
2.11 Buffer flushing 2-22
2.12 ATB data ordering 2-24

2.13 Integration mode and topology detection 2-25
2.14 Constraints and limitations of use 2-26

Chapter 3 Programmers Model
3.1 About the programmers model 3-2
3.2 Register summary 3-3
3.3 Register descriptions 3-6

Appendix A Signal Descriptions
A.1 Clocks and resets A-2
A.2 AXI slave interface signals A-3
A.3 Debug APB interface signals A-5
A.4 ATB master interface signals A-6
A.5 Hardware event observation interface signals A-7
A.6 DMA peripheral request interface signals A-8
A.7 Timestamp port signals A-9
A.8 Authentication interface signals A-10
A.9 Non-secure guaranteed interface signals A-11
A.10 Cross-trigger interface signals A-12
A.11 Test interface signals A-13
A.12 AXI low-power interface signals A-14
A.13 STM low-power interface signals A-15

Appendix B Revisions

Preface

This preface introduces the *ARM® CoreSight™ STM-500 System Trace Macrocell Technical Reference Manual*. It contains the following sections:

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

About this book

This book is for the CoreSight *STM-500 System Trace Macrocell*.

Product revision status

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

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

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

Intended audience

This book is written for:

- Designers of development tools providing support for STM functionality. Implementation-specific behavior is described in this document. You can find complementary information in the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*.
- Hardware and software engineers integrating the STM into a *System on Chip* (SoC) design.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the STM-500.

Chapter 2 *Functional Description*

Read this for a description of the interfaces, operation, and clocking and resets.

Chapter 3 *Programmers Model*

Read this for a description of the programmers model and registers.

Appendix A *Signal Descriptions*

Read this for a description of the signals.

Appendix B *Revisions*

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

Glossary

The *ARM® Glossary* is a list of terms used in ARM documentation, together with definitions for those terms. The *ARM® Glossary* does not contain terms that are industry standard unless the ARM meaning differs from the generally accepted meaning.

See *ARM® Glossary*, `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`.

Conventions

Conventions that this book can use are described in:

- *Typographical conventions* on page vii.
- *Timing diagrams* on page vii.

- *Signals* on page viii.

Typographical conventions

The following table describes the typographical conventions:

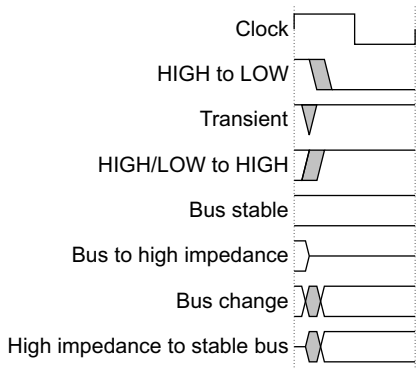
Typographical conventions

Style	Purpose
italic	Introduces special terminology, denotes 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>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *ARM® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The figure named *Key to timing diagram conventions* explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are UNDEFINED, 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.

Signals

The signal conventions are:

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

Lower-case n At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

ARM publications

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

- *ARM® System Trace Macrocell Programmers' Model Architecture Specification* (ARM IHI 0054).
- *ARM® ARMv7-M Architecture Reference Manual* (ARM DDI 0403).
- *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406).
- *ARM® Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014).
- *ARM® CoreSight™ Architecture Specification* (ARM IHI 0029).
- *ARM® Debug Interface Architecture Specification* (ARM IHI 0031).
- *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* (ARM IHI 0022).
- *ARM® AMBA® APB Protocol Specification* (ARM IHI 0024).
- *ARM® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* (ARM IHI 0032).
- *ARM® CoreLink™ DMA-330 DMA Controller Technical Reference Manual* (ARM DDI 0424).
- *ARM® CoreSight™ SoC-400 Technical Reference Manual* (ARM DDI 0480).

The following confidential books are only available to licensees:

- *ARM® CoreSight™ SoC-400 System Design Guide* (ARM DGI 0018).
- *ARM® CoreSight™ STM-500 System Trace Macrocell Integration and Implementation Manual* (ARM-EPM-043442).

Other publications

This section lists relevant documents published by third parties:
- *MIPI System Trace Protocol version 2 (STPv2)*.

- JEDEC Solid State Technology Association, *Standard Manufacturer's Identification Code*, JEP106.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- The title.
- The number, ARM DDI 0528A.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Note

ARM tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Introduction

This chapter introduces the CoreSight STM-500. It contains the following sections:

- *About the STM-500 System Trace Macrocell* on page 1-2.
- *Compliance* on page 1-4.
- *Features* on page 1-5.
- *Interfaces* on page 1-7.
- *Configurable options* on page 1-8.
- *Test features* on page 1-9.
- *Product documentation, design flow, and architecture* on page 1-10.
- *Product revisions* on page 1-11.

1.1 About the STM-500 System Trace Macrocell

Note

In this document, the word STM refers to the STM-500.

The STM-500 is a trace source that is integrated into a CoreSight system, and that is designed primarily for high-bandwidth trace of instrumentation embedded into software. This instrumentation is made up of memory-mapped writes to the STM *Advanced eXtensible Interface* (AXI) slave, which carry information about the behavior of the software.

The STM is a natural successor to the CoreSight *Instrumentation Trace Macrocell* (ITM) in mid- to high-performance applications. The STM provides the following advantages over the ITM for software instrumentation:

- It has a dedicated AXI slave interface for receiving the instrumentation information. The AXI slave interface is in addition to the *Advanced Peripheral Bus* (APB) interface that you can use for programming the STM registers. The AXI slave interface has significantly higher performance than the APB interface of the ITM.
- Multiple processors and processes can share and directly access the STM without being aware of each other, by being allocated different pages in the STM stimulus space. 128 masters, each supporting 65,536 stimulus ports. Each 4KB page of the STM stimulus space provides 16 stimulus ports. Stimulus ports are also known as channels.
- The STM can optionally stall the AXI when its FIFO becomes full, ensuring that no data is lost because of overflow, without having to poll the FIFO status in software. This behavior depends on the address written to, and can therefore be controlled by each stimulus port independently.
- An improved, configurable FIFO, supporting up to 32 transactions, reduces the likelihood of the FIFO becoming full.
- Timestamping can be requested for each write independently, based on the address written to. You can also optimize the bandwidth by requesting a timestamp for only one write transaction in a message made up of several writes.
- Timestamps are automatically correlated with other timestamping trace sources in the CoreSight system, enabling automatic correlation with, for example, processor execution trace.

In addition to the AXI slave, the STM provides a hardware event interface. The STM generates trace when signals are asserted on this interface. Alternatively, you can implement advanced custom system tracing features by generating AXI write accesses directly to the AXI slave.

Figure 1-1 on page 1-3 shows the STM integrated into a typical system.

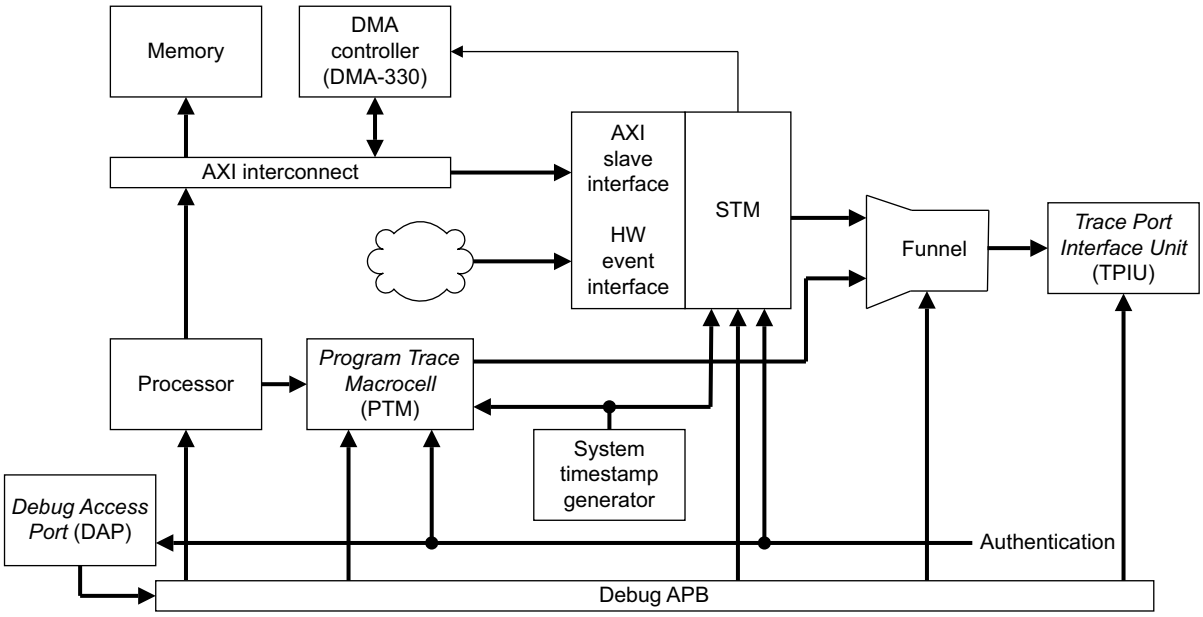


Figure 1-1 STM system integration

The STM AXI slave is connected to a system interconnect that enables all system masters, such as processors and *Direct Memory Access* (DMA) controllers, to generate trace by writing to the STM stimulus ports.

For interaction with DMA controllers, the STM provides a DMA request interface that is compatible with the AMBA DMA Controller DMA-330.

For configuration purposes, the STM is connected to a Debug APB interconnect. This enables off-chip and on-chip debug agents to access the STM.

The STM uses CoreSight authentication signals to control debug permissions.

The STM trace stream is output through the *Advanced Trace Bus* (ATB) interface and it is integrated with the rest of the CoreSight trace infrastructure.

1.2 Compliance

The STM complies with, or implements, the specifications described in:

- *System Trace Protocol*.
- *System Trace Macrocell Programmers' Model Architecture*.
- *CoreSight Architecture*.
- *Advanced Microcontroller Bus Architecture*.

This TRM complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

1.2.1 System Trace Protocol

The STM supports a trace stream that conforms to the MIPI System Trace Protocol version 2. See the *MIPI System Trace Protocol version 2 (STPv2)*.

1.2.2 System Trace Macrocell Programmers' Model Architecture

The STM implements the STM architecture version 1.1. See the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*.

1.2.3 CoreSight Architecture

The STM implements the CoreSight architecture version 2.0. See the *ARM® CoreSight™ Architecture Specification*.

1.2.4 Advanced Microcontroller Bus Architecture

The STM complies with the AMBA 4 protocol. See the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* and the *ARM® AMBA® APB Protocol Specification*.

1.3 Features

The STM has the following features:

- A fully synchronous design with one clock and two resets.
- One 64-bit AXI slave interface for extended stimulus port inputs.
- One hardware event observation interface for tracing 64 hardware events.
- One 32-bit debug APB slave interface for configuration and status.
- One 64-bit ATB master interface for trace output.
- One DMA peripheral request interface that is compatible with the AMBA DMA Controller DMA-330.
- Two depth-configurable FIFO buffers for usage-optimized configurability:
  — Data FIFO.
  — Channel information FIFO.
- A fully memory-mapped software stimulus supporting 65,536 stimulus ports and 128 masters.
- Leading zero data compression.
- Full support for guaranteed and invariant timing software stimulus writes.
- Support for single-shot and multi-shot triggers with a cross-trigger port, trigger packet insertion, and ATB trace triggers.
- An internal and an external source for STPv2 synchronization.
- Timestamping of trace events.
- Two low-power interfaces.

The STM architecture has many IMPLEMENTATION DEFINED options. Table 1-1 shows the configuration implemented by the STM-500.

Table 1-1 STM configuration

Feature	Configuration
Trace protocol	STPv2
Timestamping	Absolute
STMTSFREQR	Read-write
STMTSSTIMR	Implemented
STMSYNCR	Implemented
Claim tags	Four
TRACEID	CoreSight ATB and ATB trigger
Trigger control	Multi-shot and single-shot
STMTCSR.TSPRESCALE	Not implemented
STMTCSR.HWTEN	Not implemented
STMTCSR.SYNCEN	Always reads as `0b1`

Table 1-1 STM configuration (continued)

Feature	Configuration
STMTCSR.SWOEN	Not implemented
Number of stimulus ports	65536
Number of masters	Minimum of two
Stimulus port types	Extended only
Fundamental data size	64
Transaction Types	Invariant timing and guaranteed
STMSPER	Implemented
STMSPTER	Implemented
STMPRIVMASKR	Not implemented
STMSPOVERRIDER and STMSPMOVERRIDER	Implemented
STMSPSCR and STMSPMSCR	Implemented
Data compression on stimulus ports	Programmable
Hardware event tracing	Implemented
Number of hardware events	64
STMHETER	Implemented
Hardware error detection	Implemented
STMHEMASTR	Read only
Data compression on hardware event trace	Programmable
Hardware event multiplexing	The STMHEEXTMUXR is 8-bits wide

1.4 Interfaces

The STM-500 has the following external interfaces:

- AMBA AXI4 slave.
- Hardware event observation interface.
- DMA peripheral request interface.
- Debug APB slave interface.
- AMBA 4 ATB master interface.
- Timestamp port interface.
- Authentication interface.
- Non-secure guaranteed stimulus port access enable interface.
- Cross-trigger interface.
- Two low-power interfaces.

See *Interfaces* on page 2-3 for more information about these interfaces.

1.5 Configurable options

You can configure the depths of the STM FIFO buffers to match the instrumentation usage model. Deeper buffers improve STM performance but increase area and power consumption.

Each data FIFO entry can store up to 64 bits of data and up to 16 bits of timestamp information. Each channel FIFO entry can store one channel or master change message. It is not usually necessary to implement as many channel FIFO entries as data FIFO entries, because the channel or master does not usually change on every write to the AXI slave.

You can also specify the presence of the hardware event observation interface and the width of the AXI ID. Table 1-2 shows the configurable options.

Table 1-2 STM configurable options

Configurable option	Valid values	Description
DATA_FIFO_DEPTH	4, 8, 16, 32	Depth of the data FIFO buffer.
CHN_FIFO_DEPTH	4, 8, 16, 32[a]	Depth of the channel FIFO buffer.
HWEVOBIF_PRESENT	FALSE	Hardware event observation interface is not present.
	TRUE	Hardware event observation interface is present.
AXI_ID_WIDTH	2-24	Width of the AXI IDs, including **AWID**, **BID**, **ARID**, and **RID**.
HWEVOBIF_CONFIG_*n*	0 or 1	Specifies whether the hardware event input *n* is edge-sensitive or level-sensitive: **0** Edge sensitive. **1** Level sensitive. **Note** The RTL groups these configuration options into the following 32-bit parameters: HWEVOBIF_CONFIG_31_0 For hardware events 0-31. HWEVOBIF_CONFIG_63_32 For hardware events 32-63.

a. This value must not be more than the DATA_FIFO_DEPTH value.

If you are unsure about the instrumentation profile in your system, ARM recommends that you implement the STM in a configuration with CHN_FIFO_DEPTH set to eight and DATA_FIFO_DEPTH set to 16.

1.6 Test features

The STM includes clock gating circuitry to save power when the STM is disabled. It includes the following *Design-For-Test* (DFT) port to enable the clock during STM testing:

DFTCLKCGEN

Forces the STM clock on during DFT shift.

1.7 Product documentation, design flow, and architecture

This section describes the STM documents, how they relate to the design flow, and the relevant architectural standards and protocols.

See *Additional reading* on page viii for more information about the documents described in this section.

1.7.1 Documentation

The STM documentation is as follows:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the STM. It is required at all stages of the design flow. The choices made in the design flow can mean that some behavior described in the TRM is not relevant.

Integration and Implementation Manual

The *Integration and Implementation Manual* (IIM) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the RTL with the build configuration options.
- How to integrate the STM into an SoC. This includes describing the pins that the integrator must tie off to configure the macrocell for the required integration.
- The processes to sign off the integration and implementation of the design.

The ARM product deliverables include reference scripts and information about using them to implement your design.

Reference methodology documentation from your EDA tools vendor complements the IIM.

The IIM is a confidential document that is only available to licensees.

1.7.2 Design flow

The STM is delivered as synthesizable RTL. Before using the STM in a product, you must integrate and implement the STM. Implementation and integration choices affect the behavior and features of the STM. The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are configured. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell.

Configuration inputs

The integrator configures some features of the STM by tying inputs to specific values. These configurations affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration

The programmer configures the STM by programming particular values into registers. This affects the behavior of the STM.

1.8 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2
Functional Description

This chapter describes the interfaces, operation, and clocking and resets of the STM-500. It contains the following sections:

- *About the functions* on page 2-2.
- *Interfaces* on page 2-3.
- *Clocking and resets* on page 2-5.
- *Trace protocol* on page 2-6.
- *Timestamping* on page 2-10
- *Triggering* on page 2-11.
- *Extended stimulus port interface* on page 2-12.
- *Hardware event tracing* on page 2-18.
- *DMA control* on page 2-19.
- *Data compression* on page 2-21.
- *Buffer flushing* on page 2-22.
- *ATB data ordering* on page 2-24.
- *Integration mode and topology detection* on page 2-25.
- *Constraints and limitations of use* on page 2-26.

2.1 About the functions

Figure 2-1 shows the main functional blocks of the STM.

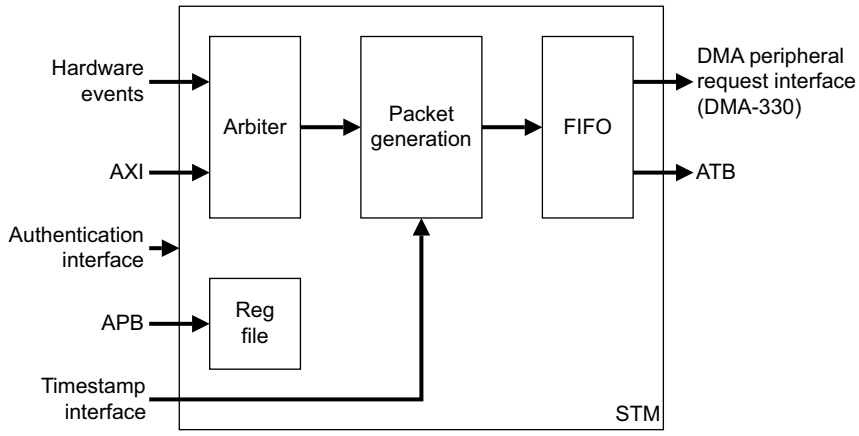


Figure 2-1 STM block diagram

The STM implements tracing of software writes to its stimulus ports using the AXI interface and tracing of hardware events in the following manner:

1. Selects stimulus ports and hardware events to be traced based on programmed controls and the state of the authentication interface.
2. Arbitrates, with higher priority assigned to AXI transactions, between traced transactions from the two interfaces, AXI and hardware events.
3. Presents the trace data to the packet generation logic, which timestamps and organizes the trace data according to the STPv2 protocol.
4. Packs the STPv2 data into the trace stream and outputs the stream on the ATB interface.

   In case of an overflow, that is, when the STM FIFO is full, the STM can either stall the AXI or drop the data indicating an overflow condition in the trace stream.

   There is no stall mechanism on the hardware interface. For hardware events, the STM either silently drops the event or indicates an overflow condition in the trace stream.
5. Periodically outputs a synchronization sequence to enable the trace receiver to align packet boundaries in the trace stream.

2.2 Interfaces

The STM interfaces are not configurable except for the `AXI_ID_WIDTH`. See the *ARM® CoreSight™ STM-500 System Trace Macrocell Integration and Implementation Manual* for more information about how to connect these interfaces in a system. The STM has the following external interfaces:

AXI slave interface

This interface connects the STM to the system bus. This design provides a 64-bit AMBA AXI4 slave. See the *ARM® AMBA® AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite* for more information about the AXI signals.

This interface occupies writable space in the memory map that you can use to generate trace.

Hardware event observation interface

This interface consists of 64 input signals, and connects to various signals from the system, such as interrupt lines, DMA request lines, and *Cross-Trigger Interface* (CTI) trigger outputs.

The STM captures hardware events on this interface and generates trace based on captured events. See *Hardware event tracing* on page 2-18.

DMA peripheral request interface

This interface connects to a Corelink DMA Controller DMA-330. When you program the STM to initiate a DMA transfer, this interface requests the DMA controller to write to the STM AXI interface. See *DMA control* on page 2-19.

Debug APB slave interface

This interface provides access to the STM configuration and status registers.

See the *ARM® AMBA® APB Protocol Specification* and the *ARM® CoreSight™ Architecture Specification* for more information about the debug APB signals.

ATB master interface

The STM uses the AMBA 4 ATBv1.1 interface for trace output. It includes handshaking signals for making flush requests to the STM and the ability for external logic to request insertion of synchronization information into the trace stream. The interface width is 64 bits.

See the *ARM® AMBA® 4 ATB Protocol Specification ATBv1.0 and ATBv1.1* for more information about the ATB signals.

Timestamp port interface

This interface provides the timestamp that is used in timestamped trace packets. The timestamp is a 64-bit natural binary encoded number.

Authentication interface

This interface provides connections for the CoreSight Authentication Interface.

The STM is a non-invasive debug component because it generates trace only in response to writes to its stimulus ports. See *Extended stimulus port interface* on page 2-12.

Non-secure guaranteed stimulus port accesses enable interface

This interface provides control over behavior of Non-secure guaranteed accesses to the extended stimulus ports. See *Extended stimulus port interface* on page 2-12.

Cross-trigger interface

The STM implements three trigger output ports, **TRIGOUTSPTE**, **TRIGOUTSW**, and **TRIGOUTHETE**, to connect to a cross-trigger interface in a CoreSight system to indicate trigger events. The output ports correspond to the dedicated trigger outputs described in the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*.

The **ASYNCOUT** output port indicates that alignment synchronization has occurred. You can use this port to generate other forms of periodic synchronization, for example by causing an interrupt on a processor. You must also connect this signal to a cross-trigger interface input.

Low-power interfaces

The STM includes two low-power interfaces that are used by a clock controller to determine whether the clock to the STM can be stopped:

- The AXI low-power interface to determine if the AXI slave and DMA interfaces are idle.
- The STM low-power interface to determine if the rest of the STM is idle.

A clock controller must only stop the clock if both low power interfaces indicate that the STM is idle.

2.3 Clocking and resets

The following sections describe the STM clock and resets:

- *Clock*.
- *Resets*.

2.3.1 Clock

The STM has a single clock input, **CLK**, which is synchronous to the system bus clock. You must use asynchronous bridges when connecting the STM interfaces to differently-clocked buses.

To minimize power consumption when not enabled, the STM implements architectural clock gating. The STM gates the internal clock for all parts of the design except for the AXI slave, DMA peripheral request, and low-power interfaces. These interfaces must be able to respond to transactions from the system interconnect, the DMA controller, and the clock controller for the low-power interfaces.

The architectural clock gating is transparent to the programmer. The internal clock is enabled when you access the STM registers or enable tracing.

2.3.2 Resets

The STM has two asynchronous active LOW resets:

ARESETn The **ARESETn** reset domain resets the AXI slave, the DMA peripheral request blocks, and the AXI low-power interface.

STMRESETn The **STMRESETn** reset domain resets the rest of the STM, including the hardware event observation interface, the APB interface register file, the STM low-power interface, and the *Trace Generation Unit* (TGU).

The STM supports independent assertion of both resets, and can:

- Maintain the STM state through an AXI reset, **ARESETn** asserted, and **STMRESETn** not asserted.
- Reset debug logic without resetting the rest of the system, **ARESETn** not asserted, and **STMRESETn** asserted.

2.4 Trace protocol

Trace generated by the STM conforms to the *MIPI System Trace Protocol version 2* (STPv2).

The STM provides 65,536 STPv2 channels for flexibility and for enabling additional definition of message boundaries. These channels directly map to the 65,536 extended stimulus ports.

The maximum size data packet generated is 64 bits.

The following sections describe trace protocol:

- *Trace packets*.
- *Alignment synchronization* on page 2-7.
- *Error packets* on page 2-8.
- *Trace output* on page 2-9.

2.4.1 Trace packets

Table 2-1 shows the valid trace packets that the STM generates.

Table 2-1 Generated trace packets

Name	Type	Description
ASYNC	Alignment	Provides alignment synchronization for the trace stream. This packet is always followed by a VERSION packet.
C8, C16	Channel	Channel indicator. The STM sends a C8 or C16 packet only when the software channel changes. C16 indicates a change in any of the upper 8 bits of the channel number. C8 indicates that the channel change is limited to any of the lower 8 bits. The default channel is 0, which is set when the STM generates a VERSION packet.
D4, D8, D16, D32, D64	Data	Data only
D4M, D8M, D16M, D32M, D64M	Data	Data with marker
D4TS, D8TS, D16TS, D32TS, D64TS	Data	Data with timestamp
D4MTS, D8MTS, D16MTS, D32MTS, D64MTS	Data	Data with marker and timestamp
M8	Master	Master indicator. The STM sends this packet only when the source master for the trace changes. The default master is 0, which is set when the STM generates a VERSION packet.
MERR	Error	Master error. Generated when the STM drops stimulus because of a full buffer.
GERR	Error	Global error. Generated when the STM drops stimulus from multiple masters because of a full buffer and the STM cannot send MERR for each master.
FLAG	Marker	Flag, can indicate message boundaries
FLAG_TS	Marker	Flag with timestamp
NULL	Filler	Filler packet. The STM can insert this to byte align the trace stream for the ATB output.
TRIG	Trigger	Trigger

Table 2-1 Generated trace packets (continued)

Name	Type	Description
TRIG_TS	Trigger	Trigger with timestamp
VERSION	Version	Indicates protocol version. Always sent immediately after ASYNC. Also resets the active master and channel values to 0. The STM generates the following VERSION value only: **3** STPv2 with natural binary STPv2 timestamps.
FREQ	Frequency	Indicates the frequency of the global system counter supplying timestamp information, in Hertz. The STM always sends this packet immediately after an ASYNC-VERSION sequence when timestamping is enabled.

Note

See *Constraints and limitations of use* on page 2-26 for information about STPv2 trace packets that the STM does not generate.

2.4.2 Alignment synchronization

To comply with STPv2, STM performs alignment synchronization of the trace stream by generating an ASYNC packet followed by a VERSION packet. If you enable timestamping, a FREQ packet follows the VERSION packet.

The STM generates alignment synchronization packets as the first packets after the STM is enabled, and in response to synchronization requests.

In addition, the **ASYNCOUT** output signal uses a one-cycle pulse to indicate every alignment synchronization carried out by the STM.

Synchronization request sources

The synchronization request sources can be internal or external:

Internal synchronization requests

The STM generates an internal synchronization request when one of the following events occur:

- The STM is enabled.
- The STMTSFREQR is programmed.
- The STMSYNCR is programmed.
- The synchronization period ends, defined in terms of number of bytes of trace generated. The synchronization period is defined by the STMSYNCR.

External synchronization requests

An external synchronization request comes from outside the STM through the **SYNCREQM** signal. This enables the system to indicate that the STM must perform alignment synchronization at the next opportunity.

Both internal and external synchronization requests are disabled when the STM is disabled, that is, the STMTCSR.EN bit is set to `0b0`.

Synchronization request combining

The STM combines internal and external synchronization requests to avoid excessive ATB bandwidth usage when requests occur near each other.

Synchronization priority escalation

Insertion of the alignment synchronization sequence usually carries secondary priority to trace generation requests.

Note

This feature addresses non-typical usage cases. In a typical trace scenario, you must not enable this feature, that is, leave the STMAUXCR at its reset value.

The STMAUXCR contains an override control to guarantee synchronization insertion when required. You can enable the override by programming the STMAUXCR.ASYNCPE bit. The behavior of override control is as follows:

- If the STM cannot insert the synchronization packets, the request remains pending.
- If the STM receives another synchronization request when there is a request pending and the STMAUXCR.ASYNCPE bit is set, the STM escalates the priority of the synchronization request.
- The STM stalls trace data if required when inserting a high priority synchronization request.

Note

The STMAUXCR is implementation defined. Controls defined in this register are not defined in the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*. There is no guarantee that other implementations of the architecture have the same controls in the STMAUXCR.

2.4.3 Error packets

The STM generates and inserts error packets in the trace stream when information is lost because of overflow.

MERR packets

MERR packets are master-specific. A MERR packet referring to a currently inactive master is preceded by an M8 packet. The STM resets the active channel to zero after the MERR packet is generated.

MERR packets have an 8-bit payload which is always `0x00`.

GERR packets

The STM generates and inserts GERR packets in the trace stream when more than one master experiences an error condition and a MERR packet cannot be output for each master.

GERR packets have an 8-bit payload that is always `0x00`.

2.4.4 Trace output

The STM always tries to perform 64-bit ATB transactions to maximize trace bandwidth efficiency. Therefore, it can hold trace packet nibbles until enough nibbles are generated to enable a 64-bit ATB transaction. The exceptions to this are when:

- An ATB flush is requested.
- The STM is disabled.
- Auto-flush is enabled.
- Authentication permissions are removed.

In these cases, the STM can perform smaller-size ATB transactions after byte aligning the data.

2.5 Timestamping

The STM supports timestamping of trace using values from a system global counter. The timestamp is a 64-bit natural binary encoded value.

2.6 Triggering

A trigger event indicates points of interest within the trace. The STM supports triggering functionality through:

- Assertion of the trigger outputs, **TRIGOUTSPTE**, **TRIGOUTSW**, and **TRIGOUTHETE**.
- Insertion of TRIG or TRIG_TS packets into the trace stream.
- Use of the trigger ATID on the ATB interface.

See the *ARM® System Trace Macrocell Programmers' Model Architecture Specification* for more information about triggers.

2.6.1 TRIGOUT ports

The **TRIGOUTSPTE**, **TRIGOUTSW**, and **TRIGOUTHETE** ports are part of a cross-trigger interface. The STM asserts these ports to indicate to the system that trigger events have occurred. The corresponding port is asserted in every cycle in which a particular trigger event has occurred. Table 2-2 shows the function of the TRIGOUT ports.

Table 2-2 TRIGOUT ports

TRIGOUT port	Trigger event
TRIGOUTSPTE	Match using the STMSPTER.
TRIGOUTSW	Write to the TRIG location.
TRIGOUTHETE	Match using the STMHETER.

2.7 Extended stimulus port interface

The STM implements an extended stimulus port interface using AXI. There are 65,536 channels implemented as a stimulus port per channel, occupying a total of 16MB in the AXI memory space. See the *ARM® System Trace Macrocell Programmers' Model Architecture Specification* for more information about the extended stimulus port interface.

The STM implements an AMBA AXI4 slave with the following attributes for extended stimulus port functionality:

- Write acceptance capability of three.
- Read acceptance capability of one.
- Read data reordering depth of one, that is, no reordering.

Figure 2-2 shows an example AXI write sequence.

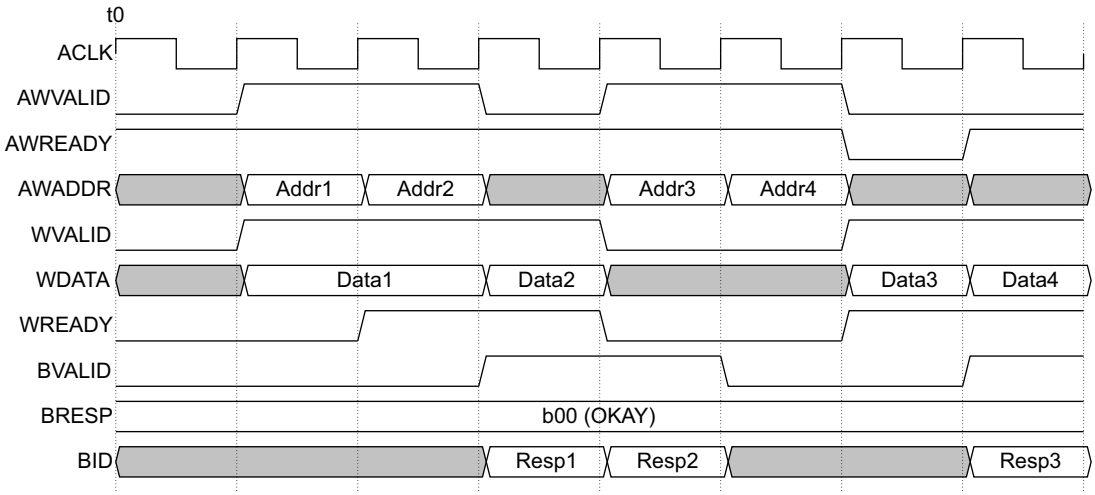


Figure 2-2 Example AXI write sequence

The following sections describe the extended stimulus port interface:

- *AXI responses*.
- *STM enabled*.
- *STM disabled* on page 2-17.
- *AXI reads* on page 2-17.
- *Stimulus port and trigger enables* on page 2-17.
- *Invariant-timing packets and overflow* on page 2-17.

2.7.1 AXI responses

The STM gives an OKAY response for all transfers. The trace output indicates error conditions during STM operation, such as dropped requests.

2.7.2 STM enabled

The following sections describe the operation when the STM is enabled:

- *Guaranteed and invariant timing transactions* on page 2-13.
- *Multiple STPv2 master support* on page 2-13.
- *Timestamp requests* on page 2-14.
- *WSTRBS usage* on page 2-14.
- *AWSIZE usage* on page 2-15.

- *Writes to disabled stimulus ports* on page 2-15.
- *Access control based on AWPROT* on page 2-15.
- *Non-secure guaranteed access control* on page 2-16.
- *Overrides* on page 2-16.

Guaranteed and invariant timing transactions

The STM supports both types of software stimulus transfers, that is, guaranteed and invariant timing. It considers transaction properties, such as master enable, stimulus port enable, override controls, and guaranteed or invariant timing, for each AXI burst beat.

Behavior on guaranteed transactions

If you make a guaranteed transaction when the STM is unable to accept the transaction immediately, the STM stalls the AXI write data channel until the write can be accepted, inserting as many wait states as necessary by keeping the **WREADY** signal LOW.

Use only a guaranteed write for critical trace information because it guarantees that a trace packet is generated for the write transaction. The wait states can adversely affect system performance.

Behavior on invariant timing transactions

If you make an invariant timing transaction when the STM is unable to trace the transaction immediately, the STM does not stall the AXI. After the address transfer, the STM drives the **WREADY** signal HIGH, enabling the data transfer. The STM discards write data if it cannot be traced.

There is no guarantee that an invariant timing write results in a trace packet appearing in the trace stream.

Multiple STPv2 master support

The STM supports multiple STPv2 masters. For AXI transactions, the master identification is based on all of the following:

- Whether the access is Secure or Non-secure.
- The upper bits of the AXI address.

Software stimulus has 128 master IDs allocated, `0x00-0x7F`. The lower half of the master ID space is for Secure accesses and the upper half for Non-secure accesses.

The STPv2 master ID from the AXI stimulus port is constructed as:

- [7] = `0b0`.
- [6] = **AWPROTS[1]**.
- [5:0] = **AWADDRS[29:24]**.

In this ID, master IDs 0-63 are allocated to Secure transactions and master IDs 64-127 to Non-secure transactions.

ARM strongly recommends that the **AWADDRS[29:24]** signal is driven uniquely for each master in the system. For example, transactions that arrive at the STM should drive the **AWADDRS[29:24]** signal with a unique value for each core in a processor cluster. Typically, this means that the system interconnect drives the **AWADDRS[29:24]** signal with a value derived from the original master of the transaction. However, it does not mean that the software instrumentation controls the **AWADDRS[29:24]** signal.

Instrumentation from each master is uniquely identified in the output trace stream.

Timestamp requests

Transactions to extended stimulus ports can include a timestamp request. You can promote transactions to extended stimulus ports to include a timestamp request by writing `0b1` to the STMTSSTIMR.FORCETS bit. If a transaction already includes a timestamp request, the FORCETS bit value has no effect.

Timestamping guaranteed writes

Timestamp requests from guaranteed writes are compulsory. This guarantees that a timestamp is included in the packet generated from the write. The AXI slave stalls all further writes if it cannot accommodate the timestamp request.

Timestamping invariant timing writes

The STM treats timestamp requests in an invariant timing transaction as sticky. A sticky timestamp request means that if the STM cannot accommodate a timestamp in the STM FIFO, the request remains pending and timestamp insertion occurs at the next opportunity. Sticky timestamp requests do not stall the STM.

The STM clears the pending sticky timestamp state when the timestamp is output regardless of whether a new transaction has its own timestamp request or not. This guarantees that a timestamp is eventually inserted into the trace stream because of the transaction. However, the timestamp might be attached to a different packet, and the number of timestamps generated might be fewer than the number requested.

WSTRBS usage

The STM uses the **WSTRBS** signal to determine the size of the transfer and the location of the data on the data bus, and ignores the value of the **AWADDRS[2:0]** signal. Table 2-3 shows the supported values of the **WSTRBS** signal, whether the encoding is permitted to be the first in a burst, which lanes of the **WDATAS** channel are used, and, by example, the packet that is produced when **AWADDR[6:3]** = `0x0`.

Table 2-3 Supported values of WSTRBS

WSTRBS	Permitted to start a burst	Data	Packet when AWADDR[6:3] = `0x0`
`0b00000000`	Yes	-	No data output.
`0b00000001`	Yes	**WDATAS[7:0]**	D8MTS
`0b00000010`	No	**WDATAS[15:8]**	D8MTS
`0b00000100`	No	**WDATAS[23:16]**	D8MTS
`0b00001000`	No	**WDATAS[31:24]**	D8MTS
`0b00010000`	No	**WDATAS[39:32]**	D8MTS
`0b00100000`	No	**WDATAS[47:40]**	D8MTS
`0b01000000`	No	**WDATAS[55:48]**	D8MTS
`0b10000000`	No	**WDATAS[63:56]**	D8MTS
`0b00000011`	Yes	**WDATAS[15:0]**	D16MTS
`0b00001100`	No	**WDATAS[31:16]**	D16MTS
`0b00110000`	No	**WDATAS[47:32]**	D16MTS

Table 2-3 Supported values of WSTRBS (continued)

WSTRBS	Permitted to start a burst	Data	Packet when AWADDR[6:3] = 0x0
0b11000000	No	**WDATAS[63:48]**	D16MTS
0b00001111	Yes	**WDATAS[31:0]**	D32MTS
0b11110000	No	**WDATAS[63:32]**	D32MTS
0b11111111	Yes	**WDATAS[63:0]**	D64MTS

All other values of the **WSTRBS** signal result in UNPREDICTABLE behavior. Unaligned transfers are not supported.

You must align all write data bursts so that the **WDATAS[0]** signal is the LSB at the start of the burst. Writes to other alignments are architecturally UNPREDICTABLE.

AWSIZE usage

Use the **AWSIZE** signal to control the address increment between beats of an incrementing burst.

Note

The **AWSIZE** signal is not used when calculating the packet type.

Writes to disabled stimulus ports

The STM responds to writes to disabled stimulus ports in the same way as for invariant timing transactions:

- The data is ignored and no trace is generated.
- Error packets are not generated when these writes are ignored.

Access control based on AWPROT

On writes, the STM compares the value of the **AWPROT[1]** signal with the current authentication status of the STM. If the value corresponds to a Secure transaction when Secure accesses are disabled, the STM drops the access as if the stimulus port were disabled.

See the *ARM® CoreSight™ Architecture Specification* for more information about Secure and Non-secure accesses to CoreSight components.

Note

The STM ignores the values of the **AWPROT[2]** and **AWPROT[0]** signals.

Table 2-4 shows the authentication control with Non-secure access for the **AWPROT[1]** signal.

Table 2-4 Authentication control with Non-secure access

AWPROT[1]	Permitted level of debug				Action on write
	Secure invasive	Non-secure invasive	Secure non-invasive	Non-secure non-invasive	
1, Non-secure	-	-	-	1	Capture
1	-	-	-	0	Drop
0, Secure	-	-	1	-	Capture
0	-	-	0	-	Drop

See the *ARM® CoreSight™ Architecture Specification* for information about mapping authentication signals, **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**, to the permitted level of debug.

Non-secure guaranteed access control

The top-level static configuration port, **NSGUAREN**, controls the behavior of the STM for Non-secure guaranteed AXI accesses:

- When the **NSGUAREN** signal is LOW, Non-secure guaranteed accesses behave like invariant timing accesses, that is, the AXI does not stall.
- When the **NSGUAREN** signal is HIGH, Non-secure guaranteed accesses are enabled, that is, the AXI can stall and the trace output is guaranteed.

Overrides

The debugger can override the stimulus ports selected by using the STMSPOVERRIDER and STMSPMOVERRIDER to always treat transactions as guaranteed or as invariant timing transactions.

When overriding transactions to be guaranteed, the STM treats the overrides as invasive debug because the debugger might increase the level of intrusion defined by system software. The guaranteed override is possible only when invasive debug is permitted. When invasive debug is disabled, the override has no effect.

Table 2-5 shows the authentication control with guaranteed override.

Table 2-5 Authentication control with guaranteed override

AWPROT[1]	Permitted level of debug				Guaranteed
	Secure invasive	Non-secure invasive	Secure non-invasive	Non-secure non-invasive	
1, Non-secure	-	1	-	-	Allowed
1	-	0	-	-	Not allowed
0, Secure	1	-	-	-	Allowed
0	0	-	-	-	Not allowed

See the *ARM® CoreSight™ Architecture Specification* for information about mapping the authentication signals, **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**, to the permitted level of debug.

There are no requirements for debug permissions when overriding transactions to make them invariant timing transactions.

2.7.3 STM disabled

When the STM is disabled, it accepts all writes immediately and ignores the write data. This behavior is identical to when all the STM stimulus ports are disabled.

2.7.4 AXI reads

All STM memory-mapped registers presented on the AXI are write-only. Reads always return an AXI OKAY read response and the read data is zero regardless of the STM state.

2.7.5 Stimulus port and trigger enables

You can enable or disable stimulus ports and triggers using the STMSPER, STMSPTER, and STMSPSCR. See the *ARM® System Trace Macrocell Programmers' Model Architecture Specification* for more information about stimulus ports and triggers.

2.7.6 Invariant-timing packets and overflow

If you perform a write access to an invariant-timing stimulus port location on the software stimulus interface, the STM immediately accepts the access, regardless of the state of the STM buffers. If there is not enough space for the generated packet in the STM buffer, the STM discards the data, and the data does not appear in the trace output.

If the data is discarded as a result of invariant timing writes made while the STM buffer is full, the STM adds a MERR packet to the STM buffer when space is available in the buffer. The STM generates only one MERR packet regardless of the number of discarded transactions, until:

- At least one of the other packet types is successfully generated and added to the buffer.
- The master ID is changed.

If the new data is discarded with a master ID different from the previously discarded transaction, and there is a MERR packet pending that has not been accepted by the STM FIFO, the STM generates a GERR packet instead.

2.8 Hardware event tracing

The STM hardware event observation interface enables monitoring and tracing of 64 hardware events, each of which is represented by a single bit. You can use this functionality to monitor interrupts, cross-triggers, and other signals in your system. The STM captures and traces, as a data packet, asserted and enabled hardware events.

Before you synthesize the STM-500, you can configure each hardware event to be rising-edge-sensitive or level-sensitive. When configured for level-sensitive observation, the STM treats each cycle on which you assert a hardware event input as a separate event.

The STM-500 can observe more than 64 hardware events, by using external multiplexing logic. You can control the multiplexing logic by using the STMHEEXTMUXR. The value of the STMHEEXTMUXR is output on the output bus HEEXTMUX from the STM.

To enable the STM to monitor and trace cross-trigger events in the system, ARM recommends that you:

1. Connect at least two trigger outputs from a CTI to hardware event inputs on the STM.
2. Connect the inverse of these CTI trigger outputs to the hardware event inputs on the STM to enable falling edges of these signals to be traced.
3. Configure the hardware events to be rising-edge-sensitive.
4. Enable the CTI connections when the **HEEXTMUX[7:0]** signal output is zero if you use external multiplexing logic and control the logic by the **HEEXTMUX[7:0]** signal.

You can enable triggers for hardware events, in which case the assertion of the event, when captured, also causes a trigger event.

The STM can merge multiple different hardware events into a single traced event.

If you assert the same hardware event multiple times while the first captured assertion is pending and has not yet been traced, the STM traces only one assertion and drops the others. If you enable error detection, the STM generates a MERR packet to indicate that merging has occurred.

See the *ARM® System Trace Macrocell Programmers' Model Architecture Specification* for more information about hardware events.

2.9 DMA control

The STM has a DMA peripheral request interface designed to function with a CoreLink DMA-330 DMA Controller. This interface enables the STM to request write accesses to its extended stimulus ports using the DMA controller. You must set up and program the DMA controller with the required transactions before enabling this functionality in the STM.

See the *ARM® CoreLink™ DMA-330 DMA Controller Technical Reference Manual* for more information about the specification of this interface.

2.9.1 Starting and stopping requests

The registers for controlling the DMA peripheral request interface are in the STM APB memory space:

- Program the STM to create DMA requests by writing `0b1` to the STMDMASTART.START bit. When programmed to create requests, the interface can create DMA controller requests whenever the STM FIFO has enough free space and a DMA transaction is not in progress. The interface creates requests repeatedly until programmed to stop.
- Program the STM to stop DMA requests by writing `0b1` to the STMDMASTOP.STOP bit. When programmed to stop, the STM does not create further DMA controller requests. Programming the interface to stop does not affect the progress of any existing active DMA transaction.

2.9.2 FIFO level monitoring

You can configure the DMA peripheral request interface to wait until a certain amount of free space is available in the STM FIFO before issuing the request.

FIFO level monitoring does not guarantee FIFO space for transactions from the DMA controller. However, this reduces the risk of a large number of invariant timing writes being dropped or guaranteed writes stalling the AXI interconnect for an extended period of time.

You can program the sensitivity of the DMA request to the current FIFO level using the STMDMACTLR.SENS bit. When FIFO conditions are met, the STM makes a DMA request. When the request is asserted, it remains asserted until accepted by the DMA controller, even if FIFO space is no longer available.

2.9.3 DMA interface behavior

The DMA interface functions as follows:

- When you start a DMA request by writing `0b1` to the STMDMASTART.START bit, the STM initiates a burst transfer by setting the **DRTYPE[1:0]** signal to `0b01`.
- When a burst is acknowledged by setting the **DATYPE[1:0]** signal to `0b01`, the STM initiates a new burst transfer as soon as there is sufficient space in the FIFO as described in *FIFO level monitoring*. The STM ignores single transfer acknowledges.
- When a flush is requested by setting the **DATYPE[1:0]** signal to `0b10`, the STM abandons a burst request and acknowledges the flush by setting the **DRTYPE[1:0]** signal to `0b10`. If DMA transfers are still enabled, the STM initiates a new burst immediately.
- The **DRLAST** signal is always `0b0`.

- The **DAVALID**, **DRVALID**, **DAREADY**, and **DRREADY** signals function with the preceding signals as described in the *ARM® CoreLink™ DMA-330 DMA Controller Technical Reference Manual*.

2.10 Data compression

The STM supports leading-zero trace data compression. You can enable support for this functionality for each of the extended stimulus port and hardware event observation front-end input interfaces:

- Program the STMTCSR.COMPEN bit to control the compression functionality for the extended stimulus ports.
- Program the STMHEMCR.COMPEN bit to control the compression functionality for the hardware event observation interface.

This compression method chooses a smaller data packet size than that specified by the stimulus if the data contains enough leading zeros. For example, if you write a 32-bit value of `0x0000FFFF` to the stimulus port and enable compression, the STM generates a D16 packet rather than a D32 packet. A value of `0x00000FFF` generates a D16 packet because the highest nonzero bit (11) is beyond the range of a D8 packet.

You must only enable compression for extended stimulus ports if the size of every write to the STM is known, for example because every write is 32 bits. This is because the original size of each STM write is not indicated in the packet when compression is enabled.

You can enable compression for the hardware event observation interface without losing information.

2.11 Buffer flushing

The STM might be required to flush its buffers, that is, output all buffered trace as soon as possible, in the following situations:

- A flush request is made over ATB using the **AFVALIDM** signal. The STM continues to accept new stimulus writes. The flush is complete when all trace generated before the flush request is output.
- The STM is disabled by reprogramming the STMTCSR.EN bit with `0b0`. The STM drops new stimulus until it is enabled again.
- Auto-flush is enabled in the STMAUXCR.
- Authentication is removed.

On a flush request, the STM continues outputting 64-bit ATB transactions as normal. However, if insufficient nibbles remain in the buffer for a 64-bit transaction, the STM does not wait. It chooses the most appropriate transaction size and pads the buffer contents with NULL nibbles for the ATB transaction.

Note

- These features in the STMAUXCR are present to address non-typical usage cases. Do not enable these features in a typical trace scenario, that is leave the STMAUXCR at its reset value.
- The STMAUXCR is IMPLEMENTATION DEFINED. Controls defined in this register are not defined in the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*. There is no guarantee that other implementations of the architecture have the same controls as the STMAUXCR.

The following sections describe buffer flushing:

- *Override using auto-flush*.
- *ATB flush request and priority inversion*.

2.11.1 Override using auto-flush

You can enable the STM to output ATB transactions that are smaller than 64 bits, without waiting for sufficient data to complete a 64-bit transaction. To override this behavior, program the STMAUXCR.FIFOAF bit with `0b1`, which enables auto-flush of the STM FIFOs.

When auto-flush is enabled, if the FIFO contains insufficient data for a 64-bit ATB transfer and has all its other entries empty, the STM performs a smaller-sized ATB transaction rather than waiting for more data, padding the data with as many NULL packets as required for byte alignment. For example:

- If there are five nibbles to output (20 bits), the STM inserts a NULL packet to create a 24-bit transaction.
- If there are three nibbles to output (12 bits), the STM inserts a NULL packet to create a 16-bit transaction.
- If there are two nibbles to output, no NULL packets are required.

2.11.2 ATB flush request and priority inversion

The STM acknowledges ATB flush requests by asserting the **AFREADYM** output signal, after all data present in the STM before the ATB flush request has been output.

The default behavior on an ATB flush request is to flush AXI stimulus historical data first, and then flush historical data from hardware event tracing. When historical AXI data has been flushed, the STM temporarily inverts the priority and assigns a higher priority to hardware event tracing until it is flushed. This helps the flush to complete as soon as possible, but can cause loss of invariant transactions on the AXI.

You can use the override control in the STMAUXCR to disable priority inversion during flush. If you set the STMAUXCR.PRIORINVDIS bit, the AXI stimulus port trace remains higher priority than hardware events trace during flush. This can delay the STM flush acknowledgement because the STM can output new AXI trace before historical hardware event data.

2.12 ATB data ordering

When ATB transfers are generated, the STM modifies the order of the data from the original packet values. The STM splits the channel, data, and timestamp values into nibbles and writes them in reverse nibble order, most significant nibble first in the data stream, after the packet header (opcode) or timestamp size.

For example, a write requesting a D32TS at stimulus port 20 of value `0x12345678` with timestamp offset of value `0x1234` produces the following data values:

- Channel packet C8 = `0x3`, `0x14` (opcode, channel number).
- Data packet D32TS = `0xF6`, `0x12345678` (opcode, data value).
- Associated timestamp TS = `0x4`, `0x1234` (size = 4 nibbles, timestamp value).

The generated ATB transactions are:

```
0x21487654_3216F413
0xXXXXXXXX_XXXXXX43
```

Xs are filled by the next packet.

Table 2-6 shows the structure of the ATB transactions.

Table 2-6 ATB transactions

ATB 32-bit transaction	ATB data nibbles	Source data nibbles
First doubleword `0x21487654_3216F413`	[63:60] = `0x2`	TS value[11:8]
	[59:56] = `0x1`	TS value[15:12]
	[55:52] = `0x4`	TS size[3:0]
	[51:48] = `0x8`	D32TS data[3:0]
	[47:44] = `0x7`	D32TS data[7:4]
	[43:40] = `0x6`	D32TS data[11:8]
	[39:36] = `0x5`	D32TS data[15:12]
	[35:32] = `0x4`	D32TS data[19:16]
	[31:28] = `0x3`	D32TS data[23:20]
	[27:24] = `0x2`	D32TS data[27:24]
	[23:20] = `0x1`	D32TS data[31:28]
	[19:16] = `0x6`	D32TS opcode[3:0]
	[15:12] = `0xF`	D32TS opcode[7:4]
	[11:8] = `0x4`	C8 data[3:0]
	[7:4] = `0x1`	C8 data[7:4]
	[3:0] = `0x3`	C8 opcode[3:0]
Second double word `0xXXXXXXXX_XXXXXX43`	[63:8] = `0xXXXXXXXX_XXXXXX`	Filled by the next packet or set to NULL.
	[7:4] = `0x4`	TS value [3:0]
	[3:0] = `0x3`	TS value [7:4]

2.13 Integration mode and topology detection

The STM implements integration mode to enable integration testing and CoreSight topology detection. Enable this mode by setting the STMITCTRL.IME bit to `0b1`. The STM interfaces available for topology detection are:

- One ATB master interface.
- Cross-trigger interfaces:
  - **TRIGOUTSPTE**.
  - **TRIGOUTSW**.
  - **TRIGOUTHETE**.
  - **ASYNCOUT**.

The STM must be disabled for integration mode to function correctly. If you use integration mode, you must reset the system before it is used in functional mode.

2.14 Constraints and limitations of use

Table 2-7 shows the trace packets that the STM does not generate and the reason they are not generated.

Table 2-7 Non-generated trace packets

Name	Type	Description and reason for non-generation
M16	Master	Identifies the currently active master where the new master number differs from the old master number in bits [15:8]. The STM design has only 129 masters, 128 for software stimulus, and one master for hardware stimulus. Therefore, the M8 packet is sufficient and the M16 packet is never required.
TIME(_TS)	Time	The STM only uses global timestamps, so this packet provides no useful information.
NULL_TS	Filler	NULL packet with timestamp. NULL cannot be generated in response to stimulus so it is impossible to force a timestamp for NULL in the STM.
USER(_TS)	User	Conveys implementation-specific metadata to the trace receiver. The STM has no such information that is not already accessible through APB reads. Therefore, it does not generate this packet type.
FREQ_TS	Timestamp frequency	Timestamp clock frequency, timestamped. FREQ cannot be generated in response to stimulus so it is impossible to force a timestamp for FREQ in the STM.
XSYNC(_TS)	Cross-synchronization	The STM does not support cross-synchronization using XSYNC.

Chapter 3
Programmers Model

This chapter describes the programmers model. It contains the following sections:

- *About the programmers model* on page 3-2.
- *Register summary* on page 3-3.
- *Register descriptions* on page 3-6.

3.1 About the programmers model

The following information applies to the registers:

- The base address is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in UNPREDICTABLE behavior.
- Unless otherwise stated in the accompanying text:
  - Do not modify Reserved register bits.
  - Ignore Reserved register bits on reads.
  - All register bits are reset to a logic zero when the **STMRESETn** signal is asserted.
- Access type in Table 3-1 on page 3-3 is described as follows:

  RW Read and write.

  RO Read only.

  WO Write only.

3.2 Register summary

Table 3-1 shows the registers in the base offset order.

Note

Table 3-1 lists the STM-500 registers. Some of these are defined in the TRM and some are defined in the *ARM® System Trace Macrocell Programmers' Model Architecture Specification*.

Table 3-1 STM register summary

Offset	Name	Type	Reset	Width	Description
0xC04	STMDMASTARTR	WO	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xC08	STMDMASTOPR	WO	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xC0C	STMDMASTATR	RO	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xC10	STMDMACTLR	RW	0x00000000	32	*DMA Control Register* on page 3-6
0xCFC	STMDMAIDR	RO	0x00000002	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xD00	STMHEER	RW	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xD20	STMHETER	RW	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xD60	STMHEBSR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xD64	STMHEMCR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xD68	STMHEEXTMUXR	RW	0x00000000	32	*Hardware Event External Multiplex Control Register* on page 3-6
0xDF4	STMHEMASTR	RO	0x00000080	32	*Hardware Event Master Number Register* on page 3-7
0xDF8	STMHEFEAT1R	RO	0x30200035	32	*Hardware Event Features 1 Register* on page 3-7
0xDFC	STMHEIDR	RO	0x00000011	32	*Hardware Event ID Register* on page 3-8
0xE00	STMSPER	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE20	STMSPTER	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE60	STMSPSCR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE64	STMSPMSCR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE68	STMSPOVERRIDER	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*

Table 3-1 STM register summary (continued)

Offset	Name	Type	Reset	Width	Description
0xE6C	STMSPMOVERRIDER	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE70	STMSPTRIGCSR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE80	STMTCSR	RW	a	32	*Trace Control and Status Register* on page 3-9
0xE84	STMTSSTIMR	WO	-	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE8C	STMTSFREQR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE90	STMSYNCR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xE94	STMAUXCR	RW	0x00000000	32	*Auxiliary Control Register* on page 3-10
0xEA0	STMFEAT1R	RO	0x006587D1	32	*STM Features 1 Register* on page 3-11
0xEA4	STMFEAT2R	RO	0x000114F2	32	*STM Features 2 Register* on page 3-13
0xEA8	STMFEAT3R	RO	0x0000007F	32	*STM Features 3 Register* on page 3-14
0xEE8	STMITTRIGGER	WO	-	32	*Integration Test for Cross-Trigger Outputs Register* on page 3-14
0xEEC	STMITATBDATA0	WO	-	32	*Integration Mode ATB Data 0 register* on page 3-15
0xEF0	STMITATBCTR2	RO	-	32	*Integration Mode ATB Control 2 register* on page 3-17
0xEF4	STMITATBID	WO	-	32	*Integration Mode ATB Identification register* on page 3-17
0xEF8	STMITATBCTR0	WO	-	32	*Integration Mode ATB Control 0 register* on page 3-18
0xF00	STMITCTRL	RW	0x00000000	32	*Integration Mode Control register* on page 3-19
0xFA0	STMCLAIMSET	RW	0x0000000F	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xFA4	STMCLAIMCLR	RW	0x00000000	32	*ARM® System Trace Macrocell Programmers' Model Architecture Specification*
0xFB0	STMLAR	WO	-	32	*Lock Access Register* on page 3-20
0xFB4	STMLSR	RO	b	32	*Lock Status Register* on page 3-20
0xFB8	STMAUTHSTATUS	RO	0x000000AA	32	*Authentication Status register* on page 3-21
0xFBC	STMDEVARCH	RO	0x47710A63	32	*Device Architecture register* on page 3-22
0xFC8	STMDEVID	RO	0x00010000	32	*Device Configuration register* on page 3-23
0xFCC	STMDEVTYPE	RO	0x00000063	32	*Device Type Identifier register* on page 3-23
0xFE0	STMPIDR0	RO	0x00000063	32	*Peripheral ID0 Register* on page 3-24
0xFE4	STMPIDR1	RO	0x000000B9	32	*Peripheral ID1 Register* on page 3-24
0xFE8	STMPIDR2	RO	0x0000000B	32	*Peripheral ID2 Register* on page 3-25
0xFEC	STMPIDR3	RO	0x00000000	32	*Peripheral ID3 Register* on page 3-26

Table 3-1 STM register summary (continued)

Offset	Name	Type	Reset	Width	Description
0xFD0	STMPIDR4	RO	0x00000004	32	*Peripheral ID4 Register* on page 3-27
0xFF0	STMCIDR0	RO	0x0000000D	32	*Component ID0 Register* on page 3-27
0xFF4	STMCIDR1	RO	0x00000090	32	*Component ID1 Register* on page 3-28
0xFF8	STMCIDR2	RO	0x00000005	32	*Component ID2 Register* on page 3-28
0xFFC	STMCIDR3	RO	0x000000B1	32	*Component ID3 Register* on page 3-29

a. See *Trace Control and Status Register* on page 3-9.

b. This value depends on the value of the **PADDRDBG31** signal.

3.3 Register descriptions

This section describes the STM registers. Table 3-1 on page 3-3 provides cross-references to individual registers.

3.3.1 DMA Control Register

The STMDMACTLR characteristics are:

Purpose Controls the DMA transfer request mechanism.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-1 shows the bit assignments.

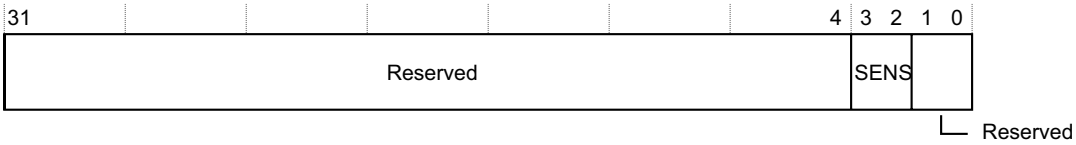


Figure 3-1 STMDMACTLR bit assignments

Table 3-2 shows the bit assignments.

Table 3-2 STMDMACTLR bit assignments

Bits	Name	Function
[31:4]	Reserved	Reserved.
[3:2]	SENS	Determines the sensitivity of the DMA request to the current buffer level in the STM: 0b00 Buffer is <25% full. 0b01 Buffer is <50% full. 0b10 Buffer is <75% full. 0b11 Buffer is <100% full.
[1:0]	Reserved	Reserved.

3.3.2 Hardware Event External Multiplex Control Register

The STMHEEXTMUXR characteristics are:

Purpose Controls multiplexing of available hardware inputs of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available only if you implement the hardware event observation interface.

Figure 3-2 shows the bit assignments.

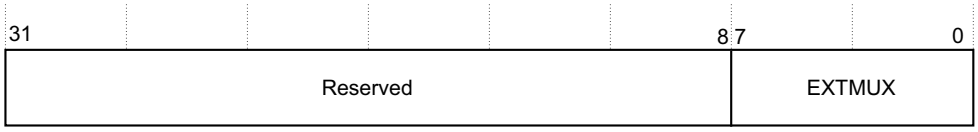


Figure 3-2 STMHEEXTMUXR bit assignments

Table 3-3 shows the bit assignments.

Table 3-3 STMHEEXTMUXR bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	EXTMUX	Specifies the value that the optional external multiplexing logic uses to select the hardware events to connect to the STM. The value of this register is output from the STM on the **HEEXTMUX[7:0]** signals. The behavior of the multiplexing logic is IMPLEMENTATION DEFINED. This field is reset to zero.

3.3.3 Hardware Event Master Number Register

The STMHEMASTR characteristics are:

Purpose Indicates the STPv2 master number of the hardware event trace. This number is the master number presented in STPv2.

Usage constraints There are no usage constraints.

Configurations This register is available if you implement the hardware event observation interface.

Figure 3-3 shows the bit assignments.

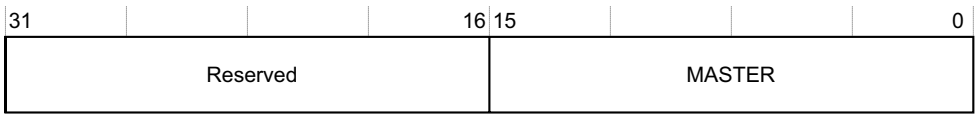


Figure 3-3 STMHEMASTR bit assignments

Table 3-4 shows the bit assignments.

Table 3-4 STMHEMASTR bit assignments

Bits	Name	Function
[31:16]	Reserved	Reserved.
[15:0]	MASTER	The STPv2 master number for the hardware event trace: `0x80` Hardware events are associated with master ID `0x80`.

3.3.4 Hardware Event Features 1 Register

The STMHEFEAT1R characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available if you implement the hardware event observation interface.

Figure 3-4 on page 3-8 shows the bit assignments.

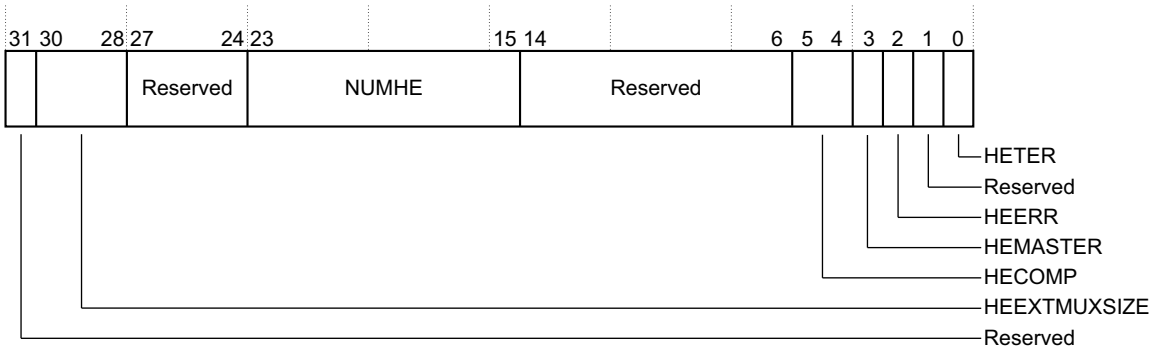


Figure 3-4 STMHEFEAT1R bit assignments

Table 3-5 shows the bit assignments.

Table 3-5 STMHEFEAT1R bit assignments

Bits	Name	Function
[31]	Reserved	Reserved.
[30:28]	HEEXTMUXSIZE	The size of the STMHEEXTMUXR.EXTMUX bit field: `0b011` 8 bits wide.
[27:24]	Reserved	Reserved.
[23:15]	NUMHE	The number of hardware events supported by the STM: `0b001000000` 64 hardware events.
[14:6]	Reserved	Reserved.
[5:4]	HECOMP	Data compression on hardware event tracing support: `0b11` Programmable data compression support. The STM implements the STMHEMCR.COMPEN bit.
[3]	HEMASTR	Specifies the STMHEMASTR support: `0b0` The STMHEMASTR is read-only.
[2]	HEERR	Hardware event error detection support: `0b1` Implemented. The STM implements the STMHEMCR.ERRDETECT bit.
[1]	Reserved	Reserved.
[0]	HETER	Specifies the STMHETER support: `0b1` Implemented.

3.3.5 Hardware Event ID Register

The STMHEIDR characteristics are:

Purpose Indicates the presence of hardware event tracing in the STM.

Usage constraints There are no usage constraints.

Configurations This register is available if you implement the hardware event observation interface.

Figure 3-5 on page 3-9 shows the bit assignments.

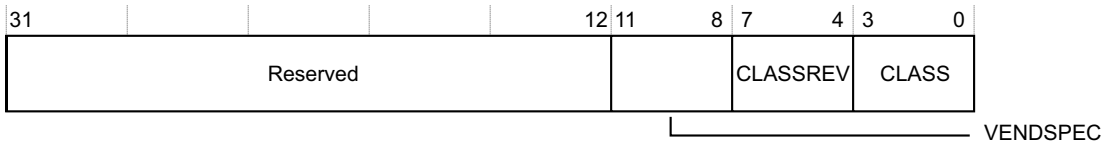


Figure 3-5 STMHEIDR bit assignments

Table 3-6 shows the bit assignments.

Table 3-6 STMHEIDR bit assignments

Bits	Name	Function
[31:12]	Reserved	Reserved.
[11:8]	VENDSPEC	Identifies vendor-specific modifications or mappings: `0b0000` Vendor-specific information.
[7:4]	CLASSREV	Identifies the revision of the programmers model: `0b0001` Revision.
[3:0]	CLASS	Identifies the programmers model: `0b0001` Hardware event control.

3.3.6 Trace Control and Status Register

The STMTCSR characteristics are:

Purpose Controls STM settings.

Usage constraints There are no usage constraints. The reset value bits[22:16] is UNKNOWN.

Configurations This register is available in all configurations.

Figure 3-6 shows the bit assignments.

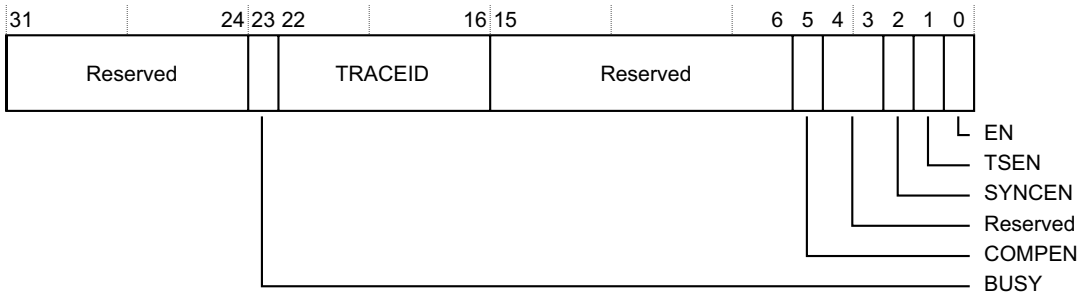


Figure 3-6 STMTCSR bit assignments

Table 3-7 shows the bit assignments.

Table 3-7 STMTCSR bit assignments

Bits	Name	Function
[31:24]	Reserved	Reserved.
[23]	BUSY	Indicates whether the STM is busy, for example the STM trace FIFO is not empty: 0b0 Not busy. 0b1 Busy.
[22:16]	TRACEID	ATB trace ID. Setting this value to all zeroes or to a value greater than 0x6F might result in UNPREDICTABLE tracing. Reset value of this bit field is UNKNOWN.
[15:6]	Reserved	Reserved.
[5]	COMPEN	Compression enable for stimulus ports: 0b0 Disabled, the STM transmits data transfers based on the size of the transaction. 0b1 Enabled, the STM compresses data transfers to save bandwidth.
[4:3]	Reserved	Reserved.
[2]	SYNCEN	The STM implements the STMSYNCR, so this bit is Read As One. 0b1 The STM implements the STMSYNCR.
[1]	TSEN	Determines whether to ignore timestamp requests: 0b0 Disable timestamping. The STM ignores requests for timestamp generation, and treats stimulus port writes that select timestamping as if timestamping were not selected. 0b1 Enable timestamping. If stimulus writes select timestamping, the STM outputs a timestamp according to STPv2.
[0]	EN	Global STM enable: 0b0 Disabled 0b1 Enabled.

3.3.7 Auxiliary Control Register

The STMAUXCR characteristics are:

Purpose Used for IMPLEMENTATION DEFINED STM controls.

Usage constraints There are no usage constraints. ARM recommends that you leave this register at its default reset value.

Configurations This register is available in all configurations.

Figure 3-7 shows the bit assignments.

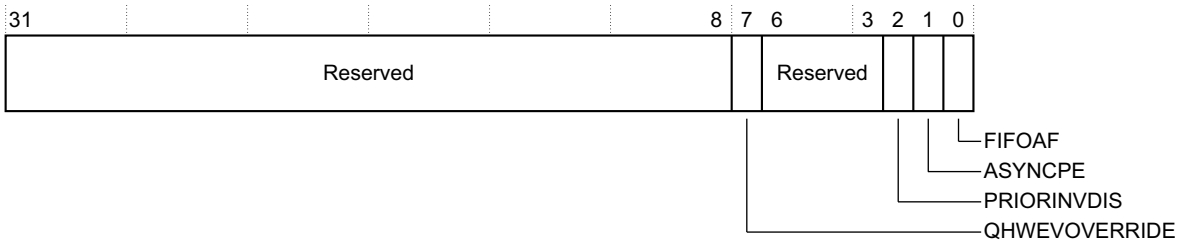


Figure 3-7 STMAUXCR bit assignments

Table 3-8 shows the bit assignments.

Table 3-8 STMAUXCR bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7]	QHWEVOVERRIDE	Low-power interface override when hardware event tracing is enabled: `0b0` The STM can accept a quiescence request on the STM low-power interface when the STMHEMCR.EN bit is set to `0b1`. `0b1` If the STMHEMCR.EN bit is set to `0b1`, all quiescence requests on the STM low-power interface are denied. Reset value is `0b0`.
[6:3]	Reserved	Reserved, software must preserve these bits on writes.
[2]	PRIORINVDIS	Controls arbitration between the AXI interface and the hardware event observation interface during flush: `0b0` Priority inversion. When the AXI flush completes, the hardware event observation interface gets priority until the hardware event observation interface flush completes. `0b1` Priority inversion disabled. The AXI always has priority over the hardware event observation interface. Reset value is `0b0`.
[1]	ASYNCPE	ASYNC priority: `0b0` Always lower than trace. `0b1` Escalates on second synchronization request. Reset value is `0b0`.
[0]	FIFOAF	Auto-flush: `0b0` Disabled. `0b1` Enabled. The STM automatically drains all data even if the ATB interface is not fully utilized. Reset value is `0b0`.

3.3.8 STM Features 1 Register

The STMFEAT1R characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-8 on page 3-12 shows the bit assignments.

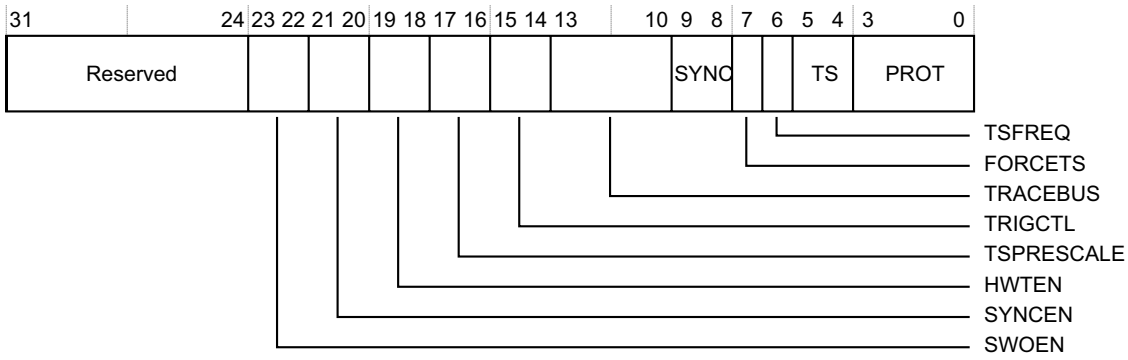


Figure 3-8 STMFEAT1R bit assignments

Table 3-9 shows the bit assignments.

Table 3-9 STMFEAT1R bit assignments

Bits	Name	Function
[31:24]	Reserved	Reserved.
[23:22]	SWOEN	Specifies the STMTCSR.SWOEN bit support: `0b01` Not implemented.
[21:20]	SYNCEN	Specifies the STMTCSR.SYNCEN bit support: `0b10` Implemented but always reads as `0b1`.
[19:18]	HWTEN	Specifies the STMTCSR.HWTEN bit support: `0b01` Not implemented.
[17:16]	TSPRESCALE	Timestamp prescale support: `0b01` Not implemented.
[15:14]	TRIGCTL	Trigger control support: `0b10` Multi-shot and single-shot triggers supported, the STM implements the STMTRIGCSR.
[13:10]	TRACEBUS	Trace bus support: `0b0001` CoreSight ATB plus ATB trigger support implemented, the STM implements the STMTCSR.TRACEID and STMTRIGCSR.ATBTRIGEN bit fields.
[9:8]	SYNC	Specifies the STMSYNCR support: `0b11` Implemented with MODE control.
[7]	FORCETS	Specifies the STMTSSTIMR support: `0b1` The STM implements the STMTSSTIMR.FORCETS bit.
[6]	TSFREQ	Timestamp frequency indication configuration: `0b1` The STMTSFREQR is read-write.
[5:4]	TS	Timestamp support: `0b01` Absolute timestamps implemented.
[3:0]	PROT	Protocol: `0b0001` STPv2 protocol.

3.3.9 STM Features 2 Register

The STMFEAT2R characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-9 shows the bit assignments.

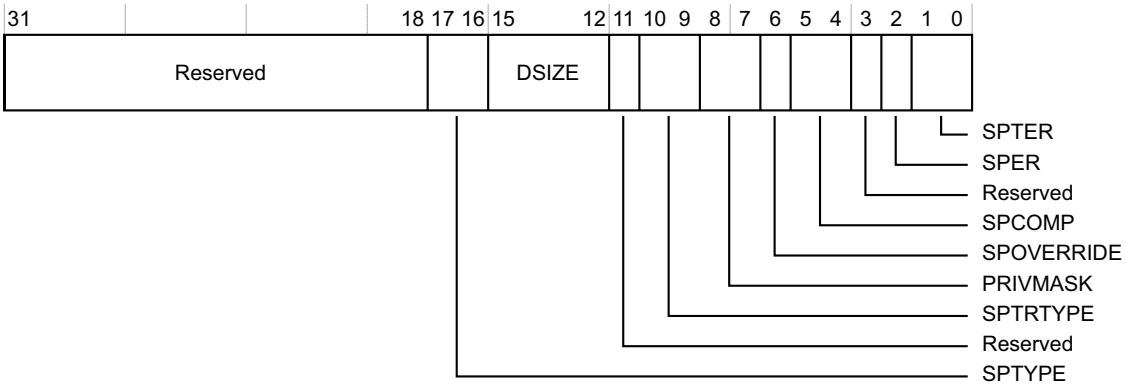


Figure 3-9 STMFEAT2R bit assignments

Table 3-10 shows the bit assignments.

Table 3-10 STMFEAT2R bit assignments

Bits	Name	Function
[31:18]	Reserved	Reserved.
[17:16]	SPTYPE	Stimulus port type support: `0b01` Only extended stimulus ports.
[15:12]	DSIZE	Fundamental data size: `0b0001` 64-bit data.
[11]	Reserved	Reserved.
[10:9]	SPTRTYPE	Stimulus port transaction type support: `0b10` Both invariant timing and guaranteed transactions.
[8:7]	PRIVMASK	Specifies the STMPRIVMASKR support: `0b01` Not implemented.
[6]	SPOVERRIDE	Specifies the STMSPOVERRIDER support: `0b1` STMSPOVERRIDER and STMSPMOVERRIDER implemented.
[5:4]	SPCOMP	Data compression on stimulus ports support: `0b11` Data compression support is programmable, the STM implements the STMTCSR.COMPEN bit.

Table 3-10 STMFEAT2R bit assignments (continued)

Bits	Name	Function
[3]	Reserved	Reserved.
[2]	SPER	Specifies the STMSPER presence: 0b0 Implemented.
[1:0]	SPTER	Specifies the STMSPTER support: 0b10 Implemented.

3.3.10 STM Features 3 Register

The STMFEAT3R characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-10 shows the bit assignments.

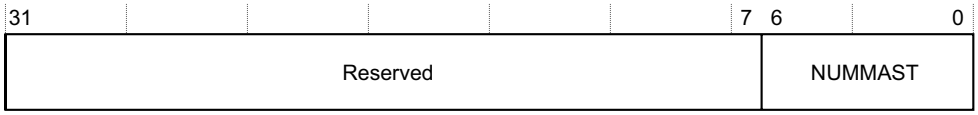


Figure 3-10 STMFEAT3R bit assignments

Table 3-11 shows the STMFEAT3R bit assignments.

Table 3-11 STMFEAT3R bit assignments

Bits	Name	Function
[31:7]	Reserved	Reserved.
[6:0]	NUMMAST	The number of stimulus port masters implemented minus 1: 0b1111111 128 ports.

3.3.11 Integration Test for Cross-Trigger Outputs Register

The STMITTRIGGER characteristics are:

Purpose Controls the values of the cross-trigger outputs in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-11 on page 3-15 shows the bit assignments.

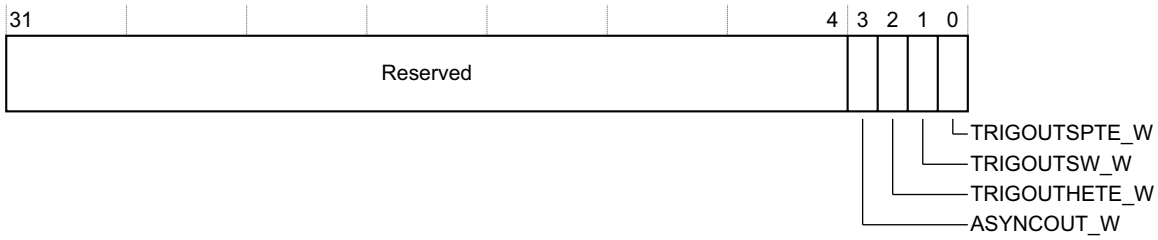


Figure 3-11 STMITTRIGGER bit assignments

Table 3-12 shows the bit assignments.

Table 3-12 STMITTRIGGER bit assignments

Bits	Name	Function
[31:4]	Reserved	Reserved.
[3]	ASYNCOUT_W	Sets the value of the **ASYNCOUT** output signal in integration mode: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[2]	TRIGOUTHETE_W	Sets the value of the **TRIGOUTHETE** output signal in integration mode: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[1]	TRIGOUTSW_W	Sets the value of the **TRIGOUTSW** output signal in integration mode: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[0]	TRIGOUTSPTE_W	Sets the value of the **TRIGOUTSPTE** output signal in integration mode: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.

3.3.12 Integration Mode ATB Data 0 register

The STMITATBDATA0 register characteristics are:

Purpose Controls the value of the **ATDATAM** output signal in integration mode:

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-12 on page 3-16 shows the bit assignments.

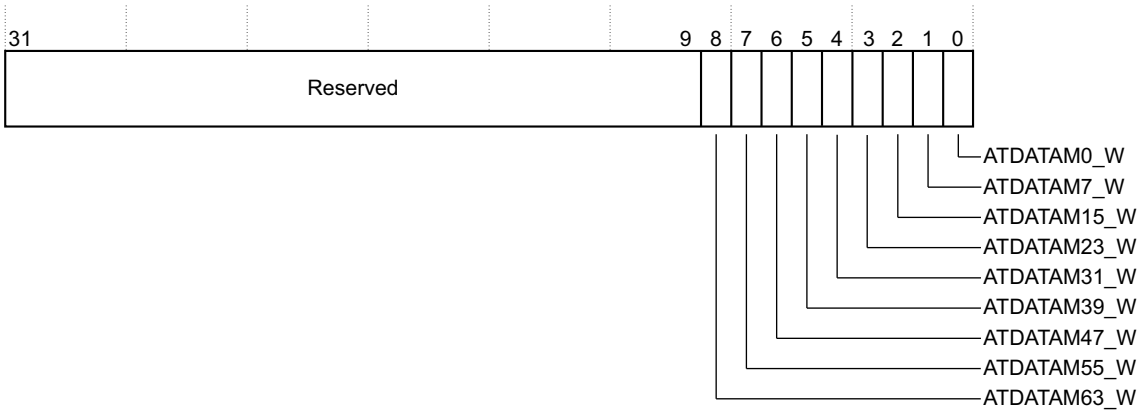


Figure 3-12 STMITATBDATA0 register bit assignments

Table 3-13 shows the bit assignments.

Table 3-13 STMITATBDATA0 register bit assignments

Bits	Name	Function
[31:9]	Reserved	Reserved.
[8]	ATDATAM63_W	Sets the value of the **ATDATAM[63]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[7]	ATDATAM55_W	Sets the value of the **ATDATAM[55]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[6]	ATDATAM47_W	Sets the value of the **ATDATAM[47]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[5]	ATDATAM39_W	Sets the value of the **ATDATAM[39]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[4]	ATDATAM31_W	Sets the value of the **ATDATAM[31]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[3]	ATDATAM23_W	Sets the value of the **ATDATAM[23]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.

Table 3-13 STMITATBDATA0 register bit assignments (continued)

Bits	Name	Function
[2]	ATDATAM15_W	Sets the value of the **ATDATAM[15]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[1]	ATDATAM7_W	Sets the value of the **ATDATAM[7]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[0]	ATDATAM0_W	Sets the value of the **ATDATAM[0]** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.

3.3.13 Integration Mode ATB Control 2 register

The STMITATBCTR2 register characteristics are:

Purpose Returns the value of the **ATREADYM** and **AFVALIDM** input signals in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-13 shows the bit assignments.

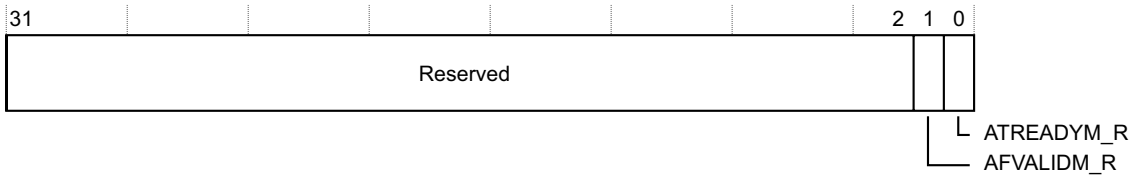


Figure 3-13 STMITATBCTR2 register bit assignments

Table 3-14 shows the bit assignments.

Table 3-14 STMITATBCTR2 register bit assignments

Bits	Name	Function
[31:2]	Reserved	Reserved.
[1]	AFVALIDM_R	Reads the value of the **AFVALIDM** input signal: `0b1` The signal is at logic `0b1`. `0b0` The signal is at logic `0b0`.
[0]	ATREADYM_R	Reads the value of the **ATREADYM** input signal: `0b1` The signal is at logic `0b1`. `0b0` The signal is at logic `0b0`.

3.3.14 Integration Mode ATB Identification register

The STMITATBID register characteristics are:

Purpose Controls the value of the **ATIDM** output signal in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-14 shows the bit assignments.

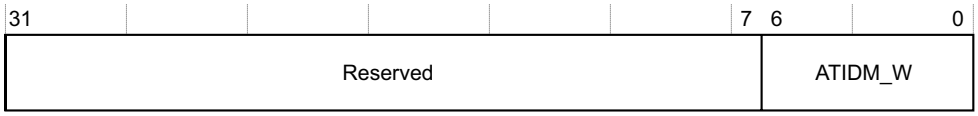


Figure 3-14 STMITATBID register bit assignments

Table 3-15 shows the bit assignments.

Table 3-15 STMITATBID register bit assignments

Bits	Name	Function
[31:7]	Reserved	Reserved.
[6:0]	ATIDM_W	Sets the value of the **ATIDM** output signal.

3.3.15 Integration Mode ATB Control 0 register

The STMITATBCTR0 register characteristics are:

Purpose Controls the value of the **ATVALIDM**, **AFREADYM**, and **ATBYTESM** output signals in integration mode.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-15 shows the bit assignments.

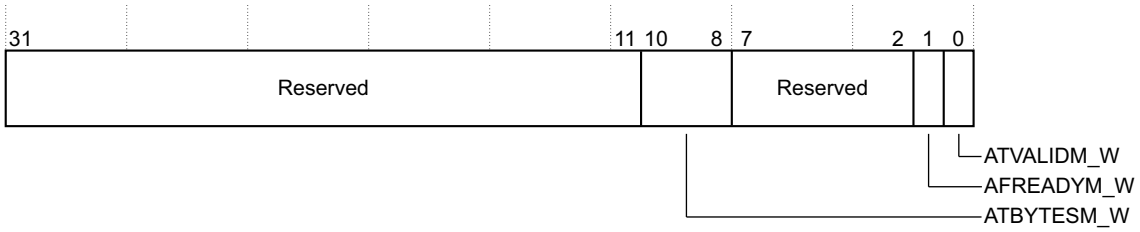


Figure 3-15 STMITATBCTR0 register bit assignments

Table 3-16 shows the bit assignments.

Table 3-16 STMITATBCTR0 register bit assignments

Bits	Name	Function
[31:11]	Reserved	Reserved.
[10:8]	ATBYTESM_W	Sets the value of the **ATBYTESM** output signal: `0b111` Drive logic `0b111`. `0b110` Drive logic `0b110`. `0b101` Drive logic `0b101`. `0b100` Drive logic `0b100`. `0b011` Drive logic `0b011`. `0b010` Drive logic `0b010`. `0b001` Drive logic `0b001`. `0b000` Drive logic `0b000`.
[7:2]	Reserved	Reserved.
[1]	AFREADYM_W	Sets the value of the **AFREADYM** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.
[0]	ATVALIDM_W	Sets the value of the **ATVALIDM** output signal: `0b1` Drive logic `0b1`. `0b0` Drive logic `0b0`.

3.3.16 Integration Mode Control register

The STMITCTRL register characteristics are:

Purpose Used to switch between functional mode and integration mode. The default behavior is functional mode. In integration mode, you can directly control the inputs and outputs of the component for integration testing and topology solving.

Note

When a device has been in integration mode, it might not function with the original behavior. After performing integration or topology detection, you must reset the system to ensure the correct behavior of CoreSight and other connected system components that are affected by the integration or topology detection.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-16 shows the STMITCTRL register bit assignments.

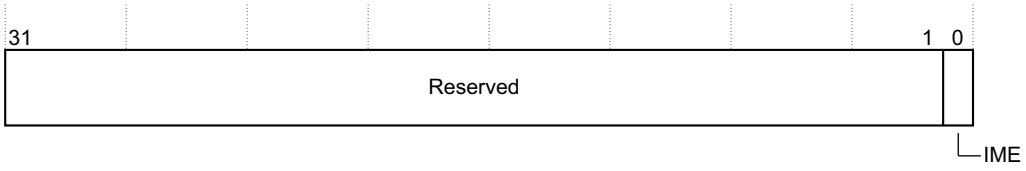


Figure 3-16 STMITCTRL register bit assignments

Table 3-17 shows the STMITCTRL bit assignments.

Table 3-17 STMITCTRL register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved.
[0]	IME	Enables the component to switch between functional and integration mode. `0b1` Enable integration mode. `0b0` Disable integration mode.

3.3.17 Lock Access Register

The STMLAR characteristics are:

Purpose Enables write access to device registers.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-17 shows the STMLAR bit assignments.

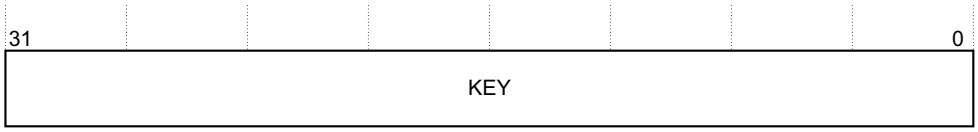


Figure 3-17 STMLAR bit assignments

Table 3-18 shows the STMLAR bit assignments.

Table 3-18 STMLAR bit assignments

Bits	Name	Function
[31:0]	KEY	Ignores writes when the **PADDRDBG31** signal is HIGH. When the **PADDRDBG31** signal is LOW, a write of `0xC5ACCE55` unlocks the lock control mechanism, enabling further writes with the **PADDRDBG31** signal LOW. Other values lock the lock control mechanism, preventing further writes with the **PADDRDBG31** signal LOW, except to the STMLAR.

3.3.18 Lock Status Register

The STMLSR characteristics are:

Purpose Indicates the status of the lock control mechanism. This lock prevents accidental writes by code under debug. The lock mechanism does not impact accesses to the extended stimulus port registers. This register must always be present although there might not be any lock access control mechanism. The lock mechanism, where present and locked, blocks write accesses to any register, except the STMLAR. The lock mechanism is only present for accesses with the **PADDRDBG31** signal LOW.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-18 shows the STMLSR bit assignments.

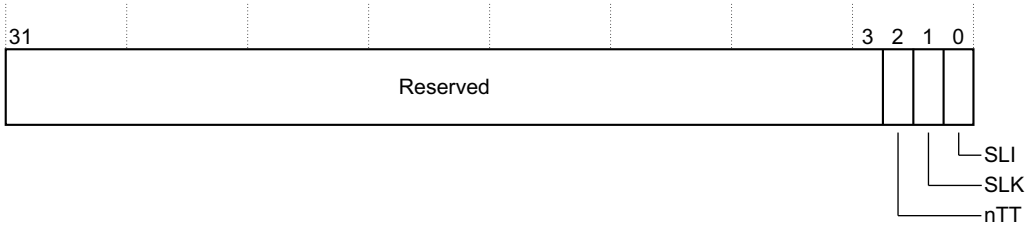


Figure 3-18 STMLSR bit assignments

Table 3-19 shows the STMLSR bit assignments.

Table 3-19 STMLSR bit assignments

Bits	Name	Function
[31:3]	Reserved	Reserved.
[2]	nTT	Indicates whether the component implements the STMLAR as 8 bit or 32 bit. `0b0` 32 bit.
[1]	SLK	Returns the current status of the lock. `0b0` Enables write access to the STM. For read accesses to the STMLSR when the **PADDRDBG31** signal is HIGH, this bit is always `0b0`. `0b1` Blocks write access to the STM. The STM ignores all write accesses to the registers when the **PADDRDBG31** signal is LOW. Reads are permitted.
[0]	SLI	Indicates that a lock control mechanism exists for the device. `0b0` No lock control mechanism exists. The STM ignores writes to the STMLAR. For read accesses to the STMLSR when the **PADDRDBG31** signal is HIGH, this bit is always `0b0`. `0b1` Lock control mechanism is present. For read accesses to the STMLSR when the **PADDRDBG31** signal is LOW, this bit is always `0b1`.

3.3.19 Authentication Status register

The STMAUTHSTATUS register characteristics are:

Purpose Reports the required security level and current status of the authentication interface.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-19 shows the bit assignments.

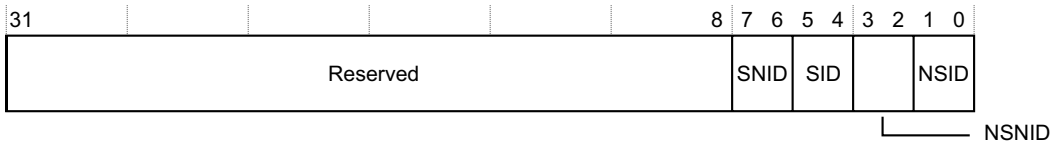


Figure 3-19 STMAUTHSTATUS register bit assignments

Table 3-20 shows the bit assignments.

Table 3-20 STMAUTHSTATUS register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:6]	SNID	Indicates the security level for Secure non-invasive debug: 0b10 Disabled. 0b11 Enabled.
[5:4]	SID	Indicates the security level for Secure invasive debug: 0b10 Disabled. 0b11 Enabled.
[3:2]	NSNID	Indicates the security level for Non-secure non-invasive debug: 0b10 Disabled. 0b11 Enabled.
[1:0]	NSID	Indicates the security level for Non-secure invasive debug: 0b10 Disabled. 0b11 Enabled.

3.3.20 Device Architecture register

The STMDEVARCH register characteristics are:

Purpose Identifies the architect and architecture of the STM. For the STM-500, the architect is ARM, and the architecture is STMv1.1.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-20 shows the bit assignments.

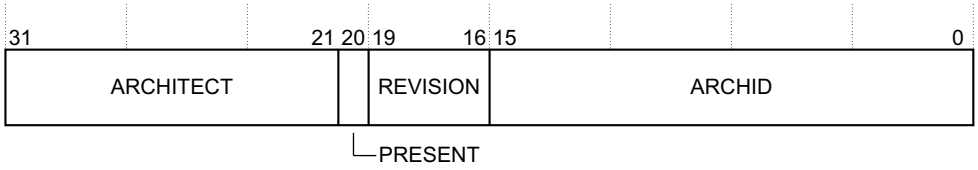


Figure 3-20 STMDEVARCH register bit assignments

Table 3-21 shows the bit assignments.

Table 3-21 STMDEVARCH register bit assignments

Bits	Name	Function
[31:21]	ARCHITECT	Defines the architect of the component: **Bits[31:28]** Indicates the JEP106 continuation code. **Bits[27:21]** Indicates the JEP106 identification code. See the *Standard Manufacturer's Identification Code* for information about JEP106. For the STM-500, ARM is the architect, and this 11-bit field returns `0x23B`.
[20]	PRESENT	Indicates the presence of the STMDEVARCH register: `0b1` The STMDEVARCH register is present.
[19:16]	REVISION	Architecture revision. Returns the revision of the architecture that the ARCHID field specifies. For the STM, this value is `0x1`, indicating the STMv1.1 architecture.
[15:0]	ARCHID	Architecture ID. Returns a value that identifies the architecture of the component. For the STM, this value is `0x0A63`, indicating the STMv1.1 architecture.

3.3.21 Device Configuration register

The STMDEVID register characteristics are:

Purpose Indicates the capabilities of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-21 shows the bit assignments.

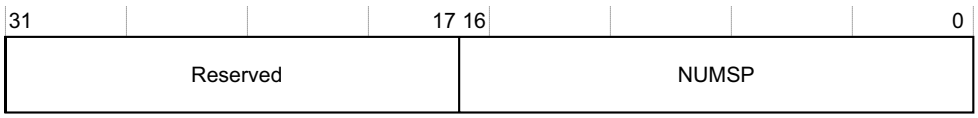


Figure 3-21 STMDEVID register bit assignments

Table 3-22 shows the bit assignments.

Table 3-22 STMDEVID register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved.
[16:0]	NUMSP	Indicates the number of stimulus ports implemented. `0x10000` 65536.

3.3.22 Device Type Identifier register

The STMDEVTYPE register characteristics are:

Purpose Provides a debugger with information about the component when the part number is not recognized. The debugger can then report this information.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-22 shows the bit assignments.

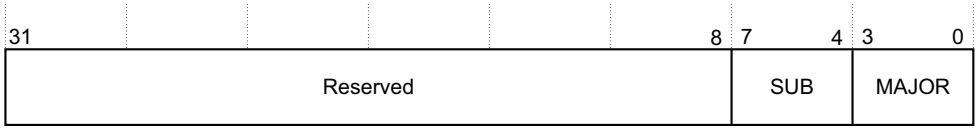


Figure 3-22 STMDEVTYPE register bit assignments

Table 3-23 shows the bit assignments.

Table 3-23 STMDEVTYPE register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	SUB	Sub-classification within the major category: `0b0110` The component generates trace based on software and hardware stimulus.
[3:0]	MAJOR	Major classification grouping for the debug or trace component: `0b0011` The component is a trace source.

3.3.23 Peripheral ID0 Register

The STMPIDR0 characteristics are:

Purpose Part of the set of peripheral identification registers. Contains part of the designer-specific part number.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-23 shows the bit assignments.

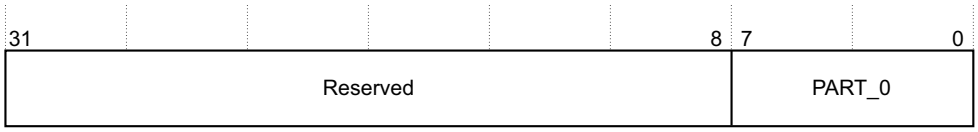


Figure 3-23 STMPIDR0 bit assignments

Table 3-24 shows the bit assignments.

Table 3-24 STMPIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	PART_0	Bits [7:0] of the component part number, specified by the designer of the component. `0b01100011` Lowest eight bits of the part number, `0x963`.

3.3.24 Peripheral ID1 Register

The STMPIDR1 characteristics are:

Purpose Part of the set of peripheral identification registers. Contains part of the designer-specific part number and part of the designer identity.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-24 shows the bit assignments.

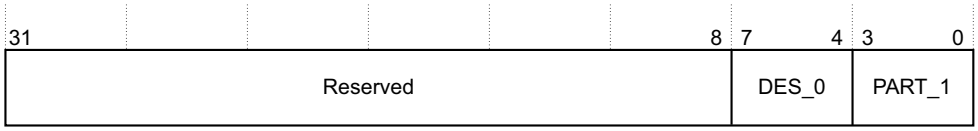


Figure 3-24 STMPIDR1 bit assignments

Table 3-25 shows the bit assignments.

Table 3-25 STMPIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	DES_0	Bits [3:0] of the JEDEC identity code indicating the designer of the component, together with the continuation code. `0b1011` Lowest four bits of the JEP106 identity code.
[3:0]	PART_1	Bits [11:8] of the component part number, specified by the designer of the component. `0b1001` Upper four bits of the part number, `0x963`.

3.3.25 Peripheral ID2 Register

The STMPIDR2 characteristics are:

Purpose Part of the set of peripheral identification registers. Contains part of the designer identity and the product revision.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-25 shows the bit assignments.

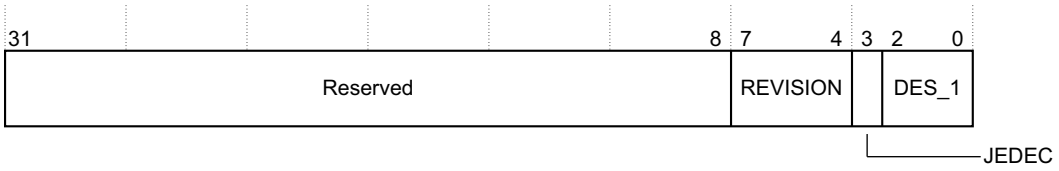


Figure 3-25 STMPIDR2 bit assignments

Table 3-26 shows the bit assignments.

Table 3-26 STMPIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	REVISION	An incremental value starting at `0x0` for the first design of this component. The value increases by one for both major and minor revisions and is used as a look-up to establish the exact major and minor revision. `0b0000` The device is at r0p0.
[3]	JEDEC	Indicates the use of a JEDEC assigned value. This bit is always set. `0b1` The designer ID is specified by JEDEC (`http://www.jedec.org`).
[2:0]	DES_1	Bits [6:4] of the JEDEC identity code indicating the designer of the component, together with the continuation code. `0b011` Upper three bits of the JEP106 identity code.

3.3.26 Peripheral ID3 Register

The STMPIDR3 characteristics are:

Purpose Part of the set of peripheral identification registers. Contains the REVAND and CMOD bit fields.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-26 shows the bit assignments.

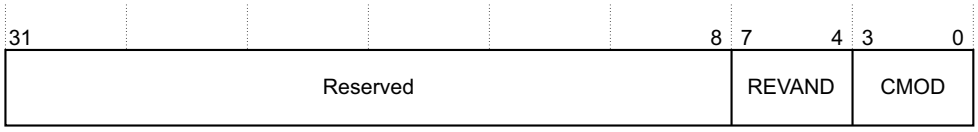


Figure 3-26 STMPIDR3 bit assignments

Table 3-27 shows the bit assignments.

Table 3-27 STMPIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	REVAND	Indicates minor errata fixes specific to the design, for example metal fixes after implementation. In most cases this field is zero. ARM recommends that the component designers ensure that the bit field can be changed by a metal fix if required, for example by driving the bit field from registers that reset to zero. `0x0` No metal fixes in the component.
[3:0]	CMOD	Where the component is reusable IP, this value indicates whether the customer has modified the behavior of the component. In most cases this field is zero. `0x0` No modifications made.

3.3.27 Peripheral ID4 Register

The STMPIDR4 characteristics are:

Purpose Part of the set of peripheral identification registers. Contains part of the designer identity and the memory footprint indicator.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-27 shows the bit assignments.

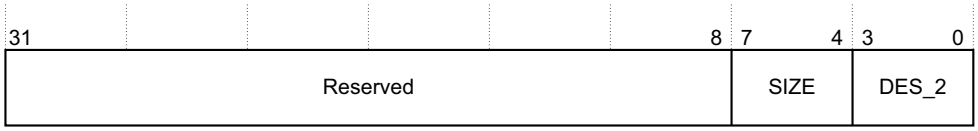


Figure 3-27 STMPIDR4 bit assignments

Table 3-28 shows the bit assignments.

Table 3-28 STMPIDR4 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	SIZE	This 4-bit value indicates the total contiguous size of the memory window used by the component in powers of two from the standard 4KB. If a component only requires the standard 4KB, this bit field must read as `0x0`, 4KB only. For 8KB set to `0x1`, for 16KB set to `0x2`, for 32KB set to `0x3`, and so on. `0x0` The device only occupies 4KB of memory.
[3:0]	DES_2	JEDEC continuation code indicating the designer of the component, together with the identity code. `0x4` ARM JEDEC identity code is on the fifth bank.

3.3.28 Component ID0 Register

The STMCIDR0 characteristics are:

Purpose A component identification register that indicates the presence of identification registers.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-28 shows the bit assignments.

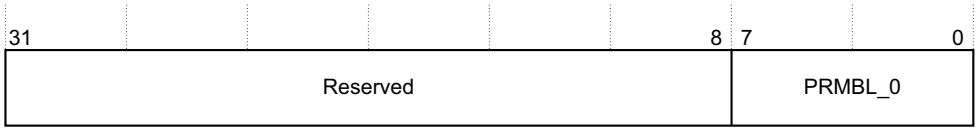


Figure 3-28 STMCIDR0 bit assignments

Table 3-29 shows the bit assignments.

Table 3-29 STMCIDR0 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	PRMBL_0	Contains bits [31:21] of the component identification. 0x0D Identification value.

3.3.29 Component ID1 Register

The STMCIDR1 characteristics are:

Purpose A component identification register that indicates the presence of identification registers. This register also indicates the component class.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-29 shows the bit assignments.

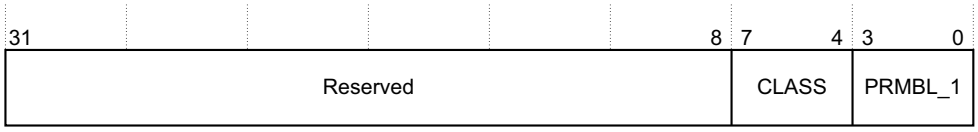


Figure 3-29 STMCIDR1 bit assignments

Table 3-30 shows the bit assignments.

Table 3-30 STMCIDR1 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	CLASS	Class of the component, for example, the ROM table or CoreSight component. 0x9 Indicates the component is a CoreSight component.
[3:0]	PRMBL_1	Contains bits [19:16] of the component identification. 0x0 Identification value.

3.3.30 Component ID2 Register

The STMCIDR2 characteristics are:

Purpose A component identification register that indicates the presence of identification registers.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Figure 3-30 on page 3-29 shows the bit assignments.

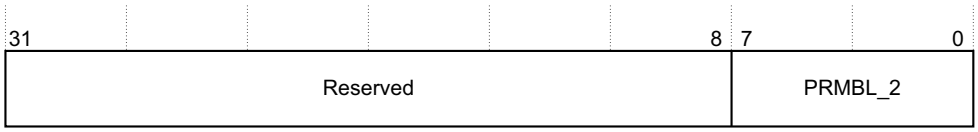


Figure 3-30 STMCIDR2 bit assignments

Table 3-31 shows the bit assignments.

Table 3-31 STMCIDR2 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	PRMBL_2	Contains bits [15:8] of the component identification. `0x05` Identification value.

3.3.31 Component ID3 Register

The STMCIDR3 characteristics are:

Purpose A component identification register that indicates the presence of identification registers.

Usage constraints There are no usage constraints.

Figure 3-31 shows the bit assignments.

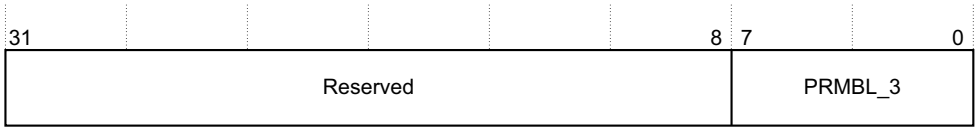


Figure 3-31 STMCIDR3 bit assignments

Table 3-32 shows the bit assignments.

Table 3-32 STMCIDR3 bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	PRMBL_3	Contains bits [7:0] of the component identification. `0xB1` Identification value.

Appendix A
Signal Descriptions

This appendix describes the signals used in the STM-500. It contains the following section:

- *Clocks and resets* on page A-2.
- *AXI slave interface signals* on page A-3.
- *Debug APB interface signals* on page A-5.
- *ATB master interface signals* on page A-6.
- *Hardware event observation interface signals* on page A-7.
- *DMA peripheral request interface signals* on page A-8.
- *Timestamp port signals* on page A-9.
- *Authentication interface signals* on page A-10.
- *Non-secure guaranteed interface signals* on page A-11.
- *Cross-trigger interface signals* on page A-12.
- *Test interface signals* on page A-13.
- *AXI low-power interface signals* on page A-14

A.1 Clocks and resets

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal name	Type	Description
CLK	Input	STM clock. Clocks all interfaces on the STM.
ARESETn	Input	AXI reset. Resets the AXI slave, DMA peripheral request, and AXI low-power interfaces.
STMRESETn	Input	STM reset. Resets all modules in the STM except the AXI slave, DMA peripheral request, and AXI low-power interfaces.

A.2 AXI slave interface signals

Table A-2 shows the AXI slave interface signals.

Table A-2 AXI slave interface signals

Signal name	Type	Description
ARIDS[AXI_ID_WIDTH–1:0]	Input	Read address ID
ARADDRS[31:0]	Input	Read address
ARLENS[7:0]	Input	Read address burst length
ARSIZES[2:0]	Input	Read address burst size
ARBURSTS[1:0]	Input	Read address burst type
ARLOCKS	Input	Read address lock type
ARCACHES[3:0]	Input	Read address cache type
ARPROTS[2:0]	Input	Read address protection type
ARVALIDS	Input	Read address valid
ARREADYS	Output	Read address ready
RREADYS	Input	Read response ready
RIDS[AXI_ID_WIDTH–1:0]	Output	Read ID
RDATAS[63:0]	Output	Read data
RRESPS[1:0]	Output	Read response
RLASTS	Output	Read last
RVALIDS	Output	Read response valid
AWIDS[AXI_ID_WIDTH–1:0]	Input	Write address ID
AWADDRS[31:0]	Input	Write address
AWLENS[7:0]	Input	Write address burst length
AWSIZES[2:0]	Input	Write address burst size
AWBURSTS[1:0]	Input	Write address burst type
AWLOCKS	Input	Write address lock type
AWCACHES[3:0]	Input	Write address cache type
AWPROTS[2:0]	Input	Write address protection type
AWVALIDS	Input	Write address valid
AWREADYS	Output	Write address ready
WDATAS[63:0]	Input	Write data
WSTRBS[7:0]	Input	Write data strobes
WLASTS	Input	Write last
WVALIDS	Input	Write valid

Table A-2 AXI slave interface signals (continued)

Signal name	Type	Description
WREADYS	Output	Write ready
BREADYS	Input	Write response ready
BIDS[AXI_ID_WIDTH–1:0]	Output	Write response ID
BRESPS[1:0]	Output	Write response
BVALIDS	Output	Write response valid

A.3 Debug APB interface signals

Table A-3 shows the debug APB interface signals.

Table A-3 Debug APB interface signals

Signal name	Type	Description
PSELDBG	Input	Select
PENABLEDBG	Input	Enable
PWRITEDBG	Input	Peripheral write
PADDRDBG[11:2]	Input	Address. The STM supports only word-aligned addresses, and assumes bits [1:0] to be zero.
PADDRDBG31	Input	Indicates source of APB access. Internal accesses have this signal LOW and external accesses have this signal HIGH.
PWDATADBG[31:0]	Input	Write data
PREADYDBG	Output	Peripheral ready
PSLVERRDBG[a]	Output	Slave error
PRDATADBG[31:0]	Output	Read data

a. The STM permanently drives this output LOW.

A.4 ATB master interface signals

This interface outputs the trace data for collection. You must connect the interface to ATB-compliant trace collection components. Table A-4 shows the ATB master interface signals.

Table A-4 ATB master interface signals

Signal name	Type	Description
ATVALIDM	Output	A transfer is valid during this cycle. All other AT signals must be ignored if this is low.
ATBYTESM[2:0]	Output	Number of bytes on **ATDATAM** to be captured minus 1.
ATDATAM[63:0]	Output	Trace data
ATIDM[6:0]	Output	An ID that uniquely identifies the source of the trace.
ATREADYM	Input	Slave is ready to accept data.
AFVALIDM	Input	ATB flush request. All buffers must be flushed because trace capture is about to stop.
AFREADYM	Output	ATB flush acknowledge. Asserted when the buffers are flushed.
SYNCREQM	Input	Trace synchronization request. This is used to indicate that the STM must insert an ASYNC-VERSION sequence into the trace output stream.

A.5 Hardware event observation interface signals

Table A-5 shows the hardware event observation interface signals.

Table A-5 Hardware event observation interface signals

Signal name	Type	Description
HWEVENTS[63:0]	Input	Hardware event observation port. Each hardware event to be observed must be connected to one bit of this bus.
HEEXTMUX[7:0]	Output	Output to control external multiplexing of many hardware events onto the **HWEVENTS** input.

A.6 DMA peripheral request interface signals

Table A-6 shows the DMA peripheral request interface signals.

Table A-6 DMA peripheral request interface

Signal name	Type	Description
DRVALID	Output	DMA request handshake valid
DRTYPE[1:0]	Output	DMA request type
DRLAST[a]	Output	DMA last request
DRREADY	Input	DMA request handshake ready
DAVALID	Input	DMA acknowledge handshake valid
DATYPE[1:0]	Input	DMA acknowledge handshake
DAREADY	Output	DMA acknowledge handshake ready

a. The STM permanently drives this output LOW.

See the *ARM® CoreLink™ DMA-330 DMA Controller Technical Reference Manual* for more information about these signals.

A.7 Timestamp port signals

The STM receives timestamp information from the system in this interface. Table A-7 shows the timestamp port signals.

Table A-7 Timestamp port signals

Signal name	Type	Description
TSVALUE[63:0]	Input	Timestamp value, encoded as a natural binary number

A.8 Authentication interface signals

This interface provides connections for the CoreSight Authentication Interface. Table A-8 shows the authentication interface signals.

Table A-8 Authentication interface signals

Signal name	Type	Description
DBGEN	Input	Invasive debug enable
NIDEN	Input	Non-invasive debug enable
SPIDEN	Input	Secure invasive debug enable
SPNIDEN	Input	Secure non-invasive debug enable

A.9 Non-secure guaranteed interface signals

This interface provides control over the behavior of Non-secure AXI accesses to guaranteed locations. Table A-9 shows the Non-secure guaranteed interface signals.

Table A-9 Non-secure guaranteed interface signals

Signal name	Type	Description
NSGUAREN	Input	Enable Non-secure guaranteed stimulus port accesses.

A.10 Cross-trigger interface signals

Table A-10 shows the cross-trigger interface signals.

Table A-10 Cross-trigger interface signals

Signal name	Type	Description
TRIGOUTSPTE	Output	Trigger output. The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMSPTER.
TRIGOUTSW	Output	Trigger output. The STM asserts this signal for one clock cycle when a trigger event is generated on writes to a TRIG location in the extended stimulus port registers.
TRIGOUTHETE	Output	Trigger output. The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMHETER.
ASYNCOUT	Output	Alignment synchronization output. The STM asserts this signal for one clock cycle when an ASYNC-VERSION-FREQ sequence is output on the ATB interface, and the **ASYNCOUT** signal can be used for cross-triggering.

A.11 Test interface signals

Table A-11 shows the test interface signals.

Table A-11 Test interface signals

Signal name	Type	Description
DFTCLKCGEN	Input	Test mode clock enable. This signal forces the internal clock gate to enable the STM clock.

A.12 AXI low-power interface signals

Table A-12 shows the AXI low-power interface signals.

Table A-12 Low-power interface signals

Signal name	Type	Description
AXIQREQn	Input	AXI quiescence request. This signal is asserted by a clock controller to request the AXI and DMA interfaces to enter a quiescent state. This signal is active LOW.
AXIQACCEPTn	Output	AXI quiescence request acceptance. This signal is asserted in response to a quiescence request on the **AXIQREQn** signal, if the AXI and DMA interfaces are idle and can have the clock stopped. This signal is active LOW.
AXIQDENY	Output	AXI quiescence request denial. This signal is asserted in response to a quiescence request on the **AXIQREQn** signal, if the AXI and DMA interfaces are not idle. This signal is active HIGH.
AXIQACTIVE	Output	Indicates whether the STM requires a clock. This signal is asserted whenever the AXI and DMA interfaces are not idle and require a clock. This signal is active HIGH.
AWAKEUP	Input	Indicates whether the AXI interface is in use. This signal is active HIGH.

A.13 STM low-power interface signals

Table A-13 shows the STM low-power interface signals.

Table A-13 STM low-power interface signals

Signal name	Type	Description
STMQREQn	Input	STM quiescence request. This signal is asserted by a clock controller to request the STM to enter a quiescent state. This signal is active LOW.
STMQACCEPTn	Output	STM quiescence request acceptance. This signal is asserted in response to a quiescence request on the **STMQREQn** signal, if the STM is idle and can have its clock stopped. This signal is active LOW.
STMQDENY	Output	STM quiescence request denial. This signal is asserted in response to a quiescence request on the **STMQREQn** signal, if the STM is not idle. This signal is active HIGH.
STMQACTIVE	Output	Indicates whether the STM requires a clock. This signal is asserted whenever the STM is not idle and requires a clock. This signal is active HIGH.
PWAKEUP	Input	Indicates whether the APB interface is in use. This signal is active HIGH.

Appendix B
Revisions

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

Table B-1 Issue A

Change	Location	Affects
No changes, first release	-	-