CoreSight™ System Trace Macrocell

Revision: r0p1

Technical Reference Manual

ARM®

ARM DDI 0444B (ID010111)

CoreSight System Trace Macrocell
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
23 April 2010	A	Non-Confidential	First release for r0p0
10 December 2010	B	Non-Confidential	First release for r0p1

Proprietary Notice

Confidentiality Status

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

Contents
CoreSight System Trace Macrocell Technical Reference Manual

Preface

This preface introduces the *CoreSight System Trace Macrocell Technical Reference Manual*. It contains the following sections:

About this book

This book is for the CoreSight *System Trace Macrocell* (STM).

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 *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.

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 Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

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

Typographical

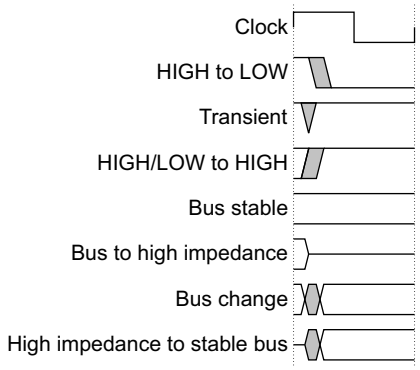
The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

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

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:

- *CoreSight System Trace Macrocell Implementation and Integration Manual* (PR430-PRDC-011726)
- *System Trace Macrocell Programmers' Model Architecture Specification* (ARM IHI 0054)
- *ARMv7-M Architecture Reference Manual* (ARM DDI 0403)
- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *Embedded Trace Macrocell Architecture Specification* (ARM IHI 0014)
- *CoreSight Architecture Specification* (ARM IHI 0029)
- *CoreSight Components Technical Reference Manual* (ARM DDI 0314)
- *CoreSight Technology System Design Guide* (ARM DGI 0012)
- *ARM Debug Interface v5 Architecture Specification* (ARM IHI 0031)
- *AMBA® AXI Protocol Specification* (ARM IHI 0022)
- *AMBA APB Protocol Specification* (ARM IHI 0024)
- *AMBA ATB Protocol Specification* (ARM IHI 0032)
- *AMBA DMA Controller DMA-330 Technical Reference Manual* (ARM DDI 0424)
- *RealView® ICE and RealView Trace User Guide* (ARM DUI 0155).

Other publications

This section lists relevant documents published by third parties:

- *MIPI System Trace Protocol version 2 (STPv2)*.

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 0444B
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the CoreSight *System Trace Macrocell* (STM). It contains the following sections:

1.1 About the System Trace Macrocell

The STM is a trace source that is integrated into a CoreSight system, 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:

- A dedicated AXI slave for receiving instrumentation. This is significantly higher performance than the APB interface of the ITM, and is separate to the APB interface for programming the STM registers.
- 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, enable significant scalability, with 16 stimulus ports per 4KB page. 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 words of data, reduces the likelihood of the FIFO becoming full.
- Timestamping can be requested for each write independently, based on the address written to. Bandwidth can be optimized by requesting a timestamp for only one write 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, PTM trace.

In addition to the AXI slave, the STM provides a hardware event interface. The STM traces when rising edges occur on signals connected to this interface. Alternatively, advanced custom system tracing features can be implemented 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.

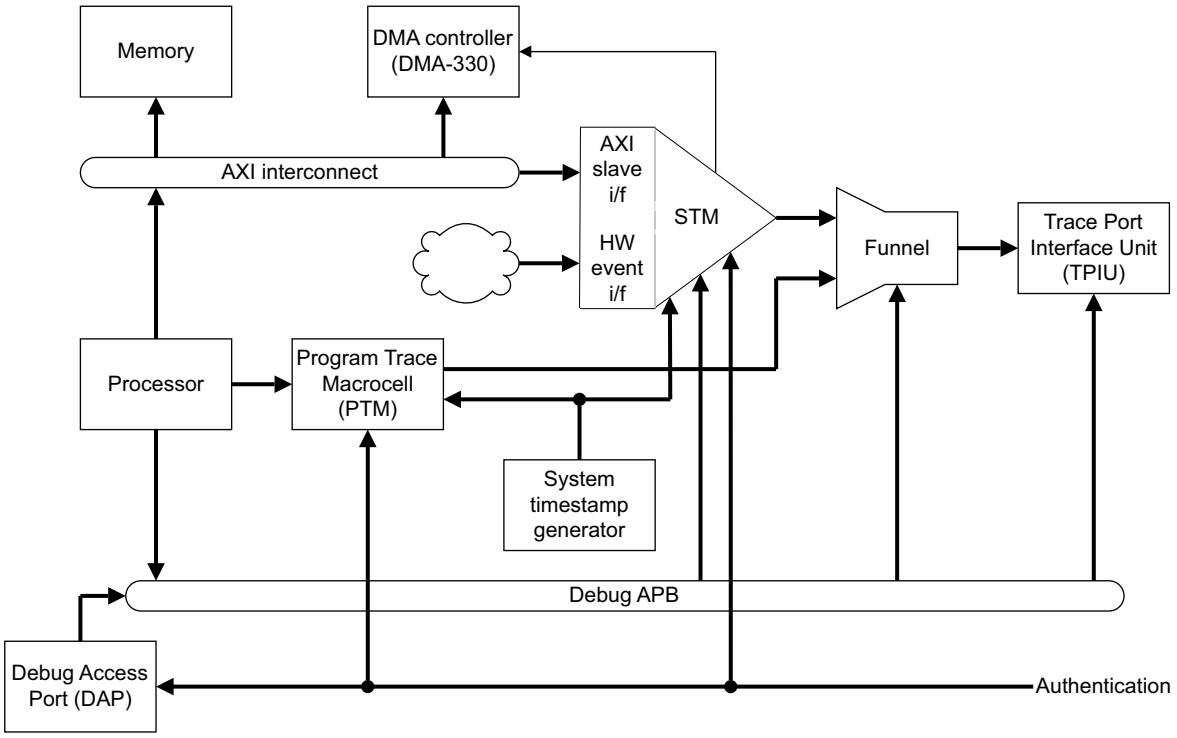


Figure 1-1 STM system integration

The STM AXI slave is connected to a system interconnect which enables all system masters, such as processors and DMA controllers, to generate trace by writing to the STM stimulus ports.

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

For configuration purposes, the STM is connected to Debug APB so that it can be accessed by off-chip and on-chip debug agents.

CoreSight authentication signals are used to control debug permissions.

The STM trace stream is output through the ATB interface and 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 STM architecture version 1.0. See the *System Trace Macrocell Programmers' Model Architecture Specification*.

1.2.3 CoreSight Architecture

The STM implements CoreSight architecture version 1.0. See the *CoreSight Architecture Specification*.

1.2.4 Advanced Microcontroller Bus Architecture

The STM complies with the AMBA 3 protocol. See the *AMBA AXI Protocol Specification* and *AMBA 3 APB Protocol Specification*.

1.3 Features

The STM has the following features:

- fully synchronous design with one clock and two resets
- one 32-bit *Advanced eXtensible Interface* (AXI) slave interface for extended stimulus port inputs
- one hardware event observation interface tracing 32 hardware events
- one 32-bit debug *Advanced Peripheral Bus* (APB) slave interface for configuration and status
- one 32-bit *Advanced Trace Bus* (ATB) master interface for trace output
- one *Direct Memory Access* (DMA) peripheral request interface compatible with the AMBA DMA Controller DMA-330
- two depth-configurable *First In First Out* (FIFO) buffers for usage-optimized configurability:
  - data FIFO
  - channel information FIFO.
- 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
- internal and external source for *System Trace Protocol version 2* (STPv2) synchronization
- timestamping of trace events.

The STM architecture has many implementation-defined options. Table 1-1 shows the configuration implemented by this STM.

Table 1-1 STM configuration

Feature	Configuration
Trace protocol	STPv2
Timestamping	Absolute
STMTSFREQR	Read-write
STMTSSTIMR	Implemented
STMSYNCR	Implemented
Claim tags	4
TRACEID	CoreSight ATB plus ATB trigger
Trigger control	Multi-shot and single-shot
STMTCSR.TSPRESCALE	Not implemented
STMTCSR.HWTEN	Not implemented

Table 1-1 STM configuration (continued)

Feature	Configuration
STMTCSR.SYNCEN	Always reads as b1
STMTCSR.SWOEN	Not implemented
Number of stimulus ports	65536
Number of masters	Minimum of 2
Stimulus port types	Extended only
Fundamental data size	32
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 HW events	32
STMHETER	Implemented
HW error detection	Implemented
STMHEMASTR	Read only
Data compression on HW trace	Programmable

1.4 Interfaces

The STM has the following external interfaces:

- AXI slave
- hardware event observation interface
- DMA peripheral request interface
- debug APB slave interface
- ATB master interface
- synchronization request interface
- timestamp port interface
- authentication interface
- non-secure guaranteed stimulus port access enable interface
- cross-trigger interface.

See *Interfaces* on page 2-3 for more information on 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 one 32-bit word of data and up to 16 bits of timestamp. 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 define the presence of the hardware event observation interface and the width of 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 data FIFO buffer
CHN_FIFO_DEPTH	4, 8, 16, 32[a]	Depth of channel FIFO buffer
HWEVOBIF_PRESENT	FALSE	Hardware event observation interface not present
	TRUE	Hardware event observation interface present
AXI_ID_WIDTH	2-24	Width of AXI IDs, including AWID, WID, BID, ARID, and RID

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 an 8 deep channel FIFO buffer and a 16 deep data FIFO buffer.

1.6 Test features

The STM includes clock gating circuitry to save power when the STM is disabled. Two *Design For Test* (DFT) ports are included, so that:

- the clock can be enabled during test of the STM
- the STM clock can be disabled during test of other components in the system, to save power.

The DFT ports are:

DFTTESTMODE

Forces the STM clock on during DFT shift.

DFTCLKDISABLE

Can be used to shut down the STM clock during testing of other components in the system.

Note

The clock enables are directly controllable during test mode. You can turn the clocks off during test instead of forcing the clock enables on, to save power during testing.

1.7 Product documentation, design flow, and architecture

This section describes the STM books, 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 books 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 *Implementation and Integration Manual* (IIM) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the *Register Transfer Level* (RTL) with the build configuration options
- How to integrate the STM into a 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 book that is only available to licensees.

1.7.2 Design flow

The STM is delivered as synthesizable RTL. Before it can be used in a product, it must be integrated and implemented. 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 pre-processed. 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.

r0p0-r0p1 No functional changes.

Chapter 2
Functional Description

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

2.1 About the functions

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

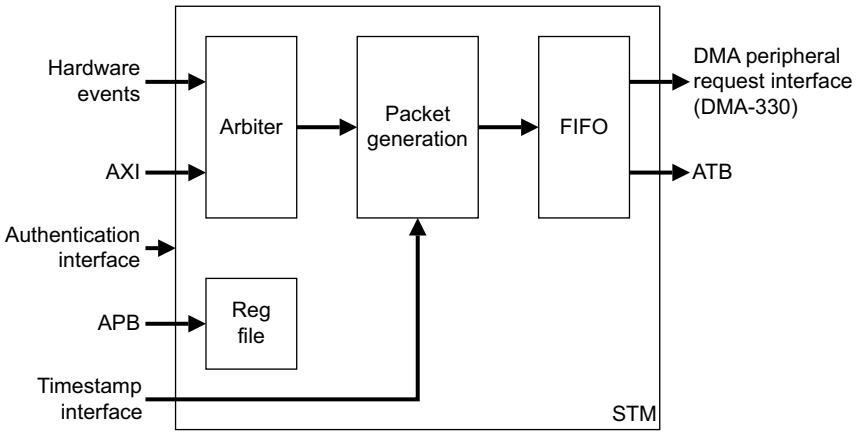


Figure 2-1 STM block diagram

The STM implements tracing of software writes to its stimulus ports using the AXI and tracing of hardware events.

The selection of which stimulus ports and which hardware events are traced is based on programmed controls and the state of the authentication interface.

Traced transactions from the two interfaces, AXI and hardware, are arbitrated with higher priority being assigned to AXI transactions. Trace data is then presented to the packet generation logic where it is timestamped and organized according to the STPv2 protocol.

STPv2 data is packed into the trace stream and is output on ATB interface.

In case of overflow, that is, when the STM FIFO is full, the STM can either stall the AXI or drop data with overflow indicated in the trace stream. There is no stall mechanism on the hardware interface. Hardware events can either be silently dropped or an overflow condition is indicated in the trace stream.

The STM periodically outputs a synchronization sequence to enable the trace receiver to align on the packet boundary in the trace stream.

2.2 Interfaces

The STM interfaces are not configurable except for the AXI ID width. See the *CoreSight System Trace Macrocell Implementation and Integration Manual* for more information about how to connect these interfaces in a system. The STM has the following external interfaces:

AXI slave This interface connects the STM to the system bus. This design provides a 32-bit AXI slave. See the *AMBA AXI Protocol Specification* for more information on the AXI signals.

This interface occupies space in the memory map which can be written to, to generate trace.

Hardware event observation interface

This interface consists of 32 input signals, and connects to various signals from the system, such as interrupt lines, DMA request lines, and *Cross-Trigger Interface* (CTI) trigger outputs.

Hardware events on this interface are captured and trace is generated based on captured events. See *Hardware event tracing* on page 2-18 for more information.

DMA peripheral request interface

This interface connects to an AMBA DMA Controller DMA-330. When the STM is programmed to initiate a DMA transfer, this interface requests the DMA controller to write to the STM AXI. See *DMA control* on page 2-19 for more information.

Debug APB slave interface

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

See the *AMBA APB Protocol Specification* and the *CoreSight Architecture Specification* for more information on the debug APB signals.

ATB master interface

This is the interface for trace output. It also provides handshaking signals for making flush requests to the STM. The interface width is 32 bits.

See the *AMBA ATB Protocol Specification* for more information on the ATB signals.

External synchronization request

This is the interface for outputting synchronization requests to the STM.

Timestamp port interface

This interface provides the timestamp that is used in timestamped trace packets. Timestamp can be encoded as natural binary or Gray code encoding:

- Timestamp encoding is defined by the state of **TSNATURAL** input. **TSNATURAL** must be tied off at implementation.
- The maximum timestamp widths supported are 48 and 64 bits. The maximum output timestamp width is defined by the state of the **TSMAXWIDTH** configuration input. **TSMAXWIDTH** must be tied off at implementation.

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 for more information.

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 for more information.

Cross-trigger interface

Three trigger output ports, **TRIGOUTSPTE**, **TRIGOUTSW**, and **TRIGOUTHETE**, are implemented to connect to a cross-trigger interface in a CoreSight system, to indicate trigger events. These correspond to the dedicated trigger outputs described in the *System Trace Macrocell Programmers' Model Architecture Specification*.

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

External synchronization interface

The **SYNCREQ** input port enables an external component to control the frequency of periodic synchronisation. This signal is provided for compatibility with future architectures and can be tied LOW in most designs.

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.

The APB interface provides a **PCLKENDBG** clock enable input to enable you to connect the STM to APB masters running on a slower APB clock that is an integer division of the STM **CLK**.

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

The architectural clock gating is transparent to the programmer. The internal clock is enabled when the STM registers are accessed or when tracing is enabled. Because the architectural clock gating is present, you do not have to gate the clock at STM block level.

2.3.2 Resets

The STM has two asynchronous active LOW resets:

ARESETn This is used to reset the AXI slave and DMA peripheral request blocks. This is the **ARESETn** reset domain.

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

The STM supports both resets being independently asserted, and can be:

- trace through AXI reset, **ARESETn** asserted, **STMRESETn** not asserted
- debug logic being reset without resetting rest of the system, **ARESETn** not asserted, **STMRESETn** asserted.

2.4 Trace protocol

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

65,536 STPv2 channels are provided for flexibility and to enable additional definition of message boundaries. These channels are directly mapped to the 65,536 extended stimulus ports.

The maximum size data packet generated is 32 bits.

The following sections describe trace protocol:

- *Trace packets*
- *Alignment synchronization* on page 2-7
- *Error packets* on page 2-8
- *Word 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. This packet is sent only when the software channel is changed. If any of the upper 8 bits of the channel number change, C16 is sent. C8 is sent if the channel change is limited to any of the lower 8 bits. The default channel is 0, set when VERSION is generated.
D4, D8, D16, D32	Data	Data only.
D4M, D8M, D16M, D32M	Data	Data with marker.
D4TS, D8TS, D16TS, D32TS	Data	Data with timestamp.
D4MTS, D8MTS, D16MTS, D32MTS	Data	Data with marker and timestamp.
M8	Master	Master indicator. This packet is sent only when the source master for the trace is changed. The default master is 0, set when VERSION is generated.
MERR	Error	Master Error. Generated when written stimulus is dropped because of a full buffer.
GERR	Error	Global Error. Generated when written stimulus from multiple masters is dropped because of a full buffer and MERR cannot be sent for each master.
FLAG	Marker	Flag, can be used to 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 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 values only: 3 = STPv2 with natural binary STPv2 timestamps 4 = STPv2 with Gray coded STPv2 timestamps.
FREQ	Frequency	Indicates the frequency of the global system counter supplying timestamp information, in Hertz. This packet is always sent immediately after an ASYNC-VERSION sequence.

Note

There are a number of STPv2 trace packets that the STM does not generate. See *Constraints and limitations of use* on page 2-25 for details of these trace packets.

2.4.2 Alignment synchronization

To comply with STPv2, alignment synchronization of the trace stream is done by generating an ASYNC packet followed by a VERSION packet. If timestamping is enabled, a FREQ packet follows the VERSION packet.

The alignment synchronization packets are generated 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

An internal synchronization request is made when one of the following occurs:

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

External synchronization requests

An external synchronization requests comes from outside the STM through the **SYNCREQ** 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 0.

Synchronization request combining

Internal and external synchronization requests are combined 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 is present to address non-typical usage cases. In a typical trace scenario, do not enable this feature, that is, leave the STMAUXCR at its reset value.

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

- If STM cannot insert synchronization request, the request remains pending.
- If another synchronization request is received when there is a request pending and override control is set, the priority of synchronization request is escalated.
- Trace data is stalled if necessary while a high priority synchronization request is inserted.

Note

The STMAUXCR is implementation defined. Controls defined in this register are not defined in the *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

Error packets are generated and inserted in the trace stream when information is lost because of overflow.

The front-end interface block which experiences the master error condition, either the AXI or hardware interface, requests that an MERR packet is generated at the next opportunity. The error conditions causing MERR generation are specified with the front-end interface functionality.

MERR packets

MERR packets are master-specific. An MERR packet referring to a currently inactive master is preceded by an M8 packet. When the MERR packet is generated, the active channel is reset to 0.

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

GERR packets

GERR packets are generated and inserted in the trace stream when more than one master experiences an error condition and MERR cannot be output for each master.

GERR has an 8-bit payload which is always `0x00`.

2.4.4 Word output

The STM always tries to perform 32-bit ATB writes to maximize trace bandwidth efficiency. The STM can therefore hold trace packet nibbles until enough nibbles have been generated to enable a 32-bit write. The exception to this is 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 a smaller-size ATB write after byte aligning the data.

2.5 Timestamping

The STM supports timestamping of trace using values from a system global counter. The configuration port, **TSMAXWIDTH**, sets the maximum timestamp field width in the STM output to 48 or 64 bits:

- tie **TSMAXWIDTH** HIGH to set a full timestamp width of 64 bits
- tie **TSMAXWIDTH** LOW to set a full timestamp width of 48 bits.

The timestamp is either natural binary or Gray code encoded depending on the value of the **TSNATURAL** configuration port.

2.6 Triggering

A trigger event indicates points of interest. 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 *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 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 STMSPTER
TRIGOUTSW	Write to TRIG location
TRIGOUTHETE	Match using STMHETER

2.7 Extended stimulus port interface

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

The STM implements an AMBA 3 AXI slave with the following attributes for extended stimulus port functionality:

- write acceptance capability of 2
- write interleave depth of 1, that is, no interleaving
- read acceptance capability of 1
- read data reordering depth of 1, that is, no reordering.

Figure 2-2 shows an example AXI write sequence.

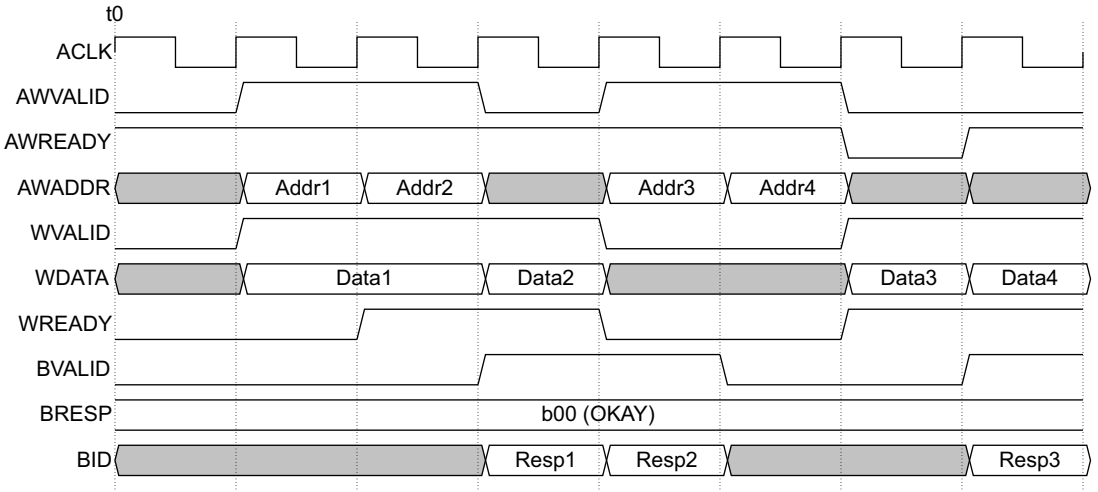


Figure 2-2 Example AXI write sequence

The following sections describe the extended stimulus port interface:

- *AXI responses*
- *STM enabled*
- *STM disabled* on page 2-16
- *AXI reads* on page 2-16
- *Stimulus port and trigger enables* on page 2-16
- *Invariant-timing packets and overflow* on page 2-16.

2.7.1 AXI responses

The STM gives an OKAY response for all transfers. Error conditions during STM operation, such as dropped requests, are indicated in the trace output.

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
- *WSTRB usage* on page 2-14

Guaranteed and invariant timing transactions

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

Behavior on guaranteed transactions

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

Only use a guaranteed write for critical trace information, because it guarantees a trace packet is generated for that write. The wait states can adversely affect system performance.

Behavior on invariant timing transactions

If an invariant timing transaction is made while the STM is unable to trace it immediately, the STM does not stall the AXI. After the address is transferred, **WREADY** is driven HIGH enabling the data to be transferred. The write data is discarded 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

Multiple STPv2 master support is implemented based on the type of AXI access, either secure or non-secure, and upper bits of AXI address.

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

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

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

In the above ID, master IDs 0-63 are allocated to secure transactions and master IDs 64-127 to non-secure transactions.

Timestamp requests

Transactions to the extended stimulus ports can include a timestamp request. You can promote transactions to the extended stimulus ports to include a timestamp request by writing b1 STMTSSTIMR.FORCETS. If transaction already includes timestamp request, FORCETS 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 the timestamp request cannot be accommodated.

Timestamping invariant timing writes

Timestamp requests in invariant timing transaction are treated as sticky. A sticky timestamp request means that if timestamp cannot be accommodated on the STM FIFO the request remains pending and timestamp is inserted at the next opportunity. Sticky timestamp requests do not cause the STM to stall.

The pending sticky timestamp state is cleared when timestamp is output regardless of whether 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.

WSTRB usage

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

Table 2-3 Supported values of WSTRBS

WSTRBS	Permitted to start a burst	Data	Packet when AWADDR[6:3] = 0
0001	Yes	**WDATAS[7:0]**	D8MTS
0010	No	**WDATAS[15:8]**	D8MTS
0100	No	**WDATAS[23:16]**	D8MTS
1000	No	**WDATAS[31:24]**	D8MTS
0011	Yes	**WDATAS[15:0]**	D16MTS
1100	No	**WDATAS[31:16]**	D16MTS
1111	Yes	**WDATAS[31:0]**	D32MTS

All other values of **WSTRBS** result in Unpredictable behavior. Unaligned transfers are not supported.

All write data bursts must be aligned so that **WDATA[0]** is the LSB at the start of the burst. Writes to other alignments are architecturally Unpredictable.

AWSIZE usage

AWSIZE is used to control the address increment between beats of an incrementing burst.

Note

AWSIZE is not used when calculating the packet type. In r0p0 of the STM, it was required that **AWSIZE** matched the packet size shown by **WSTRBS**. This requirement has been removed in r0p1.

Writes to disabled stimulus ports

The STM responds to writes to disabled stimulus ports in the same way as for invariant timing transactions. However, no error packets are added to the trace, and the request is dropped.

Access control based on AWPROT

On writes, the value of **AWPROT[1]** is checked against the current authentication status of the STM. If this value corresponds to a secure transaction when secure accesses are disabled, the access is dropped as if the stimulus port were disabled.

See the *CoreSight Architecture Specification* for more information on secure and non-secure accesses to CoreSight components.

Note

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

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

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 *CoreSight Architecture Specification* for the mapping of the 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 **NSGUAREN** is tied LOW, non-secure guaranteed accesses behave like invariant timing, that is, the AXI does not stall
- when **NSGUAREN** is tied HIGH, non-secure guaranteed accesses are enabled, that is, the AXI can stall and trace output is guaranteed.

Overrides

The debugger can override the stimulus ports selected with the STMSPOVERRIDER and STMSPMOVERRIDER to enable transactions to be always treated as guaranteed or as invariant timing.

When overriding transactions to be guaranteed, this is considered 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, 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 *CoreSight Architecture Specification* for the mapping of 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 be invariant timing.

2.7.3 STM disabled

When the STM is disabled, all writes are immediately accepted and the write data is ignored. 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 `0x00000000` as read data regardless of the STM state.

2.7.5 Stimulus port and trigger enables

Stimulus ports and triggers can be enabled and disabled using the STMSPER, STMSPTER, and STMSPSCR. See the *System Trace Macrocell Programmers' Model Architecture Specification* for more information on stimulus ports and triggers.

2.7.6 Invariant-timing packets and overflow

If a write is performed to an invariant-timing stimulus port location on the software stimulus interface, the write appears to be immediately accepted, regardless of the state of the STM buffers. If there is not enough space for the generated packet on the STM buffer, the data is discarded and does not appear in the trace output.

If data is discarded as a result of invariant timing writes made while the STM buffer is full, a MERR packet is added to the STM buffer when space next becomes available on the buffer. Only one MERR packet is generated regardless of the number of discarded transactions, until:

- at least one of any other packet type is successfully generated and added to the buffer
- the master ID is changed.

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

2.8 Hardware event tracing

The STM hardware event observation interface enables monitoring and tracing of 32 rising-edge hardware events, each of which is represented by a single bit. You can use this functionality to monitor interrupts, cross-triggers, and other signals of interest in your system. If a hardware event is asserted and is enabled, it is captured and traced as a data packet.

To enable the STM to monitor and trace cross-trigger events in the system, ARM recommends that you connect at least two trigger outputs from a CTI to hardware event inputs on the STM. ARM recommends you also 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.

Triggers can be enabled for hardware events. If triggering is enabled for a hardware event, its assertion, when captured, also causes a trigger event.

Multiple different hardware events can be merged into a single traced event.

If the same hardware event is asserted multiple times while the first captured assertion is pending and has not yet been traced, only one assertion is traced and the others are dropped. If error detection is enabled, an MERR packet is generated to indicate that merging has occurred.

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

2.9 DMA control

The STM has a DMA peripheral request interface designed to function with an AMBA DMA Controller DMA-330. This interface enables the STM to request writes to be made to its extended stimulus ports by the DMA controller. You must set up and program the DMA controller with the required transactions before this functionality is enabled in the STM. See the *AMBA DMA Controller DMA-330 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 start making DMA requests by writing b1 to the STMDMASTART.START bit. When the interface is programmed to start making requests, the interface can make requests to the DMA whenever the STM FIFO has enough free space and a DMA transaction is not currently in progress. The interface makes requests repeatedly until it is programmed to stop.
- Program the STM to stop making DMA requests by writing b1 to the STMDMASTOP.STOP bit. When the interface is programmed to stop making requests, no further requests are made to the DMA controller. 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. 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 a DMA request is started by writing b1 to the STMDMASTART.START bit, the STM initiates a burst transfer by setting **DRTYPE[1:0]** to b01.
- When a burst is acknowledged by setting **DATYPE[1:0]** to b01, a new burst transfer is initiated as soon as there is sufficient space in the FIFO as described in *FIFO level monitoring*. Single transfer acknowledges are ignored.
- When a flush is requested by setting **DATYPE[1:0]** to b10, the STM abandons a burst request and acknowledges the flush by setting **DRTYPE[1:0]** to b10. If DMA transfers are still enabled, a new burst is immediately initiated.
- **DRLAST** is always driven b0.
- **DAVALID**, **DRVALID**, **DAREADY**, and **DRREADY** function with the above signals as described in the *AMBA DMA Controller DMA-330 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 a 32-bit value of `0x0000FFFF` is written to the stimulus port and compression is enabled, a D16 packet is generated 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 the extended stimulus ports if the size of every write to the STM is already 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.

Compression can always be enabled for the hardware event observation interface without the loss of 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 **AFVALID** 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 b0. The STM drops new stimulus until it is enabled again.
- When auto-flush is enabled.
- When authentication is removed.

When a flush is requested, the STM continues outputting trace as 32-bit ATB writes as normal. However, if insufficient nibbles remain in the buffer for a 32-bit write, the STM does not wait . The most appropriate write size is chosen, and the buffer contents are padded with NULL nibbles for the ATB write.

Note

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

The following sections describe buffer flushing:

- *Override using auto-flush*
- *ATB flush request and priority inversion*
- *ATB AFREADY override* on page 2-22.

2.11.1 Override using auto-flush

You can enable the STM to output smaller-than-word amounts of data without waiting for sufficient data to complete a word transfer. To override the word output behavior, program the STMAUXCR.FIFOAF bit with a 1, which enables auto-flush of the STM FIFOs.

When auto-flush is enabled, if the FIFO contains insufficient data for a word ATB transfer and has all its other entries empty, the STM performs a smaller-sized ATB write 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 be output (20 bits), one NULL packet is inserted to make a 24-bit write
- if there are three nibbles to be output (12 bits), one NULL packet is inserted to make a 16-bit write
- if there are two nibbles to be output, no NULL packets are required.

2.11.2 ATB flush request and priority inversion

The STM acknowledges ATB flush requests by asserting an **AFREADY** output after all data present in STM before ATB flush request was made has been output.

The default behavior on 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, priority is temporarily inverted and hardware event tracing is given higher priority until it is flushed. This helps the flushed 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 cause the STM flush acknowledge to be prolonged because new AXI trace can be output before hardware historical data.

2.11.3 ATB AFREADY override

The STMAUXCR provides an override control for the **AFREADY** output. Set the STMAUXCR.AFREADYHIGH bit to drive the **AFREADY** output HIGH regardless of the state of the STM. This results in the flush being acknowledged before it has completed.

2.12 ATB data ordering

When ATB transfers are generated, the order of the data is modified from the original packet values. Channel, data, and timestamp values are split into nibbles and written in reverse nibble order, MS 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 writes are:

```
0x3216F413
0x21487654
0xXXXXXX43
```

Xs are filled by the next packet.

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

Table 2-6 ATB writes

ATB 32-bit write	ATB data nibbles	Source data nibbles
First word `0x3216F413`	[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]
	[7:0] = `0x1`	C8 data [7:4]
	[3:0] = `0x3`	C8 opcode [3:0]
Second word `0x21487654`	[31:28] = `0x2`	TS value [11:8]
	[27:24] = `0x1`	TS value [15:12]
	[23:20] = `0x4`	TS size [3:0]
	[19:16] = `0x8`	D32TS data [3:0]
	[15:12] = `0x7`	D32TS data [7:4]
	[11:8] = `0x6`	D32TS data [11:8]
	[7:0] = `0x5`	D32TS data [15:12]
	[3:0] = `0x4`	D32TS data [19:16]
Third word `0xXXXXX43`	[31:8] = `0xXXXXXX`	Filled by the next packet, or set to NULL
	[7:0] = `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 bit 0 in the Integration Mode Control Register. The STM interfaces available for topology detection are:

- one ATB master interface
- cross-trigger interface, **TRIGOUTSPTE**, **TRIGOUTSW**, **TRIGOUTHETE**, and **ASYNCOUT**.

You must disable the STM for correct functionality in integration mode. If integration mode is used in the CoreSight architecture, the system must be reset before being 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 1 master for hardware stimulus, therefore the M8 packet is sufficient and the M16 packet is never required.
D64, D64TS, D64M, D64MTS	Data	The STM does not generate 64-bit data packets.
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 which is not already accessible through APB reads and therefore 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 3-5.

This chapter is auto-generated from IP-XACT code.

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 0 by a system or power-on reset.
- 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 base offset order.

Table 3-1 STM register summary

Offset	Name	Type	Reset	Description
0xC04	STMDMASTARTR	WO	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xC08	STMDMASTOPR	WO	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xC0C	STMDMASTATR	RO	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xC10	STMDMACTLR	RW	0x00000000	*DMA Control Register* on page 3-5
0xCFC	STMDMAIDR	RO	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xD00	STMHEER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xD20	STMHETER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xD64	STMHEMCR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xDF4	STMHEMASTR	RO	0x00000080	*Hardware Event Master Number Register* on page 3-5
0xDF8	STMHEFEAT1R	RO	0x00100035	*Hardware Event Features 1 Register* on page 3-6
0xDFC	STMHEIDR	RO	0x00000001	*Hardware Event Features ID Register* on page 3-7
0xE00	STMSPER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE20	STMSPTER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE60	STMSPSCR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE64	STMSPMSCR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE68	STMSPOVERRIDER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE6C	STMSPMOVERRIDER	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE70	STMSPTRIGCSR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE80	STMTCSR	RW	0x00000004	*Trace Control and Status Register* on page 3-8
0xE84	STMTSSTIMR	WO	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE8C	STMTSFREQR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE90	STMSYNCR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xE94	STMAUXCR	RW	0x00000000	*Auxiliary Control Register* on page 3-9
0xEA0	STMSPFEAT1R	RO	0x006587D1	*STM Features 1 Register* on page 3-10
0xEA4	STMSPFEAT2R	RO	0x000104F2	*STM Features 2 Register* on page 3-12
0xEA8	STMSPFEAT3R	RO	0x0000007F	*STM Features 3 Register* on page 3-13
0xEE8	STMITTRIGGER	WO	-	*Integration Test for Cross-Trigger Outputs Register* on page 3-13
0xEEC	STMITATBDATA0	WO	-	*Integration Mode ATB Data 0 Register* on page 3-14
0xEF0	STMITATBCTR2	RO	-	*Integration Mode ATB Control 2 Register* on page 3-15
0xEF4	STMITATBID	WO	-	*Integration Mode ATB Identification Register* on page 3-16

Table 3-1 STM register summary (continued)

Offset	Name	Type	Reset	Description
0xEF8	STMITATBCTR0	WO	-	*Integration Mode ATB Control 0 Register* on page 3-17
0xF00	STMITCTRL	RW	0x00000000	*Integration Mode Control Register* on page 3-17
0xFA0	STMCLAIMSET	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xFA4	STMCLAIMCLR	RW	-	*System Trace Macrocell Programmers' Model Architecture Specification*
0xFB0	STMLAR	WO	-	*Lock Access Register* on page 3-18
0xFB4	STMLSR	RO	a	*Lock Status Register* on page 3-19
0xFB8	STMAUTHSTATUS	RO	0x000000AA	*Authentication Status Register* on page 3-20
0xFC8	STMDEVID	RO	0x00010000	*Device Configuration Register* on page 3-21
0xFCC	STMDEVTYPE	RO	0x00000063	*Device Type Identifier Register* on page 3-22
0xFE0	STMPIDR0	RO	0x00000062	*Peripheral ID0 Register* on page 3-22
0xFE4	STMPIDR1	RO	0x000000B9	*Peripheral ID1 Register* on page 3-23
0xFE8	STMPIDR2	RO	0x0000001B	*Peripheral ID2 Register* on page 3-24
0xFEC	STMPIDR3	RO	0x00000000	*Peripheral ID3 Register* on page 3-24
0xFD0	STMPIDR4	RO	0x00000004	*Peripheral ID4 Register* on page 3-25
0xFF0	STMCIDR0	RO	0x0000000D	*Component ID0 Register* on page 3-26
0xFF4	STMCIDR1	RO	0x00000090	*Component ID1 Register* on page 3-26
0xFF8	STMCIDR2	RO	0x00000005	*Component ID2 Register* on page 3-27
0xFFC	STMCIDR3	RO	0x000000B1	*Component ID3 Register* on page 3-28

a. This value depends on whether you are reading the register with **PADDRDBG31** HIGH or LOW.

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 Register characteristics are:

Purpose Controls the DMA transfer request mechanism.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xC10
- **Type** RW
- **Reset** 0x00000000
- **Width** 32

Figure 3-1 shows the STMDMACTLR Register bit assignments.

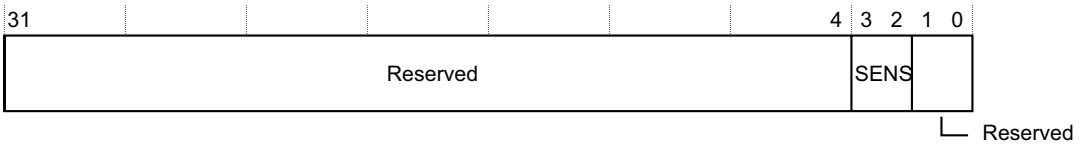


Figure 3-1 STMDMACTLR Register bit assignments

Table 3-2 shows the STMDMACTLR Register bit assignments.

Table 3-2 STMDMACTLR Register 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: b00 = Buffer is <25% b01 = Buffer is <50% b10 = Buffer is <75% full b11 = Buffer is <100% full.
[1:0]	Reserved	Reserved.

3.3.2 Hardware Event Master Number Register

The STMHEMASTR Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xDF4
Type RO
Reset 0x00000080
Width 32

Figure 3-2 shows the STMHEMASTR Register bit assignments.

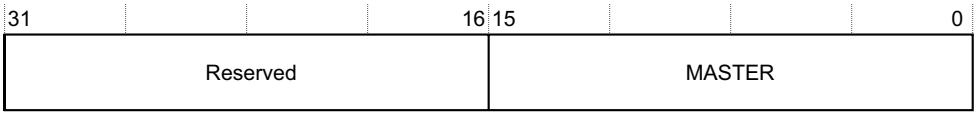


Figure 3-2 STMHEMASTR Register bit assignments

Table 3-3 shows the STMHEMASTR Register bit assignments.

Table 3-3 STMHEMASTR Register bit assignments

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

3.3.3 Hardware Event Features 1 Register

The STMHEFEAT1R Register characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xDF8
Type RO
Reset 0x00100035
Width 32

Figure 3-3 shows the STMHEFEAT1R Register bit assignments.

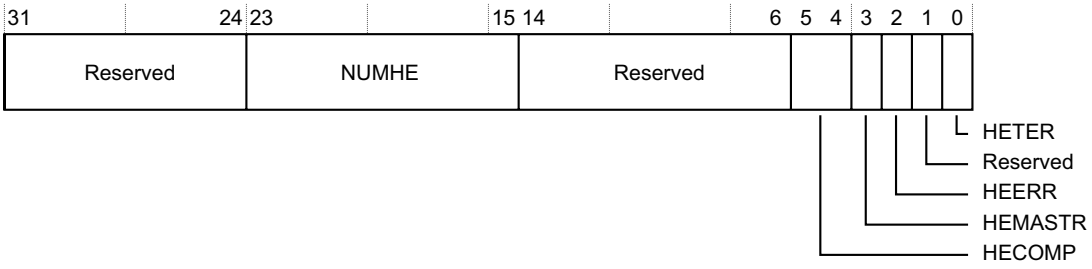


Figure 3-3 STMHEFEAT1R Register bit assignments

Table 3-4 shows the STMHEFEAT1R Register bit assignments.

Table 3-4 STMHEFEAT1R Register bit assignments

Bits	Name	Function
[31:24]	Reserved	Reserved.
[23:15]	NUMHE	The number of hardware events supported by the STM: b000100000 = 32 hardware events.
[14:6]	Reserved	Reserved.
[5:4]	HECOMP	Data compression on hardware event tracing support: b11 = Data compression support is programmable. STMHEMCR.COMPEN is implemented.
[3]	HEMASTR	STMHEMASTR support: b0 = STMHEMASTR is read-only.
[2]	HEERR	Hardware event error detection support: b1 = Hardware event error detection implemented. STMHEMCR.ERRDETECT is implemented.
[1]	Reserved	Reserved.
[0]	HETER	STMHETER support: b1 = STMHETER is implemented.

3.3.4 Hardware Event Features ID Register

The STMHEIDR Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xDFC
- **Type** RO
- **Reset** 0x00000001
- **Width** 32

Figure 3-4 shows the STMHEIDR Register bit assignments.

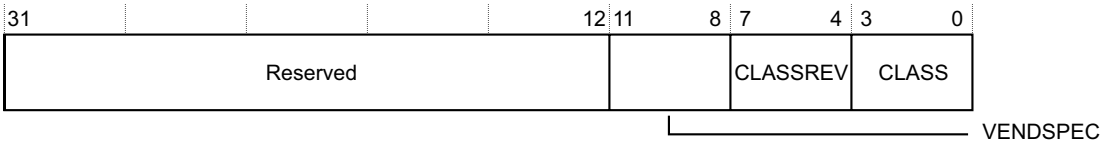


Figure 3-4 STMHEIDR Register bit assignments

Table 3-5 shows the STMHEIDR Register bit assignments.

Table 3-5 STMHEIDR Register bit assignments

Bits	Name	Function
[31:12]	Reserved	Reserved.
[11:8]	VENDSPEC	The VENDSPEC field identifies any vendor specific modifications or mappings: b0000 = Vendor specific information.
[7:4]	CLASSREV	The CLASSREV field identifies the revision of the programmers model: b0000 = Revision.
[3:0]	CLASS	The CLASS field identifies the programmers model: b0001 = Hardware Event Control.

3.3.5 Trace Control and Status Register

The STMTCSR Register characteristics are:

Purpose Controls the STM settings.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xE80
Type RW
Reset 0x00000004
Width 32

Figure 3-5 shows the STMTCSR Register bit assignments.

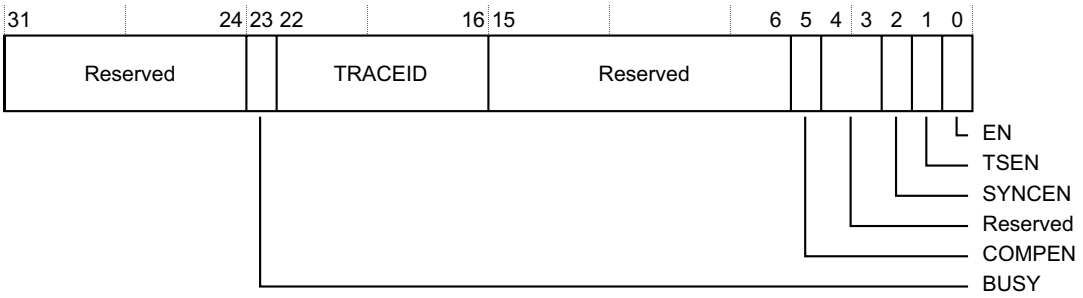


Figure 3-5 STMTCSR Register bit assignments

Table 3-6 shows the STMTCSR Register bit assignments.

Table 3-6 STMTCSR Register bit assignments

Bits	Name	Function
[31:24]	Reserved	Reserved.
[23]	BUSY	STM is busy, for example the STM trace FIFO is not empty: b0 = STM is not busy. b1 = STM is busy.
[22:16]	TRACEID	ATB Trace ID. Setting this value to all zeroes might result in Unpredictable tracing.

Table 3-6 STMTCSR Register bit assignments (continued)

Bits	Name	Function
[15:6]	Reserved	Reserved.
[5]	COMPEN	Compression Enable for Stimulus Ports: b0 = Compression disabled, data transfers are transmitted at the size of the transaction. b1 = Compression enabled, data transfers are compressed to save bandwidth.
[4:3]	Reserved	Reserved.
[2]	SYNCEN	STMSYNCR is implemented so this value is Read As One. b1 = STMSYNCR implemented.
[1]	TSEN	This bit controls if timestamp requests are ignored or not: b0 = Timestamping disabled. Requests for timestamp generation are ignored, and stimulus port writes selecting timestamping are treated as if it were not selected. b1 = Timestamping enabled. If stimulus writes select timestamping, a timestamp is output according to STPv2.
[0]	EN	Global STM enable: b0 = STM disabled b1 = STM enabled.

3.3.6 Auxiliary Control Register

The STMAUXCR Register characteristics are:

Purpose Used for implementation defined STM controls.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xE94
- **Type** RW
- **Reset** 0x00000000
- **Width** 32

Figure 3-6 shows the STMAUXCR Register bit assignments.

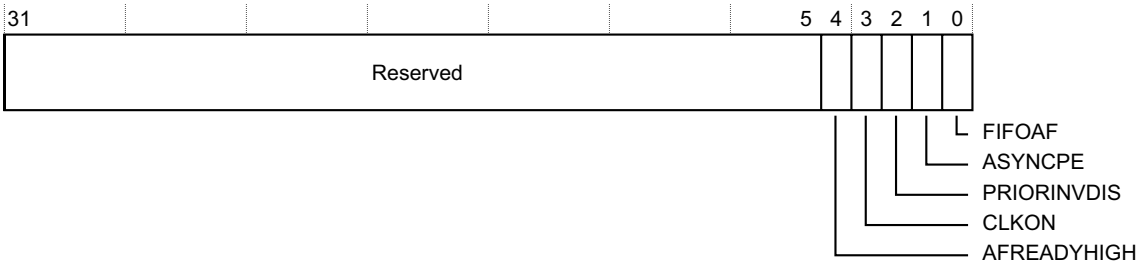


Figure 3-6 STMAUXCR Register bit assignments

Table 3-7 shows the STMAUXCR Register bit assignments.

Table 3-7 STMAUXCR Register bit assignments

Bits	Name	Function
[31:5]	Reserved	Reserved.
[4]	AFREADYHIGH	Provides override control for the **AFREADY** output: b0 = No override, **AFREADY** is controlled by the state of STM b1 = Override, **AFREADY** is driven HIGH. Reset value is b0.
[3]	CLKON	Provides override control for architectural clock gate enable: b0 = No override, clock gate is controlled by the state of STM b1 = Override, clock is enabled. Reset value is b0.
[2]	PRIORINVDIS	Controls arbitration between AXI and HW during flush: b0 = Priority inversion, when AXI flush is finished, HW gets priority until HW flush is done b1 = Priority inversion disabled, AXI always has priority over HW. Reset value is b0.
[1]	ASYNCPE	ASYNC priority: b0 = ASYNC priority is always lower than trace. b1 = ASYNC priority escalates on second synchronization request. Reset value is b0.
[0]	FIFOAF	Auto-flush: b0 = Auto-flush disabled. b1 = Auto-flush enabled. The STM automatically drains all data it has even if the ATB interface is not fully utilized. Reset value is b0.

ARM recommends that you leave the STMAUXCR Register at its default reset value.

3.3.7 STM Features 1 Register

The STMSPFEAT1R Register characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** `0xEA0`
- **Type** RO
- **Reset** `0x006587D1`
- **Width** 32

Figure 3-7 on page 3-11 shows the STMSPFEAT1R Register bit assignments.

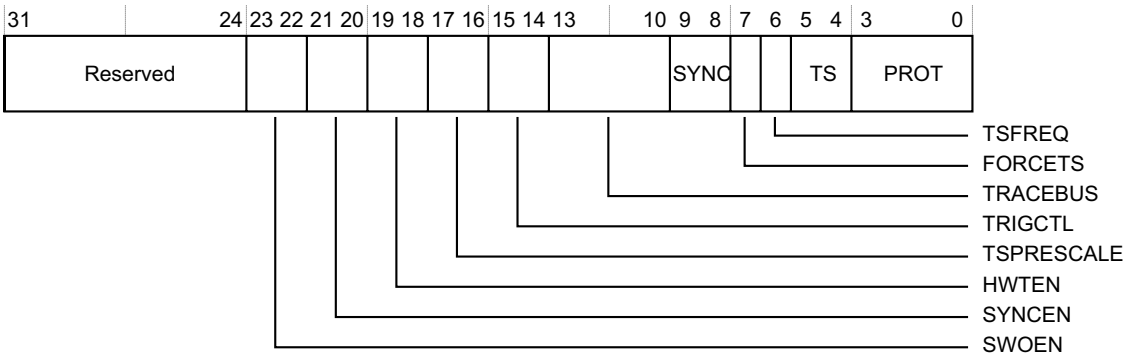


Figure 3-7 STMSPFEAT1R Register bit assignments

Table 3-8 shows the STMSPFEAT1R Register bit assignments.

Table 3-8 STMSPFEAT1R Register bit assignments

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

3.3.8 STM Features 2 Register

The STMSPFEAT2R Register characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEA4
Type RO
Reset 0x000104F2
Width 32

Figure 3-8 shows the STMSPFEAT2R Register bit assignments.

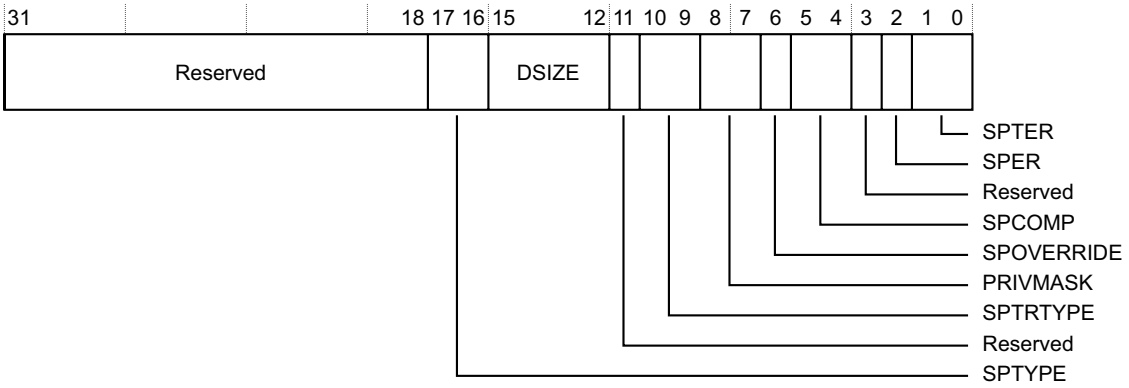


Figure 3-8 STMSPFEAT2R Register bit assignments

Table 3-9 shows the STMSPFEAT2R Register bit assignments.

Table 3-9 STMSPFEAT2R Register bit assignments

Bits	Name	Function
[31:18]	Reserved	Reserved.
[17:16]	SPTYPE	Stimulus port type support: b01 = Only extended stimulus ports are implemented.
[15:12]	DSIZE	Fundamental data size: b0000 = 32-bit data.
[11]	Reserved	Reserved.
[10:9]	SPTRTYPE	Stimulus port transaction type support: b10 = Both invariant timing and guaranteed transactions are supported.
[8:7]	PRIVMASK	STMPRIVMASKR support: b01 = STMPRIVMASKR not implemented.
[6]	SPOVERRIDE	STMSPOVERRIDER support: b1 = STMSPOVERRIDER and STMSPMOVERRIDER implemented.
[5:4]	SPCOMP	Data compression on stimulus ports support: b11 = Data compression support is programmable. STMTCSR.COMPEN is implemented.

Table 3-9 STMSPFEAT2R Register bit assignments (continued)

Bits	Name	Function
[3]	Reserved	Reserved.
[2]	SPER	STMSPER presence: b0 = STMSPER is implemented.
[1:0]	SPTER	STMSPTER support: b10 = STMSPTER is implemented.

3.3.9 STM Features 3 Register

The STMSPFEAT3R Register characteristics are:

Purpose Indicates the features of the STM.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEA8
Type RO
Reset 0x0000007F
Width 32

Figure 3-9 shows the STMSPFEAT3R Register bit assignments.

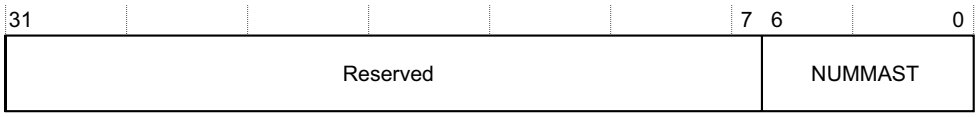


Figure 3-9 STMSPFEAT3R Register bit assignments

Table 3-10 shows the STMSPFEAT3R Register bit assignments.

Table 3-10 STMSPFEAT3R Register bit assignments

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

3.3.10 Integration Test for Cross-Trigger Outputs Register

The STMITTRIGGER Register characteristics are:

Purpose Integration Test for Cross-Trigger Outputs Register.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEE8
Type WO
Reset -

Width 32

Figure 3-10 shows the STMITTRIGGER Register bit assignments.

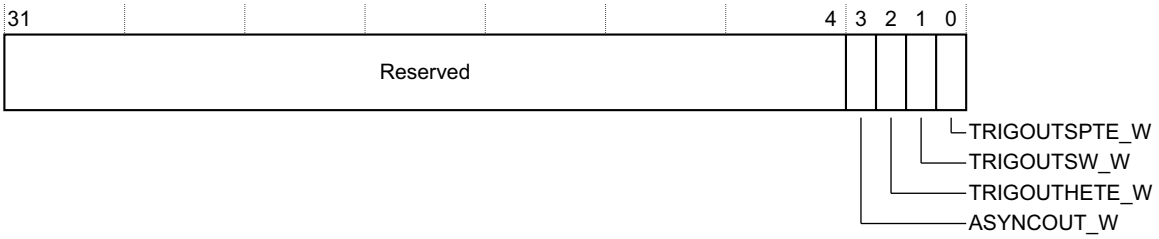


Figure 3-10 STMITTRIGGER Register bit assignments

Table 3-11 shows the STMITTRIGGER Register bit assignments.

Table 3-11 STMITTRIGGER Register bit assignments

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

3.3.11 Integration Mode ATB Data 0 Register

The STMITATBDATA0 Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEEC
Type WO
Reset -
Width 32

Figure 3-11 on page 3-15 shows the STMITATBDATA0 Register bit assignments.

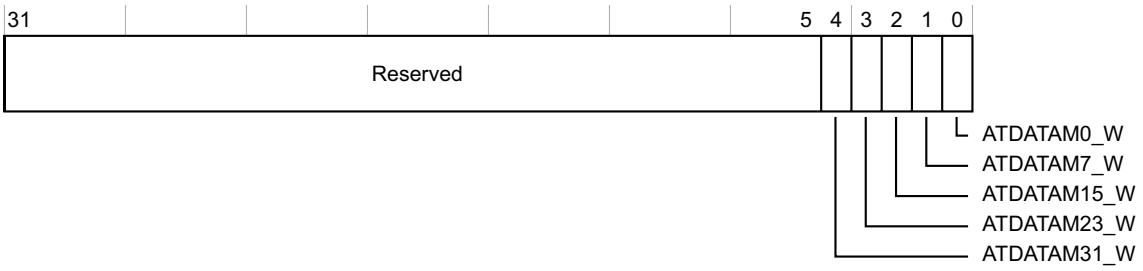


Figure 3-11 STMITATBDATA0 Register bit assignments

Table 3-12 shows the STMITATBDATA0 Register bit assignments.

Table 3-12 STMITATBDATA0 Register bit assignments

Bits	Name	Function
[31:5]	Reserved	Reserved.
[4]	ATDATAM31_W	Set the value of the **ATDATAM[31]** output: b1 = Drive logic 1 on **ATDATAM[31]** output. b0 = Drive logic 0 on **ATDATAM[31]** output.
[3]	ATDATAM23_W	Set the value of the **ATDATAM[23]** output: b1 = Drive logic 1 on **ATDATAM[23]** output. b0 = Drive logic 0 on **ATDATAM[23]** output.
[2]	ATDATAM15_W	Set the value of the **ATDATAM[15]** output: b1 = Drive logic 1 on **ATDATAM[15]** output. b0 = Drive logic 0 on **ATDATAM[15]** output.
[1]	ATDATAM7_W	Set the value of the **ATDATAM[7]** output: b1 = Drive logic 1 on **ATDATAM[7]** output. b0 = Drive logic 0 on **ATDATAM[7]** output.
[0]	ATDATAM0_W	Set the value of the **ATDATAM[0]** output: b1 = Drive logic 1 on **ATDATAM[0]** output. b0 = Drive logic 0 on **ATDATAM[0]** output.

3.3.12 Integration Mode ATB Control 2 Register

The STMITATBCTR2 Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** `0xEF0`
- **Type** RO
- **Reset** `0x00000000`
- **Width** 32

Figure 3-12 on page 3-16 shows the STMITATBCTR2 Register bit assignments.

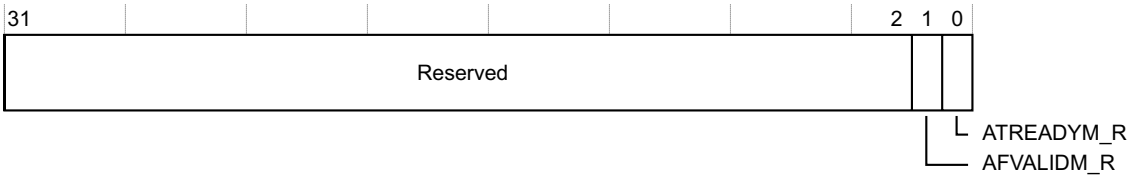


Figure 3-12 STMITATBCTR2 Register bit assignments

Table 3-13 shows the STMITATBCTR2 Register bit assignments.

Table 3-13 STMITATBCTR2 Register bit assignments

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

3.3.13 Integration Mode ATB Identification Register

The STMITATBID Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEF4
Type WO
Reset -
Width 32

Figure 3-13 shows the STMITATBID Register bit assignments.

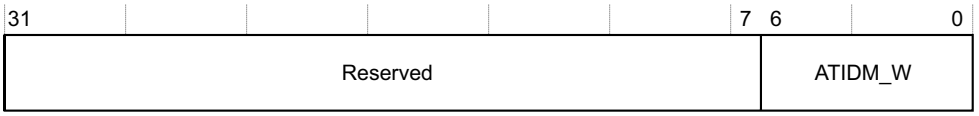


Figure 3-13 STMITATBID Register bit assignments

Table 3-14 shows the STMITATBID Register bit assignments.

Table 3-14 STMITATBID Register bit assignments

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

3.3.14 Integration Mode ATB Control 0 Register

The STMITATBCTR0 Register characteristics are:

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

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xEF8
Type WO
Reset -
Width 32

Figure 3-14 shows the STMITATBCTR0 Register bit assignments.

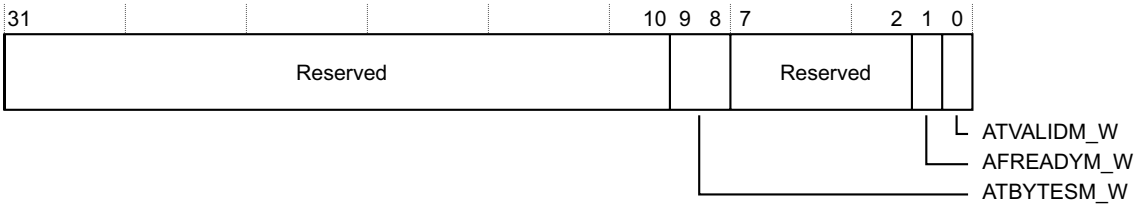


Figure 3-14 STMITATBCTR0 Register bit assignments

Table 3-15 shows the STMITATBCTR0 Register bit assignments.

Table 3-15 STMITATBCTR0 Register bit assignments

Bits	Name	Function
[31:10]	Reserved	Reserved.
[9:8]	ATBYTESM_W	Sets the value of the **ATBYTESM** output: b11 = Drive logic b11 on the **ATBYTESM** output. b10 = Drive logic b10 on the **ATBYTESM** output. b01 = Drive logic b01 on the **ATBYTESM** output. b00 = Drive logic b00 on **ATBYTESM** output.
[7:2]	Reserved	Reserved.
[1]	AFREADYM_W	Sets the value of the **AFREADYM** output: b1 = Drive logic 1 on the **AFREADYM** output. b0 = Drive logic 0 on the **AFREADYM** output.
[0]	ATVALIDM_W	Sets the value of the **ATVALIDM** output: b1 = Drive logic 1 on the **ATVALIDM** output. b0 = Drive logic 0 on the **ATVALIDM** output.

3.3.15 Integration Mode Control Register

The STMITCTRL Register characteristics are:

Purpose Used to enable topology detection. This register enables the component to switch from a functional mode, the default behavior, to integration mode where the inputs and outputs of the component can be directly controlled 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 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.

Attributes
- **Offset** 0xF00
- **Type** RW
- **Reset** 0x00000000
- **Width** 32

Figure 3-15 shows the STMITCTRL Register bit assignments.

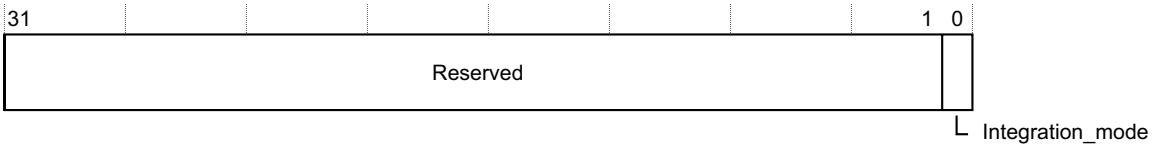


Figure 3-15 STMITCTRL Register bit assignments

Table 3-16 shows the STM ITCTRL Register bit assignments.

Table 3-16 STMITCTRL Register bit assignments

Bits	Name	Function
[31:1]	Reserved	Reserved.
[0]	Integration_mode	Enables the component to switch from functional mode to integration mode and back. If no integration functionality is implemented, this register must read as zero. b1 = Enable integration mode. b0 = Disable integration mode.

3.3.16 Lock Access Register

The STMLAR Register characteristics are:

Purpose This is used to enable write access to device registers.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xFB0
- **Type** WO
- **Reset** -
- **Width** 32

Figure 3-16 on page 3-19 shows the STMLAR Register bit assignments.

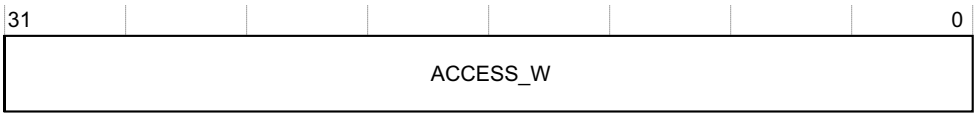


Figure 3-16 STMLAR Register bit assignments

Table 3-17 shows the STMLAR Register bit assignments.

Table 3-17 STMLAR Register bit assignments

Bits	Name	Function
[31:0]	ACCESS_W	A write of `0xC5ACCE55` enables further write access to this device. Any other write removes write access.

3.3.17 Lock Status Register

The STMLSR Register characteristics are:

Purpose This indicates the status of the lock control mechanism. This lock prevents accidental writes by code under debug. Accesses to the extended stimulus port registers are not affected by the lock mechanism. This register must always be present although there might not be any lock access control mechanism. The lock mechanism, where present and locked, must block write accesses to any control register, except the Lock Access Register. For most components this covers all registers except for the Lock Access Register.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** `0xFB4`
- **Type** RO
- **Reset** `0x00000003`
- **Width** 32

Figure 3-17 shows the STMLSR Register bit assignments.

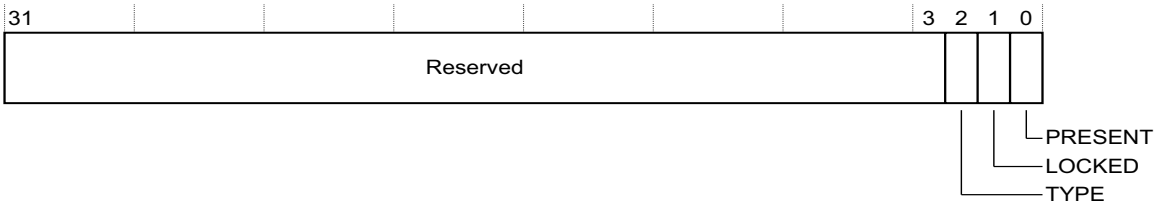


Figure 3-17 STMLSR Register bit assignments

Table 3-18 shows the STMLSR Register bit assignments.

Table 3-18 STMLSR Register bit assignments

Bits	Name	Function
[31:3]	Reserved	Reserved.
[2]	TYPE	Indicates if the Lock Access Register is implemented as 8-bit or 32-bit. b0 = This component implements a 32-bit Lock Access Register.
[1]	LOCKED	Returns the current status of the Lock. b0 = Write access is allowed to this device. b1 = Write access to the component is blocked. All writes to control registers are ignored. Reads are permitted.
[0]	PRESENT	Indicates that a lock control mechanism exists for this device. b0 = No lock control mechanism exists, writes to the Lock Access Register are ignored. b1 = Lock control mechanism is present.

3.3.18 Authentication Status Register

The STMAUTHSTATUS Register characteristics are:

Purpose Reports the required security level and current status of those enables. Where functionality changes on a given security level, this change in status must be reported in this register.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xFB8
- **Type** RO
- **Reset** The reset value depends on the authentication interface signals, which are system dependent.
- **Width** 32

Figure 3-18 shows the STMAUTHSTATUS Register bit assignments.

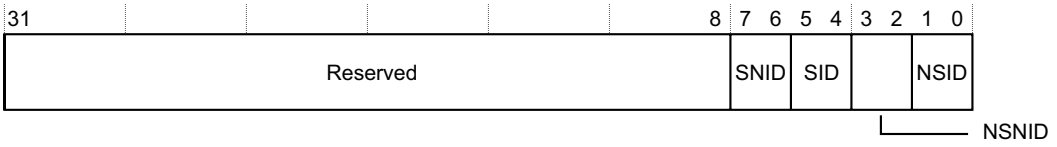


Figure 3-18 STMAUTHSTATUS Register bit assignments

Table 3-19 shows the STMAUTHSTATUS Register bit assignments.

Table 3-19 STMAUTHSTATUS Register bit assignments

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

3.3.19 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.

Attributes
Offset 0xFC8
Type RO
Reset 0x00010000
Width 32

Figure 3-19 shows the STMDEVID Register bit assignments.

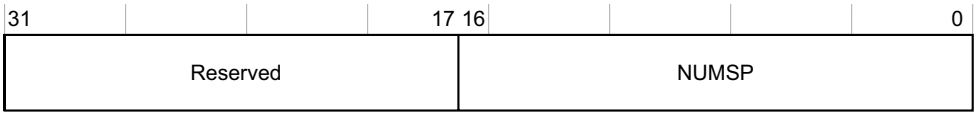


Figure 3-19 STMDEVID Register bit assignments

Table 3-20 shows the STMDEVID Register bit assignments.

Table 3-20 STMDEVID Register bit assignments

Bits	Name	Function
[31:17]	Reserved	Reserved.
[16:0]	NUMSP	This value indicates the number of stimulus ports implemented. 0x10000 = 65,536 stimulus ports implemented.

3.3.20 Device Type Identifier Register

The STMDEVTYPE Register characteristics are:

Purpose Provides a debugger with information about the component when the Part Number field 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.

Attributes
- **Offset** 0xFCC
- **Type** RO
- **Reset** 0x00000063
- **Width** 32

Figure 3-20 shows the STMDEVTYPE Register bit assignments.

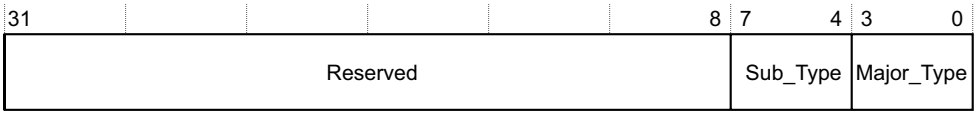


Figure 3-20 STMDEVTYPE Register bit assignments

Table 3-21 shows the STMDEVTYPE Register bit assignments.

Table 3-21 STMDEVTYPE Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	Sub_Type	Sub-classification within the major category: b0110 = This component generates trace based on software and hardware stimulus.
[3:0]	Major_Type	Major classification grouping for this debug or trace component: b0011 = This component has ATB output.

3.3.21 Peripheral ID0 Register

The STMPIDR0 Register 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.

Attributes
- **Offset** 0xFE0
- **Type** RO
- **Reset** 0x00000062
- **Width** 32

Figure 3-21 on page 3-23 shows the STMPIDR0 Register bit assignments.

Figure 3-21 STMPIDR0 Register bit assignments

Table 3-22 shows the STMPIDR0 Register bit assignments.

Table 3-22 STMPIDR0 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	Part_Number_bits7to0	Bits [7:0] of the component part number. This is selected by the designer of the component. b01100010 = Lowest 8 bits of the Part Number, `0x962`.

3.3.22 Peripheral ID1 Register

The STMPIDR1 Register 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.

Attributes
- **Offset** `0xFE4`
- **Type** RO
- **Reset** `0x000000B9`
- **Width** 32

Figure 3-22 shows the STMPIDR1 Register bit assignments.

Figure 3-22 STMPIDR1 Register bit assignments

Table 3-23 shows the STMPIDR1 Register bit assignments.

Table 3-23 STMPIDR1 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	JEP106_bits3to0	Bits [3:0] of the JEDEC identity code indicating the designer of the component, together with the continuation code. b1011 = Lowest 4 bits of the JEP106 Identity Code.
[3:0]	Part_Number_bits11to8	Bits [11:8] of the component part number. This is selected by the designer of the component. b1001 = Upper 4 bits of the Part Number, `0x962`.

3.3.23 Peripheral ID2 Register

The STMPIDR2 Register 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.

Attributes
- **Offset** 0xFE8
- **Type** RO
- **Reset** 0x0000001B
- **Width** 32

Figure 3-23 shows the STMPIDR2 Register bit assignments.

Figure 3-23 STMPIDR2 Register bit assignments

Table 3-24 shows the STMPIDR2 Register bit assignments.

Table 3-24 STMPIDR2 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	Revision	The Revision field is an incremental value starting at 0x0 for the first design of this component. This only increases by 1 for both major and minor revisions and is used as a look-up to establish the exact major and minor revision. b0001 = This device is at r0p1.
[3]	JEDEC	Always set. Indicates that a JEDEC assigned value is used. b1 = The designer ID is specified by JEDEC (http://www.jedec.org).
[2:0]	JEP106_bits6to4	Bits [6:4] of the JEDEC identity code indicating the designer of the component, together with the continuation code. b011 = Upper 3 bits of the JEP106 Identity Code.

3.3.24 Peripheral ID3 Register

The STMPIDR3 Register characteristics are:

Purpose Part of the set of Peripheral Identification registers. Contains the RevAnd and Customer Modified fields.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
- **Offset** 0xFEC
- **Type** RO
- **Reset** 0x00000000

Width 32

Figure 3-24 shows the STMPIDR3 Register bit assignments.

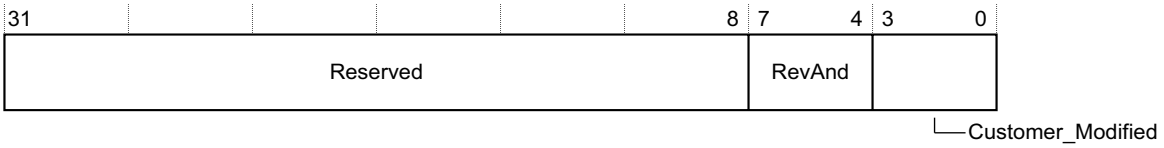


Figure 3-24 STMPIDR3 Register bit assignments

Table 3-25 shows the STMPIDR3 Register bit assignments.

Table 3-25 STMPIDR3 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	RevAnd	This field indicates minor errata fixes specific to this design, for example metal fixes after implementation. In most cases this field is zero. It is recommended that component designers ensure this field can be changed by a metal fix if required, for example by driving it from registers that reset to zero. `0x0` = Indicates that there have been no metal fixes to this component.
[3:0]	Customer_Modified	Where the component is reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is zero. `0x0` = Indicates that there have been no modifications made.

3.3.25 Peripheral ID4 Register

The STMPIDR4 Register 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.

Attributes
Offset `0xFD0`
Type RO
Reset `0x00000004`
Width 32

Figure 3-25 shows the STMPIDR4 Register bit assignments.

Figure 3-25 STMPIDR4 Register bit assignments

Table 3-26 shows the STMPIDR4 Register bit assignments.

Table 3-26 STMPIDR4 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:4]	FourKB_Count	This is a 4-bit value that indicates the total contiguous size of the memory window used by this component in powers of 2 from the standard 4KB. If a component only requires the standard 4KB, this 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` = Indicates that the device only occupies 4KB of memory.
[3:0]	JEP106_cont	JEDEC continuation code indicating the designer of the component, together with the identity code. `0x4` = Indicates that ARM's JEDEC identity code is on the 5th bank.

3.3.26 Component ID0 Register

The STMCIDR0 Register characteristics are:

Purpose A component identification register, that indicates that the identification registers are present.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset `0xFF0`
Type RO
Reset `0x0000000D`
Width 32

Figure 3-26 shows the STMCIDR0 Register bit assignments.

Figure 3-26 STMCIDR0 Register bit assignments

Table 3-27 shows the STMCIDR0 Register bit assignments.

Table 3-27 STMCIDR0 Register bit assignments

Bits	Name	Function
[31:8]	Reserved	Reserved.
[7:0]	Preamble	Contains bits [24:31] of the component identification. `0x0D` = Identification value.

3.3.27 Component ID1 Register

The STMCIDR1 Register characteristics are:

Purpose A component identification register, that indicates that the identification registers are present. This register also indicates the component class.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xFF4
Type RO
Reset 0x00000090
Width 32

Figure 3-27 shows the STMCIDR1 Register bit assignments.

Figure 3-27 STMCIDR1 Register bit assignments

Table 3-28 shows the STMCIDR1 Register bit assignments.

Table 3-28 STMCIDR1 Register bit assignments

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

3.3.28 Component ID2 Register

The STMCIDR2 Register characteristics are:

Purpose A component identification register, that indicates that the identification registers are present.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset 0xFF8
Type RO
Reset 0x00000005
Width 32

Figure 3-28 shows the STMCIDR2 Register bit assignments.

Figure 3-28 STMCIDR2 Register bit assignments

Table 3-29 shows the STMCIDR2 Register bit assignments.

Table 3-29 STMCIDR2 Register bit assignments

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

3.3.29 Component ID3 Register

The STMCIDR3 Register characteristics are:

Purpose A component identification register, that indicates that the identification registers are present.

Usage constraints There are no usage constraints.

Configurations This register is available in all configurations.

Attributes
Offset `0xFFC`
Type RO
Reset `0x000000B1`
Width 32

Figure 3-29 shows the STMCIDR3 Register bit assignments.

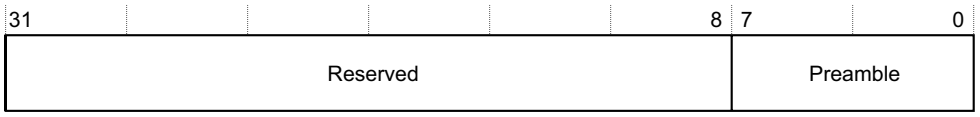


Figure 3-29 STMCIDR3 Register bit assignments

Table 3-30 shows the STMCIDR3 Register bit assignments.

Table 3-30 STMCIDR3 Register bit assignments

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

Appendix A
Signal Descriptions

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

- *Signal descriptions* on page A-2.

A.1 Signal descriptions

The following sections describe the STM signals:

- *Clocks and resets*
- *AXI slave*
- *Debug APB interface* on page A-4
- *ATB master interface* on page A-4
- *Hardware event observation interface signals* on page A-4
- *DMA peripheral request interface signals* on page A-5
- *Timestamp port signals* on page A-5
- *Authentication interface signals* on page A-6
- *Non-secure guaranteed interface signals* on page A-6
- *Cross-trigger interface signals* on page A-6
- *External synchronization request interface signals* on page A-6
- *Test interface signals* on page A-7.

A.1.1 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 and DMA peripheral request interface.
STMRESETn	Input	STM reset. Resets all modules in the STM except the AXI slave and DMA peripheral request interface.

A.1.2 AXI slave

Table A-2 shows the AXI slave signals.

Table A-2 AXI slave signals

Signal name	Type	Description
ARIDS[AXI_ID_WIDTH–1:0]	Input	Read address ID
ARADDRS[31:0]	Input	Read address
ARLENS[3:0]	Input	Read address burst length
ARSIZES[2:0]	Input	Read address burst size
ARBURSTS[1:0]	Input	Read address burst type
ARLOCKS[1:0]	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

Table A-2 AXI slave signals (continued)

Signal name	Type	Description
RIDS[AXI_ID_WIDTH–1:0]	Output	Read ID
RDATAS[31:0]	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 address
AWLENS[3:0]	Input	Write address burst length
AWSIZES[2:0]	Input	Write address burst size
AWBURSTS[1:0]	Input	Write address burst type
AWLOCKS[1:0]	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
WIDS[AXI_ID_WIDTH–1:0]	Input	Write data ID
WDATAS[31:0]	Input	Write data
WSTRBS[3:0]	Input	Write data strobes
WLASTS	Input	Write last
WVALIDS	Input	Write valid
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.1.3 Debug APB interface

Table A-3 shows the debug APB interface signals.

Table A-3 Debug APB interface signals

Signal name	Type	Description
PCLKENDBG	Input	APB clock enable.
PSELDBG	Input	Select.
PENABLEDBG	Input	Enable.
PWRITEDBG	Input	Peripheral write.
PADDRDBG[11:2]	Input	Address. Only word-aligned addresses are supported so bits [1:0] are assumed 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. This output is permanently driven LOW in the STM.

A.1.4 ATB master interface

This is the interface through which the trace data is output for collection. Connect it 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[1:0]	Output	Number of bytes on **ATDATA** to be captured, minus 1.
ATDATAM[31:0]	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.

A.1.5 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[31:0]	Input	Hardware event observation port. Each hardware event to be observed must be connected to one bit of this bus. Only rising edges of these signals indicate the event has occurred.

A.1.6 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. This output is permanently driven LOW in the STM.

See the *AMBA DMA Controller DMA-330 Technical Reference Manual* for more information about these signals.

A.1.7 Timestamp port signals

This is the interface where the STM receives timestamp information from the system. Table A-7 shows the timestamp port signals.

Table A-7 Timestamp port signals

Signal name	Type	Description
TSVALUE[63:0]	Input	Timestamp value. This port is always 64 bits wide. If **TSMAXWIDTH** is b0, bits [63:48] of the timestamp are ignored.
TSNATURAL	Input	Timestamp encoding. Tie this to a value appropriate to the system timestamp encoding. Values: b0 = Reflected binary gray-code timestamp encoding b1 = Natural binary timestamp encoding.
TSMAXWIDTH	Input	Maximum timestamp width in STM timestamp packets. This should be tied by the system designer to a value appropriate to the system timestamp counter width. Values: b0 = 48-bit maximum timestamp width b1 = 64-bit maximum timestamp width. If the actual width of the timestamp input to the STM is less than the selected maximum, the remaining upper bits must be tied to zero and not left undriven.

A.1.8 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.1.9 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.1.10 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. This signal is asserted for one clock cycle when a trigger event is detected on match using STMSPTER.
TRIGOUTSW	Output	Trigger output. This signal is asserted 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. This signal is asserted for one clock cycle when a trigger event is detected on match using STMHETER.
ASYNCOUT	Output	Alignment synchronization output. This signal is asserted for one clock cycle when an ASYNC-VERSION-FREQ sequence is completely output on the ATB, and can be used for cross-triggering.

A.1.11 External synchronization request interface signals

Table A-11 shows the external synchronization request interface signals.

Table A-11 External synchronization request interface signals

Signal name	Type	Description
SYNCREQ	Input	Trace synchronization request. This is used to indicate that the STM must insert an ASYNC-VERSION sequence into the trace output stream.

A.1.12 Test interface signals

Table A-12 shows the test interface signals.

Table A-12 Test interface signals

Signal name	Type	Description
DFTTESTMODE	Input	Test mode. This signal is used to force the STM clock on in test mode.
DFTCLKDISABLE	Input	Test clock disable. This signal is used to switch off the STM clock in capture phase of DFT test of other components in the system.

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	-	-

Table B-2 Differences between Issue A and Issue B

Change	Location	Affects
Added STM configuration list	Table 1-1 on page 1-5	All
Deleted sections on *Unaligned transfers* and *Unaligned data in writes and use of WSTRBS*	*STM enabled* on page 2-12	r0p1
Added section on **WSTRBS** usage	*WSTRB usage* on page 2-14	r0p1
Added section on **AWSIZE** usage	*AWSIZE usage* on page 2-15	r0p1
Added section on DMA interface behavior	*DMA interface behavior* on page 2-19	r0p1
Updated STMPIDR2 reset value	Table 3-1 on page 3-3 and *Peripheral ID2 Register* on page 3-24	r0p1
Updated revision value	Table 3-24 on page 3-24	r0p1

Glossary

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

Advanced eXtensible Interface (AXI)
A bus protocol that supports separate address/control and data phases, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels to enable low-cost DMA, ability to issue multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

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

AXI is targeted at high performance, high clock frequency system designs and includes a number of features that make it very suitable for high speed sub-micron interconnect.

Advanced Microcontroller Bus Architecture (AMBA)
A family of protocol specifications that describe a strategy for the interconnect. AMBA is the ARM open standard for on-chip buses. It is an on-chip bus specification that describes a strategy for the interconnection and management of functional blocks that make up a *System-on-Chip* (SoC). It aids in the development of embedded processors with one or more CPUs or signal processors and multiple peripherals. AMBA complements a reusable design methodology by defining a common backbone for SoC modules.

Advanced Peripheral Bus (APB)
A simpler bus protocol than AXI and AHB. It is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. Connection to the main system bus is through a system-to-peripheral bus bridge that helps to reduce system power consumption.

Aligned A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.

AMBA *See* Advanced Microcontroller Bus Architecture.

Advanced Trace Bus (ATB)
A bus used by trace devices to share CoreSight capture resources.

APB *See* Advanced Peripheral Bus.

ATB *See* Advanced Trace Bus.

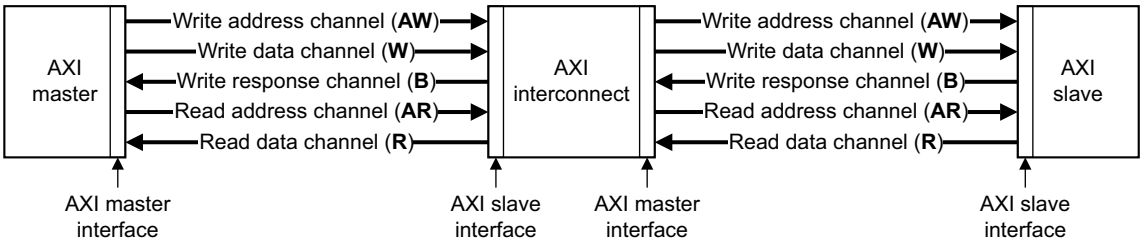
ATB bridge A synchronous ATB bridge provides a register slice to facilitate timing closure through the addition of a pipeline stage. It also provides a unidirectional link between two synchronous ATB domains.

An asynchronous ATB bridge provides a unidirectional link between two ATB domains with asynchronous clocks. It is intended to support connection of components with ATB ports residing in different clock domains.

AXI *See* Advanced eXtensible Interface.

AXI channel order and interfaces
The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.



AXI terminology The following AXI terms are general. They apply to both masters and slaves:

Active read transaction

A transaction for which the read address has transferred, but the last read data has not yet transferred.

Active transfer

A transfer for which the **xVALID**[1] handshake has asserted, but for which **xREADY** has not yet asserted.

1. The letter **x** in the signal name denotes an AXI channel as follows:

AW	Write address channel.
W	Write data channel.
B	Write response channel.
AR	Read address channel.
R	Read data channel.

Active write transaction

A transaction for which the write address or leading write data has transferred, but the write response has not yet transferred.

Completed transfer

A transfer for which the **xVALID**/**xREADY** handshake is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address, one or more data transfers and a response transfer (writes only).

Transmit An initiator driving the payload and asserting the relevant **xVALID** signal.

Transfer A single exchange of information. That is, with one **xVALID**/**xREADY** handshake.

The following AXI terms are master interface attributes. To obtain optimum performance, they must be specified for all components with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that a master interface can generate. It is specified for master interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read issuing capabilities.

Read ID capability

The maximum number of different **ARID** values that a master interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that a master interface can generate.

Write ID capability

The maximum number of different **AWID** values that a master interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the master interface is capable of transmitting data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate.

The following AXI terms are slave attributes. To obtain optimum performance, they must be specified for all components with an AXI slave:

Combined acceptance capability

The maximum number of active transactions that a slave interface can accept. It is specified for slave interfaces that use combined storage for active write and read transactions. If not specified then it is assumed to be equal to the sum of the write and read acceptance capabilities.

Read acceptance capability

The maximum number of active read transactions that a slave interface can accept.

Read data reordering depth

The number of active read transactions for which a slave interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that a slave interface can accept.

Write interleave depth

The number of active write transactions for which the slave interface can receive data. This is counted from the earliest transaction.

Beat Alternative word for an individual transfer within a burst. For example, an INCR4 burst comprises four beats.

See also Burst.

Burst A group of transfers to consecutive addresses. Because the addresses are consecutive, there is no requirement to supply an address for any of the transfers after the first one. This increases the speed at which the group of transfers can occur. Bursts over AMBA are controlled using signals to indicate the length of the burst and how the addresses are incremented.

See also Beat.

Byte lane strobe A signal that is used for unaligned or mixed-endian data accesses to determine which byte lanes are active in a transfer. One bit of this signal corresponds to eight bits of the data bus.

Clock gating Gating a clock signal for a macrocell with a control signal and using the modified clock that results to control the operating state of the macrocell.

Cross Trigger Interface (CTI)
Part of an Embedded Cross Trigger device. The CTI provides the interface between a processor/ETM and the CTM within an ECT.

Cross Trigger Matrix (CTM)
The CTM combines the trigger requests generated from CTIs and broadcasts them to all CTIs as channel triggers within an Embedded Cross Trigger device.

CTI *See* Cross Trigger Interface.

CTM *See* Cross Trigger Matrix.

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

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

Direct Memory Access (DMA)
An operation that accesses main memory directly, without the processor performing any accesses to the data concerned.

DMA *See* Direct Memory Access.

Gray code Continuous binary code in which only one bit changes for a change to the next state up or down.

Implementation-defined
The behavior is not architecturally defined, but is defined and documented by individual implementations.

Implementation-specific
The behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.

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

Processor A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.

RealView ICE A system for debugging embedded processor cores that uses a JTAG interface.

Reserved A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.

SBO *See* Should Be One.

SBZ *See* Should Be Zero.

SBZP *See* Should Be Zero or Preserved.

Scan chain A scan chain is made up of serially-connected devices that implement boundary scan technology using a standard JTAG TAP interface. Each device contains at least one TAP controller containing shift registers that form the chain connected between **TDI** and **TDO**, through which test data is shifted. Processors can contain several shift registers to enable you to access selected parts of the device.

Should Be One (SBO) Write as 1, or all 1s for bit fields, by software. Writing as 0 produces Unpredictable results.

Should Be Zero (SBZ) Write as 0, or all 0s for bit fields, by software. Writing as 1 produces Unpredictable results.

Should Be Zero or Preserved (SBZP)
Write as 0, or all 0s for bit fields, by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.

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

UNP *See* Unpredictable.

Unpredictable Means that the behavior of the STM cannot be relied on. Such conditions have not been validated. When applied to the programming of an event resource, only the output of that event resource is Unpredictable. Unpredictable behavior can affect the behavior of the entire system.